SCORM® 2004 3rd EDITION

Sharable Content Object Reference Model

Content Aggregation Model

NOVEMBER 16, 2006
VERSION 1.0

Advanced Distributed Learning

This page intentionally left blank.

Advanced Distributed Learning (ADL)

SCORM® 2004 3rd Edition Content Aggregation Model (CAM) Version 1.0

Available at ADLNet.gov

For questions and comments visit Ask The Experts at ADLNet.gov

This page intentionally left blank.

Chief Technical Architect
Philip Dodds

Technical Editor
Schawn E. Thropp

Acknowledgements

ADL would like to thank the following organizations and their members for their continued commitment to building interoperable e-learning standards and specifications:

Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE) (http://www.ariadne-eu.org/)

Aviation Industry CBT Committee (AICC) (http://www.aicc.org/)

Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC) (http://ltsc.ieee.org/)

IMS Global Learning Consortium, Inc. (http://www.imsglobal.org/)

ADL would also like to thank the ADL Community for their commitment and contribution to the evolution of SCORM.

Table of Contents

SECTION 1 SCORM® CONTENT AGGREGATION MODEL (CAM) OVERVIEW 1-1
1.1. INTRODUCTION TO THE SCORM CONTENT AGGREGATION MODEL (CAM) BOOK 1-3
1.1.1. What is Covered in the SCORM Content Aggregation Model (CAM) Book? 1-3
1.1.2. Using the SCORM CAM Book 1-4
1.1.3. Relationship with Other SCORM Books 1-5
1.2. THE SCORM CONTENT AGGREGATION MODEL 1-8
SECTION 2 THE SCORM® CONTENT MODEL 2-1
2.1. SCORM CONTENT MODEL COMPONENTS 2-3
2.1.1. Asset 2-3
2.1.2. Sharable Content Object (SCO) 2-4
2.1.3. Activities 2-5
2.1.4. Content Organization 2-6
2.1.5. Content Aggregation 2-7
SECTION 3 SCORM® CONTENT PACKAGING 3-1
3.1. CONTENT PACKAGING OVERVIEW 3-3
3.2. CONTENT PACKAGE COMPONENTS 3-4
3.2.1. Package 3-5
3.2.2. Manifest 3-5
3.2.3. Package Interchange File (PIF) 3-6
3.3. COMPONENTS OF A MANIFEST 3-7
3.3.1. Metadata 3-7
3.3.2. Organizations 3-8
3.3.3. Resources 3-15
3.3.4. Content 3-16
3.4. BUILDING CONTENT PACKAGES 3-17
3.4.1. Manifest File 3-17
3.4.2. Content Package Manifest Extensions 3-45
3.4.3. Content Package Manifest Href Handling 3-45
3.5. SCORM CONTENT PACKAGE APPLICATION PROFILES 3-52
3.5.1. Resource Content Package 3-52
3.5.2. Content Aggregation Content Package 3-55
3.5.3. SCORM Content Package Application Profile Requirements 3-57
3.6. BEST PRACTICES AND PRACTICAL GUIDELINES 3-59
3.6.1. Multiple Organizations for a Single Course 3-59
3.6.2. Using the <dependency> Element 3-59
SECTION 4 METADATA 4-1
4.1. METADATA OVERVIEW 4-3
4.2. LOM METADATA CREATION 4-5
4.2.1. <lom> Element 4-7
4.2.2. <general> Element 4-9
4.2.3. <lifeCycle> Element 4-18
4.2.4. <metaMetadata> Element 4-25
4.2.5. <technical> Element 4-34
4.2.6. <educational> Element 4-46
4.2.7. <rights> Element 4-56
4.2.8. <relation> Element 4-59
4.2.9. <annotation> Element 4-65
4.2.10. <classification> Element 4-68
4.2.11. Common Data Types 4-76

4.3. LOM XML SCHEMA VALIDATION APPROACHES 4-81
4.3.1. Strict Schema Validation Approach 4-81
4.3.2. Custom Schema Validation Approach 4-82
4.3.3. Loose Schema Validation Approach 4-82
4.4. METADATA EXTENSIONS 4-83
4.4.1. Data Element Extension 4-84
4.4.2. Vocabulary Extension 4-85
4.5. METADATA AND SCORM CONTENT MODEL COMPONENTS 4-86
4.5.1. Metadata Describing Content Aggregations 4-87
4.5.2. Metadata Describing Content Organizations 4-88
4.5.3. Metadata Describing Activities 4-89
4.5.4. Metadata Describing SCOs 4-90
4.5.5. Metadata Describing Assets 4-91
SECTION 5 SCORM® SEQUENCING AND PRESENTATION 5-1
5.1. SEQUENCING AND PRESENTATION 5-3
5.1.1. <sequencing> Element 5-3
5.1.2. <controlMode> Element 5-5
5.1.3. <sequencingRules> Element 5-7
5.1.4. <limitConditions> Element 5-16
5.1.5. <auxiliaryResources> Element 5-18
5.1.6. <rollupRules> Element 5-19
5.1.7. <objectives> Element 5-26
5.1.8. <randomizationControls> Element 5-34
5.1.9. <deliveryControls> Element 5-36
5.1.10. <constrainedChoiceConsiderations> Element 5-37
5.1.11. <rollupConsiderations> Element 5-39
5.1.12. <sequencingCollection> Element 5-41
5.2. PRESENTATION/NAVIGATION INFORMATION 5-43
5.2.1. <presentation> Element 5-43
5.3. RELATIONSHIP TO CONTENT PACKAGING 5-47
APPENDIX A ACRONYM LISTING A-1
ACRONYM LISTING A-3
APPENDIX B REFERENCES B-1
REFERENCES B-3
APPENDIX C DOCUMENT REVISON HISTORY C-1
DOCUMENT REVISION HISTORY C-3

List of Figures

Figure 1.1a: The SCORM Content Aggregation Model Book as Part of the SCORM Bookshelf 1-3
Figure 2.1.1a: Examples of Assets 2-3
Figure 2.1.2a: Conceptual Makeup of a SCO 2-4
Figure 2.1.3a: Conceptual Representation of Activities 2-5
Figure 2.1.4a: Conceptual Illustration of a Content Organization 2-6
Figure 2.1.5a: Conceptual Illustration of a Content Aggregation 2-8
Figure 3.2a: Content Package Conceptual Diagram 3-4
Figure 3.3a: Components of a Manifest 3-7
Figure 3.3.2.3.1a: IMS Content Hierarchy Terminology 3-12
Figure 3.3.3a: Conceptual Illustration of Manifest Resources 3-15
Figure3.5.1a: Example of an Asset Represented as a <file> Element 3-53
Figure 3.5.1b: Example of an Asset Represented as a <resource> Element 3-54
Figure 3.5.1c: Example of a SCO Represented as a <resource> Element 3-55
Figure 3.5.2a: Example of a Content Aggregation Content Package and its Components 3-56
Figure 4.5.1a: Application of Metadata Describing a Content Aggregation 4-87
Figure 4.5.2a: Application of Metadata Describing a Content Organization 4-88
Figure 4.5.13a: Application of Metadata Describing an Activity 4-89
Figure 4.5.4a: Application of Metadata Describing a SCO 4-90
Figure 4.5.5a: Application of Metadata Describing Assets 4-91
Figure 5.3a: Sequencing Rules and Content Packaging Structure Relationship 5-47

List of Tables

Table 3.3.2.3a: Example of Curricular Taxonomy Models 3-11
Table 3.4.1a: SCORM Content Packaging Application Profile Table Format 3-18
Table 3.4.1b: Explanation of Content Packaging Application Profile Multiplicity Requirements 3-18
Table 3.5.3a: SCORM Content Package Application Profile Manifest Element Requirements 3-57
Table 4.2a: Explanation of Multiplicity Requirements 4-5

This page intentionally left blank.

SECTION 1
SCORM® Content Aggregation Model (CAM) Overview

This page intentionally left blank.

1.1. Introduction to the SCORM Content Aggregation Model (CAM) Book

The Sharable Content Object Reference Model (SCORM) is often described as a set of books on a bookshelf. The Content Aggregation Model (CAM) book is one of a set of books (refer to Figure 1.1a: *The Content Aggregation Model Book as Part of the SCORM Bookshelf*). More information on the other SCORM books and their relationships to one another can be found in the SCORM 2004 3rd Edition Overview book. The SCORM CAM book describes the components used in a learning experience, how to package those components for exchange from system to system, how to describe those components to enable search and discovery and how to define sequencing information for the components. The SCORM CAM promotes the consistent storage, labeling, packaging, exchange and discovery of learning content.

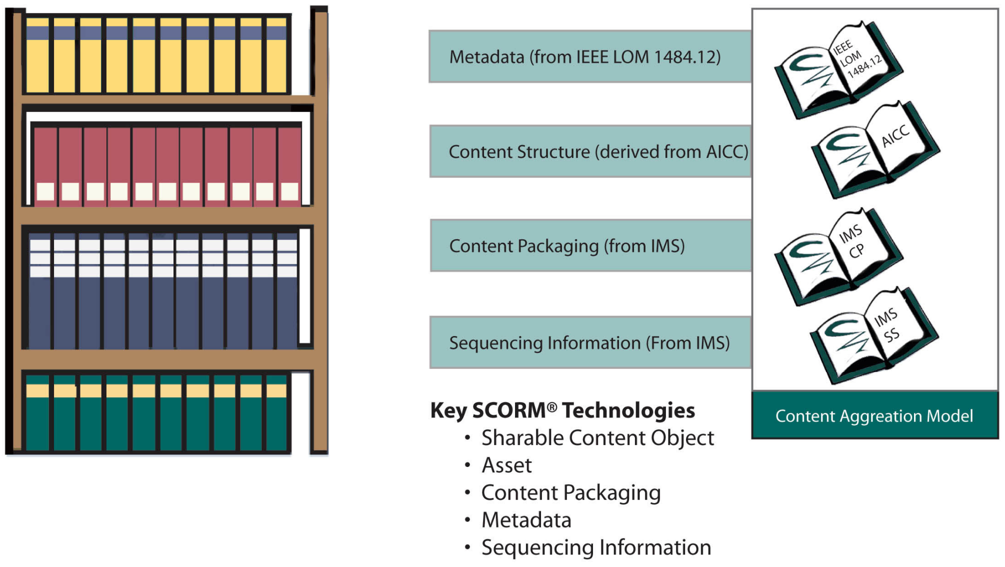

Metadata (from IEEE LOM 1484.12)
Content Structure (derived from AICC)
Content Packaging (from IMS)
Sequencing Information (From IMS)
IEEE LOM 1484.12
AICC
IMS CP
IMS SS
Content Aggregation Model
Key SCORM® Technologies
• Sharable Content Object
• Asset
• Content Packaging
• Metadata
• Sequencing Information

Figure 1.1a: The SCORM Content Aggregation Model Book as Part of the SCORM Bookshelf

1.1.1. What is Covered in the SCORM Content Aggregation Model (CAM) Book?

There are several key concepts that are introduced in the SCORM CAM book. The book describes responsibilities and requirements for building content and content organizations (e.g., course, lessons, modules, etc.). The book contains information on creating content packages, applying metadata to the components in the content package and applying sequencing and navigation details in the context of a content package. SCORM Content Packaging, as described in this book, provides a consistent form for describing content structures, learning content, the metadata that describes the various components of the content structures and sequencing and navigation rules. This consistency facilitates search and discovery of content packages and their resources (helping facilitate reuse of SCORM conformant content), building of content organizations that will behave in a similar manner from system to system and standard understanding of the contents of the content package. General subjects discussed include:

- Content Model: Definition of common terminology used throughout the CAM book.
- Content Packaging: Descriptions and requirements for aggregating and bundling learning content.
- Metadata: Descriptions and requirements for describing SCORM components.
- Sequencing and Navigation: Descriptions and requirements for defining sequencing and navigation information.

1.1.2. Using the SCORM CAM Book

This book will assist authoring tool vendors, content developers and anyone else wishing to create, edit or use:

- SCORM Content Model Components (Assets, Sharable Content Objects (SCOs), Activities, Content Organizations and Content Aggregations),
- SCORM Content Packages (with or without sequencing and navigation information), or
- Metadata

Various requirements are defined throughout the book that describes how to create, edit or use content packages, metadata and content model components.

Early portions of this book, Section 1: *The SCORM® Content Aggregation Model (CAM) Overview* through Section 2: *The SCORM® Content Model*, cover general SCORM CAM-related concepts. These sections are recommended reading for those seeking an introduction to the concepts behind the SCORM CAM and who may not wish to delve into its technical details. Others who may find these sections useful include those wishing to learn about updates to the SCORM CAM. Section 2.1.3: *Content Organization*, for instance, discusses how Activities affect the SCORM CAM.

Section 3: *SCORM® Content Packaging* is the first section of this book providing technical details specific to the CAM. It describes Manifests, Content Packages, SCORM Content Aggregation Content Package Application Profile, SCORM Resource Content Package Application Profile and Best Practices and Practical Guidelines. This section covers not only the technical details about the various individual components of SCORM Content Packages, but it also covers how to assemble content packages, showing code illustrations of manifests with explanations.

Section 4: *Metadata* covers all aspects of creating metadata for labeling purposes, to include Learning Object Metadata (LOM) Extensible Markup Language (XML) validation approaches and metadata extensions. The section also describes how to associate metadata to SCORM Content Model Components in a content package.

Section 5: *SCORM® Sequencing and Navigation* covers ways in which the introduction of sequencing and navigation information affects the SCORM CAM. The section also outlines how to build sequencing and navigation information in XML and how to place those rules in a content package manifest. The section describes the requirements for building XML that represents the desired sequencing strategies.

1.1.3. Relationship with Other SCORM Books

While the various SCORM books are intended to stand alone, there are areas of overlap or mutual coverage. For instance, while this book focuses primarily on elements of SCORM content such as SCOs and Assets, those objects are launched by SCORM conformant Learning Management Systems (LMSs), and so the SCORM Run-Time Environment (RTE) book [2], covering content launch is mentioned numerous times.

Similarly, while the Sequencing and Navigation (SN) book covers the details of SCORM sequencing and navigation processes, including detailed coverage of how an LMS evaluates navigation requests and related activities, this book deals with manifests which contain the sequencing information described by the SCORM SN book, and so some of the basics of sequencing and navigation are touched on.

To help clarify areas of overlap, Section 1.1.3.1: *The SCORM Run-Time Environment Book* and Section 1.1.3.2: *The SCORM Sequencing and Navigation Book* provides brief descriptions of the contents of these SCORM books.

1.1.3.1. The SCORM Run-Time Environment Book

The purpose of the SCORM RTE book is to provide a means for interoperability between SCOs and LMSs. SCORM provides a means for learning content to be interoperable across multiple LMSs regardless of the tools used to create the content. For this to be possible, there must be a common way to launch content, a common way for content to communicate with an LMS and predefined data elements that are exchanged between an LMS and content during its execution. The three components of the SCORM RTE are defined in this document as Launch, Application Program Interface (API) and Data Model. The technical details of these elements are described in the SCORM RTE book, but a brief overview of each of these elements of the RTE follows.

Launch includes defining the relationship between LMSs and SCORM content such that all SCORM conformant content is dependant upon a SCORM conformant LMS to be delivered and displayed to the learner. In addition, LMSs have the responsibility to determine which SCORM content is to be delivered next. These new responsibilities, described in the SCORM SN book, are also touched on in the SCORM RTE book.

The SCORM API, as described in the SCORM RTE book, provides a set of predefined methods that are agreed upon by both LMS vendors and content authoring tool vendors to be made available for purposes of communication between an LMS and the SCOs it launches. These functions complete the launch process by providing a means to establish a “handshake” between the SCO and the LMS that launched it, and to break that handshake when the learning session with the SCO is terminated. In addition, they provide the means for SCORM content to "set" and "get" data on the LMS, such as assessment results, and to check for and warn the user about any errors that may occur during these processes.

The SCORM RTE Data Model, as described in the SCORM RTE book, provides the data elements that can be used to "get" and "set" data from and to an LMS. For instance, when passing a test score from a learner, a SCO would use the SCORM RTE Data Model element known as `cmi.score.scaled` to inform the LMS how a user performed in the test. This and all other SCORM Run-Time Environment Data Model elements are described in detail in the SCORM RTE book.

Various concepts described in the SCORM CAM book have impacts on the SCORM RTE. Data defined in a content package manifest impact some initial values for some of the SCORM Run-Time Environment Data Model elements. Data from the manifest is used in the process of delivering and launching content to the learner and impacts the run-time environment. These and other relationships are described throughout the CAM.

1.1.3.2. The SCORM Sequencing and Navigation Book

The SCORM SN book is based on the IMS Simple Sequencing (SS) Specification Version 1.0, which defines a method for representing the intended behavior of an authored learning experience such that any SCORM conformant LMS will sequence discrete learning activities in a consistent way.

The SCORM SN Model defines how IMS SS applies and is extended in a SCORM environment. It defines the required behaviors and functionality that SCORM conformant LMSs must implement to process sequencing information at run-time. More specifically, it describes the branching and flow of learning activities in terms of an Activity Tree, based on the results of a learner's interactions with launched content objects and an authored sequencing strategy. An Activity Tree is a conceptual structure of learning activities managed by the LMS for each learner.

The SCORM SN book describes how learner-initiated and system-initiated navigation events can be triggered and processed, resulting in the identification of learning activities for delivery. Each learning activity identified for delivery will have an associated content object. The SCORM RTE book describes how identified content objects are launched. The sequence of launched content objects, for a given learner and content structure, provides a learning experience (learner interaction with content objects); the SCORM RTE model describes how the LMS manages the resulting learning experience and how that learning experience may affect the Activity Tree.

Various concepts described in the SCORM CAM book have relationships to the SCORM SN book. The SCORM CAM describes how to build sequencing information and represent that information in Extensible Markup Language (XML). The SCORM CAM then describes how to build onto the existing manifest to apply the sequencing information. The SCORM SN book contains more details on the relationship between the XML binding of the sequencing information and the processes and behaviors of that information.

1.2. The SCORM Content Aggregation Model

The SCORM CAM represents a learning taxonomy 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 a learning experience. Learning experiences consist of activities that are supported by electronic or non-electronic learning resources.

One activity in the process of creating and delivering learning experiences involves the creation, discovery and gathering together, or aggregation, of simple assets into more complex learning resources and then organizing the resources into a predefined sequence of delivery. The SCORM CAM supports this process and is made up of the following:

- Content Model: Nomenclature defining the content components of a learning experience.
- Content Packaging: Defines how to represent the intended behavior of a learning experience (Content Structure) and how to aggregate activities of learning resources for movement between different environments (Content Packaging).
- Metadata: A mechanism for describing specific instances of the components of the content model.
- Sequencing and Navigation: A rule-based model for defining a set of rules that describes the intended sequence and ordering of activities. The activities may or may not reference learning resources to be delivered to the learner.

SECTION 2
The SCORM® Content Model

This page intentionally left blank.

2.1. SCORM Content Model Components

The SCORM Content Model describes the SCORM components used to build a learning experience from learning resources. The Content Model also defines how these lower-level sharable, learning resources are aggregated and organized into higher-level units of instruction. The SCORM Content Model is made up of Assets, Sharable Content Objects (SCOs), Activities, Content Organization and Content Aggregations.

2.1.1. Asset

The Asset is the basic building block of a learning resource. Assets are an electronic representation of media, such as text, images, sound, assessment objects or any other piece of data that can be rendered by a Web client and presented to a learner (refer to Figure 2.1.1a). More than one asset can be collected together to build other assets. In some cases, Assets may be launched as part of the learning experience.

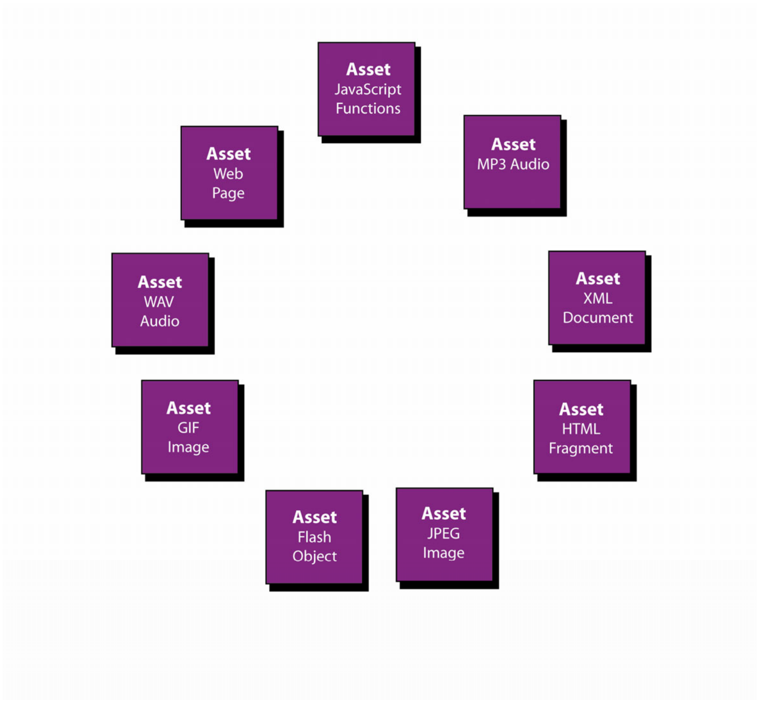


Figure 2.1.1a: Examples of Assets

An Asset can be described with metadata (Refer to the Asset Metadata definition below) to allow for search and discovery within repositories, thereby enabling opportunities for reuse and facilitating maintenance.

2.1.2. Sharable Content Object (SCO)

A SCO is a collection of one or more Assets that represent a single launchable learning resource that uses the SCORM RTE to communicate with an LMS. A SCO represents the lowest level of granularity of a learning resource that is tracked by an LMS using the SCORM Run-Time Environment Data Model. The only difference between a SCO and an Asset is that the SCO communicates with an LMS using the Institute for Electrical and Electronics Engineers (IEEE) ECMAScript Application Programming Interface for Content to Runtime Services Communication standard [1]. Figure 2.1.2a below shows an example of a SCO composed of several Assets.

To improve reusability, a SCO should be independent of its learning context. For example, a SCO could be reused in different learning experiences to fulfill different learning objectives. In addition, an Activity may aggregate more than one SCO resource (and/or Asset resource) 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 contexts is feasible. 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 to be tracked by an LMS at run-time. Reuse requirements for an organization will impact decisions about the size of SCOs. Other factors that may impact the decisions about the size of SCOs include how much information is required to achieve a learning outcome and the point where a branching decision is required for sequencing.

A SCO can be described with metadata (refer to the SCO Metadata definition below) to allow for search and discovery within repositories, thereby enabling opportunities for reuse.

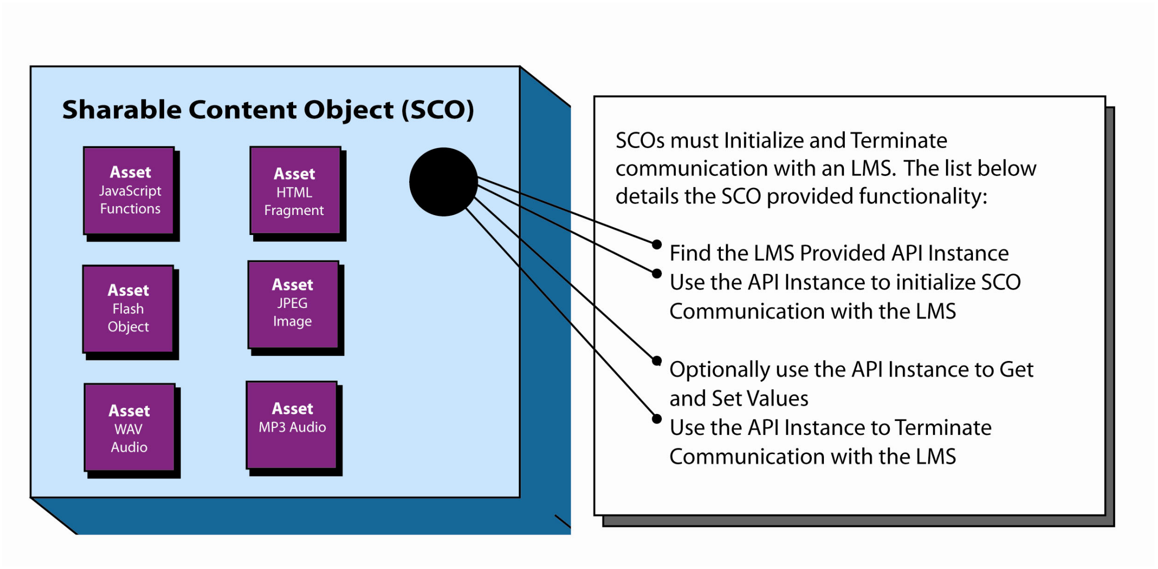

Sharable Content Object (SCO)
Asset JavaScript Functions
Asset HTML Fragment
Asset Flash Object
Asset JPEG Image
Asset WAV Audio
Asset MP3 Audio
SCOs must Initialize and Terminate communication with an LMS. The list below details the SCO provided functionality:
Find the LMS Provided API Instance
Use the API Instance to initialize SCO Communication with the LMS
Optionally use the API Instance to Get and Set Values
Use the API Instance to Terminate Communication with the LMS

Figure 2.1.2a: Conceptual Makeup of a SCO

A SCO is required to adhere to the requirements defined in the SCORM RTE book [2]. This implies that it must have a means to locate an LMS provided API Instance and must invoke the minimum API methods (`Initialize("")` and `Terminate("")`). There is no obligation to invoke any of the other API methods as those are optional and depend upon the nature of the content.

The requirement that a SCO must utilize the SCORM RTE yields the following benefits:

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

2.1.3. Activities

A learning activity may be loosely described as a meaningful unit of instruction; it is conceptually something the learner does while progressing through instruction. A learning activity may provide a learning resource (SCO or Asset) to the learner or it may be composed of several sub-activities.

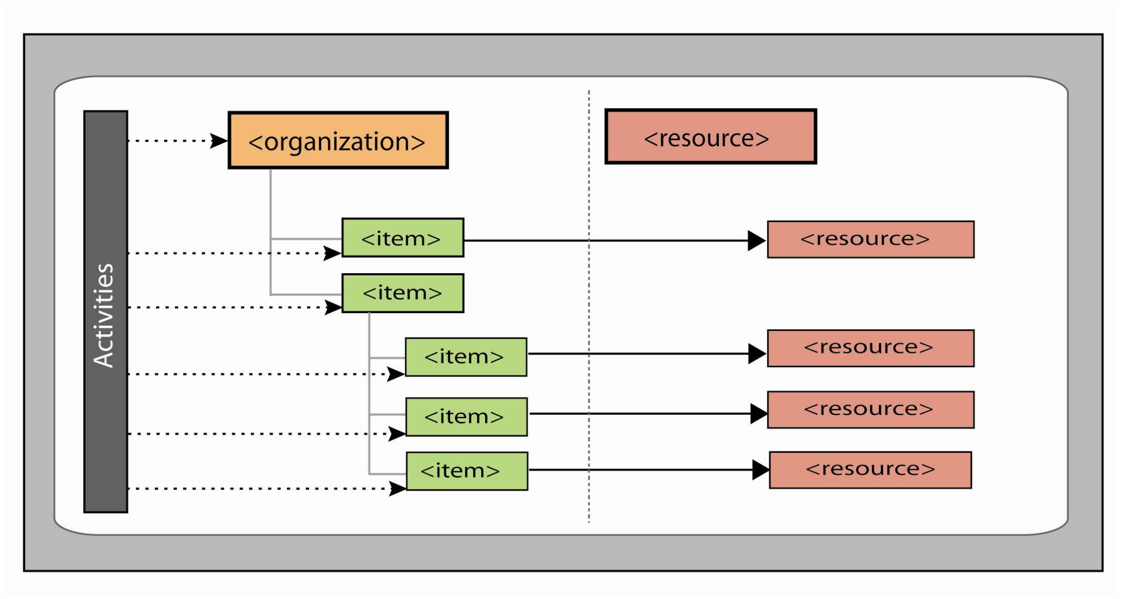


Figure 2.1.3a: Conceptual Representation of Activities

The Activities represented in a Content Organization may consist of other Activities (sub-Activities), which may themselves consist of other activities. There is no set limit to the number of levels of nesting for Activities. While a specific learning taxonomy may be associated with hierarchical levels of Activities, (e.g., course, chapter, module, etc.), this is not a requirement. Activities that do not consist of other Activities (leaf activities) will have an associated learning resource (SCO or Asset) that is used to perform the activity.

Activities that consist of other Activities are also called Clusters. Refer to the SCORM SN book for more details on how behaviors can be defined for Activities and Clusters.

Each Activity in a Content Organization can reference metadata to allow for search and discovery within repositories, thereby enabling opportunities for reuse and facilitating maintenance.

2.1.4. Content Organization

A Content Organization is a representation or map that defines the intended use of the content through structured units of instruction (Activities). The map shows how Activities relate to one another. Figure 2.1.4a below shows an example of a Content Organization.

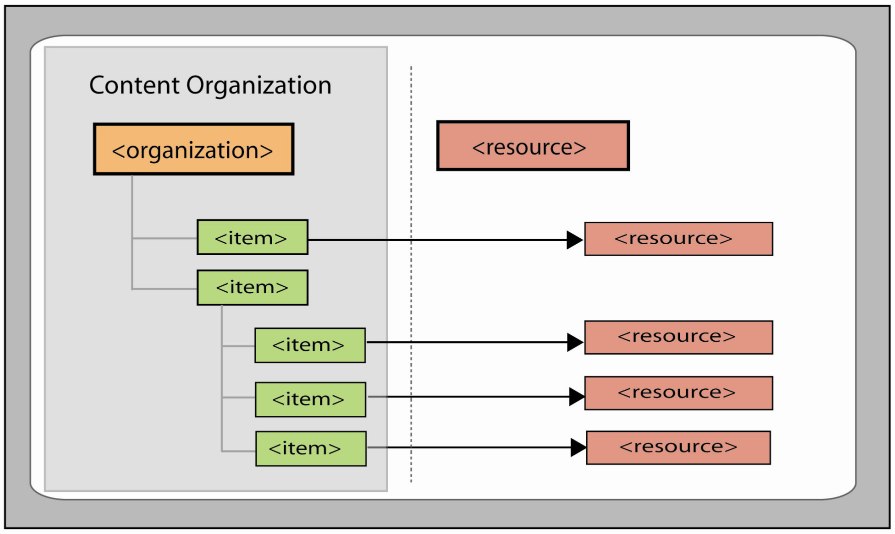


Figure 2.1.4a: Conceptual Illustration of a Content Organization

Content Organization can be described with metadata, thereby enabling opportunities for reuse and facilitating maintenance.

Sequencing only applies to Activities. The intended sequencing of Activities is defined as part of the Content Organization, by structuring Activities in relation to one another and by associating sequencing information with each Activity. The LMS is responsible for interpreting the sequencing information described in the Content Organization and applying sequencing behaviors to control the actual sequence of the learning resources at run-time.

This development strategy represents a 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 sequencing and 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 sequencing methods for content. Before the arrival of SCORM and the shift toward an interoperable development strategy, it was extremely difficult to share content between different authoring environments and equally difficult to reuse content in other contexts that involved different sequencing requirements.

Within SCORM, sequencing information is defined on the Activities represented in the Content Organization and is external to the learning resources associated with those Activities. It is the responsibility of the LMS to launch learning resources associated with the activities in response to applying the defined sequencing behaviors. This is conceptually important because learning resource reuse is limited if a learning resource has embedded sequencing information that is *context-specific* to the course. For example, 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 self-contained.

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 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 organizations. This is an important area that content developers should pay attention to when determining what learning resources should be used and how they are to be aggregated.

2.1.5. Content Aggregation

Content Aggregation can be used as both an action and as a way of describing a conceptual entity. Content Aggregation can be used to describe the action or process of composing a set of functionally related content objects so that the set can be applied in a learning experience. In terms of the SCORM Content Model, a Content Aggregation is also used to describe the entity created as part of this process or action. Sometimes the term is loosely used to describe the content package. The Content Aggregation can then be used to deliver the content and prescribed content structure, transferred between systems or even stored in a repository.

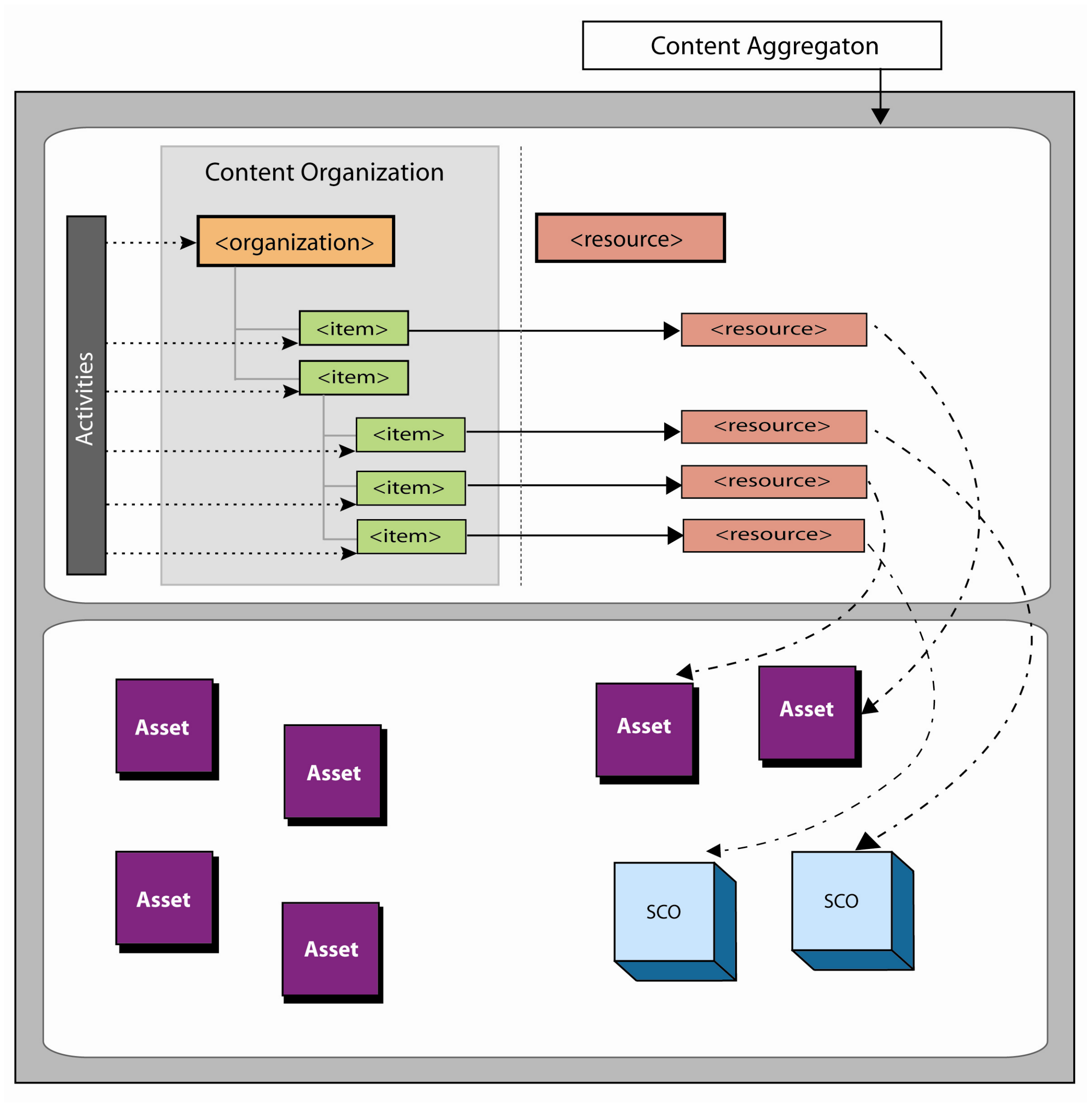


Figure 2.1.5a: Conceptual Illustration of a Content Aggregation

SECTION 3
SCORM® Content Packaging

This page intentionally left blank.

3.1. Content Packaging Overview

Once learning content is designed and built, there is a need to make the content available to learners, authoring tools, repositories or LMSs. The IMS Content Packaging Specification was designed to provide a standard way to structure and exchange learning content.

The purpose of the Content Package is to provide a standardized way to exchange learning content between different systems or tools. The Content Package also provides a place for describing the structure (or organization) and the intended behavior of a collection of learning content.

Content packages are expected to be used to move learning content or collections of learning content between LMSs, development tools and content repositories. The IMS Content Packaging Specification [3] provides a common "input/output" format that any system can support.

SCORM Content Packaging is a set of specific requirements and guidance, or application profiles, of the IMS Content Packaging Specification. SCORM Content Packages adheres *strictly* to the IMS Content Packaging Specification and provides additional explicit requirements and implementation guidance for packaging Assets, SCOs and Content Organization.

This section is organized as follows:

Section 3.2: *Content Package Components* defines the key concepts that deal with a content package. These key concepts are useful for getting a base understanding of a content package before describing the specific requirements.

Section 3.3: *Components of a Manifest* defines the makeup of a Content Package manifest. The manifest acts as the "packaging slip" for the content package. It describes the components of the content package.

Section 3.4: *Building Content Packages* defines the process of building a content package. The section focuses on the creation of the content package and the manifest file. The section describes the XML components of the manifest and the requirements for using those XML components.

Section 3.5: *SCORM Content Package Application Profiles* defines specifically how to create SCORM conformant packages that contain Assets, SCOs and Content Organizations. This section describes the two types of application profiles and the requirements associated with those profiles.

Section 3.6: *Best Practices and Practical Guidelines* defines a collection of best practices and guidelines when building or processing content packages.

3.2. Content Package Components

This section contains an overview of content packages, the nomenclature used to describe content packages and the makeup of content packages. The IMS Content Packaging Specification describes data structures that are used to provide interoperability of Internet-based content with authoring tools, LMSs and run-time environments. The objective of the IMS Content Packaging Specification is to define a standardized set of structures that can be used to exchange content. The scope of the IMS Content Packaging Specification is focused on defining interoperability between systems that wish to import, export, aggregate and disaggregate content packages.

A Content Package contains two major components:

- A special XML document describing the content structure and associated resources of the package called the manifest file (`imsmanifest.xml`). Refer to Section 3.3: *Components of a Manifest* for more details on manifests. A manifest is required to be present at the root of the content package.
- The content (i.e., physical files) making up the content package.

Figure 3.2a is a conceptual diagram that illustrates the components of a Content Package.

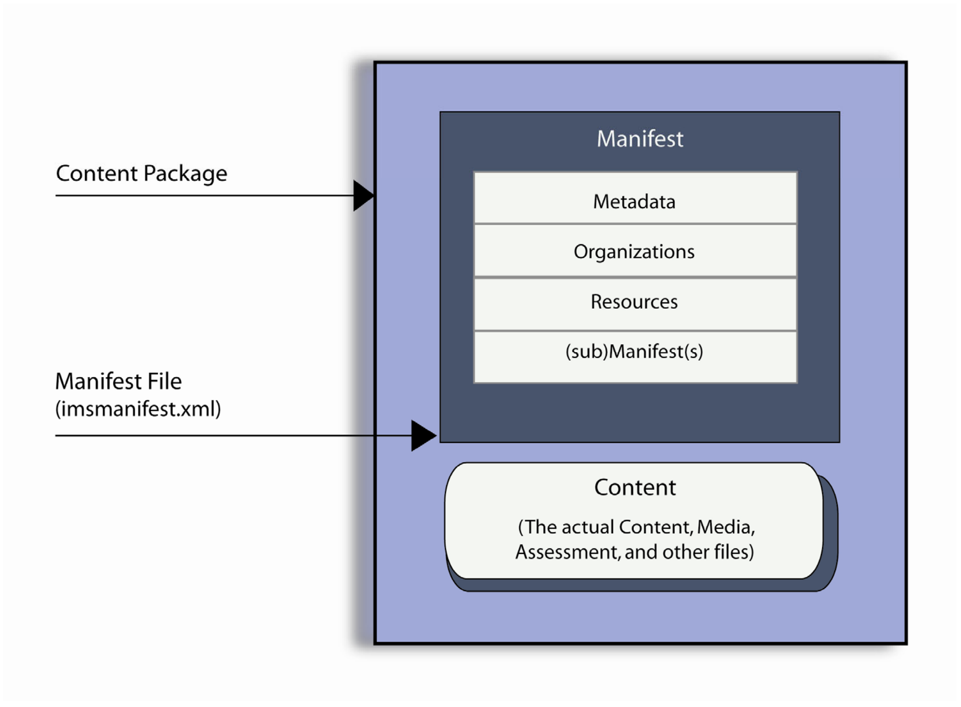


Figure 3.2a: Content Package Conceptual Diagram

3.2.1. Package

A package represents a unit of learning. The unit of learning may be part of a course that has instructional relevance outside of a course organization and can be delivered independently, as a portion of a course, an entire course or as a collection of courses. Once a package arrives at its destination, the package must allow itself to be disaggregated or aggregated. A package must be able to stand alone; that is, it must contain all the information needed to use the packaged contents for learning when it has been unpacked.

3.2.2. Manifest

A manifest is an XML document that contains a structured inventory of the content of a package. If the content package is intended for delivery to an end user, the manifest also contains information about how the content is organized.

The scope of a manifest is elastic. A manifest can describe part of a course that can stand by itself outside of the context of a course (an "instructional object"), an entire course, a collection of courses, or just a collection of content that is to be shipped from one system to another. When packaging a collection of courses, such a content package would typically have to be disaggregated in order to be delivered to learners in a practical LMS run-time system. How to do this disaggregation is out of scope with this version of SCORM. At this point there is no consensus or standard on how to publish a very large or very complex package in a practical LMS, because different LMS systems and repositories use different methods to represent or store the learning content to deliver to learners.

The general rule is that a package always contains a single top-level manifest that may contain one or more (sub)manifests. The top-level manifest always describes the package. Any nested (sub)manifests describe the content at the level at which the (sub)manifest is scoped, such as course, instructional object or other.

ADL Note: The IMS Global Consortium, Inc., is working on a new version of the IMS Content Packaging Specification. One of the major issues that IMS is resolving deals with (sub)manifests, their use, requirements of use and XML syntax requirements. At this time, ADL recommends not to use (sub)manifests until completion of the IMS work. Any questions, concerns or further recommendations on (sub)manifests should be sent to ADL.

The manifest shall adhere to the following requirements (as defined by the IMS Content Packaging Specification):

- The manifest file shall be named `imsmanifest.xml`.
- The `imsmanifest.xml` and any of its supporting control files (e.g., DTD, XSD) shall be at the root of the content package. If extensions are used to describe organizational defined features and those features are represented in XML, then any and all control files also are required to be at the root of the package. This

includes any and all control files needed to validate XML instances including those referenced by the `<adlcp:location>` element.

- All requirements defined in the IMS Content Packaging XML Binding Specification, any restrictions and additional requirements to the IMS binding as defined in Section 3.4: *Building Content Packages*.

3.2.3. Package Interchange File (PIF)

The Package Interchange File (PIF) is a binding of the content package components in the form of a compressed archive file. The PIF contains the `imsmanifest.xml`, all control files and the resources referenced in the content package (those that are local to the PIF, i.e., contained in the content package). SCORM recommends that content packages be created as PIFs. The PIF provides a concise Web delivery format that can be used to transport content packages between systems. If a PIF is used for representing the content package, SCORM requires that the PIF be conformant with RFC 1951 [12]. In addition to this requirement, SCORM mandates that the archive format be PKZip v2.04g (.zip). This `.zip` file is conformant to RFC1951.

3.3. Components of a Manifest

The manifest file represents the information needed to describe the contents of the package. Figure 3.3a describes the makeup of a manifest file.

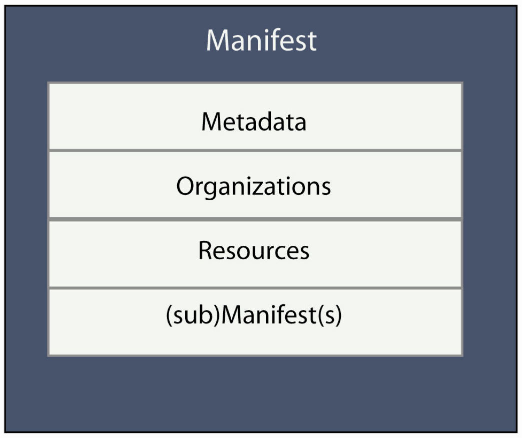


Figure 3.3a: Components of a Manifest

The manifest is composed of four major sections:

1. Metadata: Data describing the content package as a whole.
2. Organizations: Contains the content structure or organization of the learning resources making up a stand-alone unit or units of instruction..
3. Resources: Defines the learning resources bundled in the content package.
4. (sub)Manifest(s): Describes any logically nested units of instruction (which can be treated as stand-alone units).

3.3.1. Metadata

Metadata is defined as "data about data". The metadata depicted in Figure 3.3a is used to describe the content package (i.e., Content Aggregation) as a whole. This metadata enables the search and discoverability of the content package itself. It also enables a mechanism for describing the characteristics of the content package.

ADL Note: Metadata can also be used at various locations within the manifest to describe the different aspects of the content package (refer to Section 4.5.1: *Associating Metadata with SCORM Components* for more information).

3.3.2. Organizations

The Organizations component is used to describe how the content is organized in the content package. It may contain one or more Organization components, each of which describes a particular structure for the content of the package. The current version of the IMS Content Packaging Specification only defines one form of content organization, which is in the shape of a tree or hierarchy. Neither the IMS Content Packaging Specification, nor SCORM, specifies whether this hierarchy should represent a particular learning taxonomy or nomenclature. In other words, SCORM does not specify terms such as “module,” “lesson” and so on to describe the levels of hierarchy in a content organization. Such terms tend to be resolved in a particular way. Therefore, the choice of nomenclature is left to the content developer.

3.3.2.1. Multiple Content Organizations

The IMS Content Packaging Specification separates learning resources from the way those resources are organized, allowing for a single content package to have one or more organizations using the same learning resources within different contexts or uses.

In order to deliver the content package to a learner in an LMS, there must be at least one content organization. In some cases, it is useful to define different ways to use the content in a package. Each of these ways is represented in a different content organization. When a content package has several content organizations, at least one organization must be designated as the default organization, in order to avoid having the system processing the content package decide which content organization to use.

3.3.2.2. Content Organization

The content organization describes how the content of the content package is organized for use. In SCORM Version 1.2, the only defined use for a content organization was as a kind of table of contents for the resources in the content package. In SCORM 2004, the content organization is also used as a way to describe a hierarchy of learning activities that exploit the learning resources. These resources are described in another part of the manifest.

The content organization should not be confused with the physical structure of the content package, or with the structure of the manifest itself. For example, the files in a content package are often organized in a hierarchy of folders, but that structure in itself cannot tell the user of a content package how to use the content of the package.

The purpose of the content organization is to provide the content developer with the means to specify cohesive units of instruction that use collections of learning resources. Such a unit of instruction is a hierarchy of learning activities, for which specific behaviors and rules may be prescribed in such a way that this activity structure and the associated behaviors can be reproduced in any SCORM conformant LMS environment.

For any activity defined in a content organization, the content developer may or may not define specific behaviors and rules. In the absence of specific rules, the content organization is just a map that can be used to navigate at will through the learning resources defined in the content package. By adding specific rules and behaviors, the content organization becomes a guide that prescribes how an LMS must manage the learner's experience and use of the learning resources.

A content organization can be seen as a structured map of learning resources, or a structured activity map to guide the learner through a hierarchy of learning activities that use the learning resources. One content developer may choose to structure the content organization as a table of contents for the learning resources, while another content developer may choose to structure the content organization as an adaptive guided path through a learning experience, invoking learning resources only if and when they are needed. A third content developer may create a content organization where some discovery activities include a free form use of some of the learning resources, while other activities are more formally managed.

An LMS may create its own internal representation for a content package and its contents. The specification of a content organization in SCORM does not imply that LMS systems are required to adopt the content organization model or to store the package components using the same structural organization.

The Organization component defined in the IMS Content Packaging Specification provides the framework for the information that is required to represent the content structure. By design, the Organization component also contains the ability to add, for every item in the organization, additional information such as metadata or behavior rules and prescriptions. SCORM refers to such an Organizations component, and to what it can represent as the "Content Organization."

The Content Organization is not an inventory of the actual learning resources, and it does not describe those resources. The IMS Content Packaging model also provides a clean way to inventory and bundle all of the content (i.e., physical files) required to deliver the learning resources, as well as to identify relationships between files that belong to one or more learning resources, including externally referenced resources that are not contained as physical files within a package.

3.3.2.2.1. Authoring Content and Content Collections

Content organizations provide the means to represent the structure of collections of learning resources. This is a relatively new approach to designing learning content. In the past, Computer Based Training (CBT) authoring tools provided the means to create parts of a course as well as how and when those parts were to be presented to the learner. The learning resources and the content organization were usually inseparable, and developed using the same tools and proprietary data formats. The shift to Internet-based technologies and the notion of building reusable content objects changed the authoring process considerably.

Within SCORM, it is the LMS that is responsible for delivering the content according to the prescriptions embodied in the content organization. That means that the LMS must know how and when a designer intended the learning resources included in a content package to be presented to the learner. The content organization, which is located in the organizations section of the package manifest, allows the designer to provide the LMS with this information. This means that authoring a unit of instruction consists of authoring or collecting learning resources, and also authoring a content organization that uses those learning resources – perhaps using completely different authoring tools.

In SCORM, there are two distinct products of authoring: Authored content organization information, which is used as a prescription by the LMS and processed during run-time, and authored learning resources, which are launched in a browser environment as prescribed by the content organization. Unlike many older CBT models, here the structure is separate from the content. The structure is now fully exposed and can be represented in a standardized form. Content packages can now be delivered in a predictable way, to produce comparable learning experiences in different LMS environments.

3.3.2.3. Representing Content Structure

A SCORM content organization includes components that are intended to define different aspects of a content structure:

- **Content Hierarchy:** This is a tree-shaped representation, much like a table of contents that represents a logical organization for the learning resources or activities that use the learning resources. In many cases, but not all, this hierarchical tree can be traversed in a specific order that represents the default order in which an author intends for the learner to progress through the material.
- **Metadata:** This is optional, *context-specific* descriptive data about the content structure or organization. Such metadata can be used to describe how a particular learning resource is to be used in a particular content organization (e.g., competency or objectives that may be met by a learner when the learning resource is used in a particular activity).
- **Sequencing, Adaptive Sequencing and Navigation:** Optional prescriptions may be embedded in the content organization if the content developer wants to control which learning resources are to be presented to the learner as the learner navigates through the content. By default, if no sequencing and navigation prescription is defined, a learner may choose any content item at will. Adding specific prescriptions can alter this default behavior. For example, adding a flow prescription to the items in the content organization will direct the LMS to guide the navigation in the order defined by the organization tree. More complex adaptive sequencing can be based on the completion status of certain learning resources or on more complex computation of user preferences or assessment results.

Content Organizations are intended to represent a wide variety of approaches to the aggregation of content. A content organization can represent a content aggregation ranging from very, very small learning resources – as simple as a few lines of Hypertext Markup Language (HTML) or a short media clip – to highly interactive learning resources that are tracked by an LMS. The SCORM CAM is neutral about the complexity of content, the number of hierarchical levels of a particular unit of instruction (i.e., taxonomy) and the instructional methodology employed.

Table 3.3.2.3a depicts examples of several possible curricular taxonomy models as used by the Armed Forces of the United States and Canada, which can be represented in a SCORM content organization.

Table 3.3.2.3a: Example of Curricular Taxonomy Models

U.S. Army	U.S. Air Force	U.S. Marine Corps	Canadian Armed Forces
Course	Course	Course	Course
Module	Block	Phase	Performance Objective
Lesson	Module	SubCourse (Annex)	Enabling Objective
Learning Objective	Lesson	Lesson	Teaching Point
Learning Step	Learning Objective	Task	
		Learning Objective	
		Learning Step	

3.3.2.3.1. Content Hierarchy

A collection of learning resources can usually be organized into a hierarchy, which can be represented as a content organization. Another approach to organizing learning resources is to define a hierarchy of activities that rely on the resources to provide a specific learning experience. In this case the top-level of the hierarchy is the main activity, which can consist of sub-activities, which can themselves consist of sub-activities. Depending on the design methodology, this hierarchical grouping might be used to represent concepts like Course, Chapter, Topic or similar terms that represent how the content is organized for delivery to a learner.

The IMS Content Packaging Specification defines a set of terms that are used in representing the content hierarchy. Figure 3.3.2.3.1a depicts how a content organization forms a hierarchical structure that maps to a collection of learning resources (as defined by the IMS Content Packaging Specification). The hierarchical structure is a tree of nested Item elements. The root of the tree is the Organization element. An Item element may reference a Resource element, which describes a specific learning resource to be launched when the Item is used as an activity. The same Resource element may be referenced by more than one Item element.

SCORM and the IMS SS Specification [5] are application profiles of the IMS Content Packaging Specification, and as such they add a couple of restrictions. One is that only leaf Items (Items that do not have children) may reference a learning resource. Another is that an Organization must contain at least one Item.

In addition, SCORM requires that a learning resource be either a SCO or an Asset. These will be described in more detail in the section on Resources.

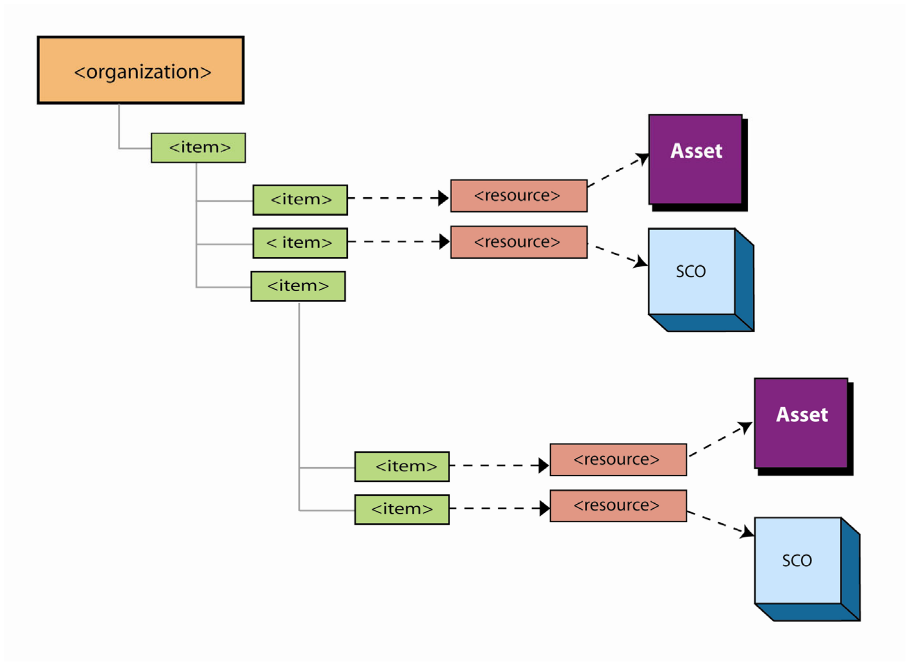


Figure 3.3.2.3.1a: IMS Content Hierarchy Terminology

3.3.2.3.2. Metadata

When a learning resource is created it may be useful to describe the learning resource with metadata. It is important to keep in mind that not all learning resources may need metadata, because the learning resource may not have been built for reuse. Sometimes learning resources are developed and intended to be used in a specific context. Metadata allows the learning resource to be found when it is stored in a repository. It also makes the learning resource more reusable since one can, by inspecting the metadata, decide to reuse it without having to actually open and inspect the resource itself. Such metadata is considered *context-independent,* since it describes the learning resource regardless of where it can be found, independently of any particular use in the context of a specific learning strategy. For example, imagine a simple SCO that teaches how to thread a needle. Metadata describing the SCO might describe the skill to be acquired – inserting a thread through the eye of a needle – and might further describe that a simulation is part of the learning experience. This metadata does not, however, describe how the needle might be used. When the use of a learning resource is defined as part of a learning strategy, additional metadata may be used to describe the activity that uses the resource.

Metadata that is specific to a particular learning strategy is called *context-specific* metadata and is incorporated in the content organization. For example, the metadata may include an explanation of why that particular activity is inserted at that particular place in an instructional sequence.

Context-independent metadata usually refers to immutable metadata that describes digital assets, content objects or collections of objects. *Context-dependent* metadata, on the other hand, usually refers to metadata that makes sense only in the context of a particular content organization.

Developing and applying metadata to learning resources and collections of learning resources is a new concept to many. Best practices for doing this have yet to be developed. In some cases, the principal purpose for metadata is discovery and reuse of content. In other cases, it is strictly informational and provides authors with information about the design and intent of the described learning object or item in a content organization. Some have theorized that metadata could be provided to learners to help them navigate through content. No consensus on common usage of metadata has so far emerged, but provisions have been made in these specifications for a variety of potentially valuable uses of metadata.

If a content package is intended only for delivery to learners, and there is no intent to ever reorganize it or disaggregate it to reuse its components in another organization, adding detailed metadata for each element in the content package may be counter-productive, since such metadata only makes the content package and its manifest more expensive to store, transmit and manage. On the other hand, if there is any chance that the content package may have to be modified, reorganized or disaggregated for reuse at some time in the future, then it makes sense to provide metadata for every element that might be reused or need to be interpreted. In case of doubt, past experience with content seems to indicate that it is probably best to add metadata, even though they may be stripped when a streamline delivery package is required.

3.3.2.3.3. Sequencing, Adaptive Sequencing and Navigation

Sequencing and Navigation refers to the behaviors that an LMS must follow in order to present a specific learning experience as intended by an author or content developer. This learning experience may be free play, in which the learner can choose any item in the content organization in any order, or it may be guided by a flow through the structure of the content organization. The learning experience can be adaptive, with different behaviors that depend on the learner's performance or other variables that can be tracked by the LMS.

SCORM defines a default set of sequencing information that governs sequencing and navigation for a specific content organization. However, the default sequencing information only provides for free play. Many content developers and instructional designers prefer to use specific learning or instructional strategies. In those cases, the content developer can define specific sequencing information that prescribes how an LMS will manage the learning experience.

The sequencing information is associated with the elements in the content organization tree and each of these elements represents an activity that the learner may engage in under control of the sequencing information. Application of sequencing information typically results in either the launching of a learning resource, or a choice the learner must make within the constraints defined by the content developer. Because the sequencing and navigation information is part of the content organization, which is itself part of the package manifest, the intended behaviors can be embedded in a content package in such a form that the package can be used to deliver the same learning experience on any SCORM conformant LMS.

SCORM Sequencing and Navigation provides, among other things, the ability to define highly adaptive activity sequencing. For example, it allows for conditional branching to another activity depending on whether the learner has completed some task, attained an acceptable score or achieved a certain objective. Sequencing and Navigation information embedded in a content organization can prescribe whether and how an LMS may allow the learner to use learning resources in the content package, based on how other learning resources of the same package have been used in past activities.

In the past, CBT authoring tools typically provided custom sequencing and navigation features that were encoded in proprietary data formats. However, new requirements emerged, such as the ability to publish and deploy browser-based content through different LMS systems. Another requirement was to be able to separate structure and instructional strategy logic from the learning resource used in implementing the strategy. This led to the need to standardize some means to define and encode adaptive sequencing and navigation behaviors, so that content organizations can be moved, used and reused across different LMS environments.

The standardization process for sequencing and navigation has proved difficult due to the variety of complex design approaches required in order to effectively train certain tasks or prepare learners for complex roles or responsibilities. Past versions of SCORM provided no specific sequencing capabilities, effectively allowing only pure free play, because it is a difficult and complex subject that required more time to come up with workable solutions. There are many, and often divergent, requirements in the learning design community. No approach has been found to solve all possible use cases. However, the approach used in SCORM, which is based on the IMS SS Specification [5], is flexible enough to allow a wide variety of learning and instructional design approaches.

Section 5: *SCORM® Sequencing and Navigation* describes how the sequencing and navigation rules are embedded into the XML representation of content organizations in a package manifest, in compliance with the IMS SS Specification and SCORM. This specification enables robust sequencing and navigation information to be associated with content packages extending the content organization schema with sequencing prescriptions. These prescriptions are based on a common model for expressing rules, events and conditions as well as run-time behaviors associated with various sequencing and navigation methods.

The IMS SS Specification enables systems to deliver learning resources in a predictable manner, while reacting consistently to learners' interactions with learning resources. The intended approach fosters reusability of learning resources by allowing content developers to define sequencing and navigation behavior or instructional strategies independently of the actual learning resources. The adaptive sequencing information is encoded in the content organization, allowing learning resources to be reused in multiple contexts (i.e., multiple different manifests or organizations, each having their own set of sequencing and navigation information).

3.3.3. Resources

The resources component of a manifest can describe external resources, as well as the content located in the content package. These files may be media files, text files, assessment objects or other pieces of data in electronic form. Conceptual groupings and relationships between files can be represented within the resources component. The combination of resources is generally categorized as "content." The resources are referred to at various points within the organizations component, which provides the structure for the resources.

In Figure 3.3.3a, a single Resource is made up of multiple components. In SCORM, these components are simple assets. If the Resource was built to communicate with an LMS (refer to the SCORM RTE book [2]) then the Resource is a SCO. If the Resource was not built to communicate with an LMS the Resource is considered an Asset. The collection of Resource components makes up the Resources that an Organization can refer to. This collection of Resources and the Organization defines the Content Organization.

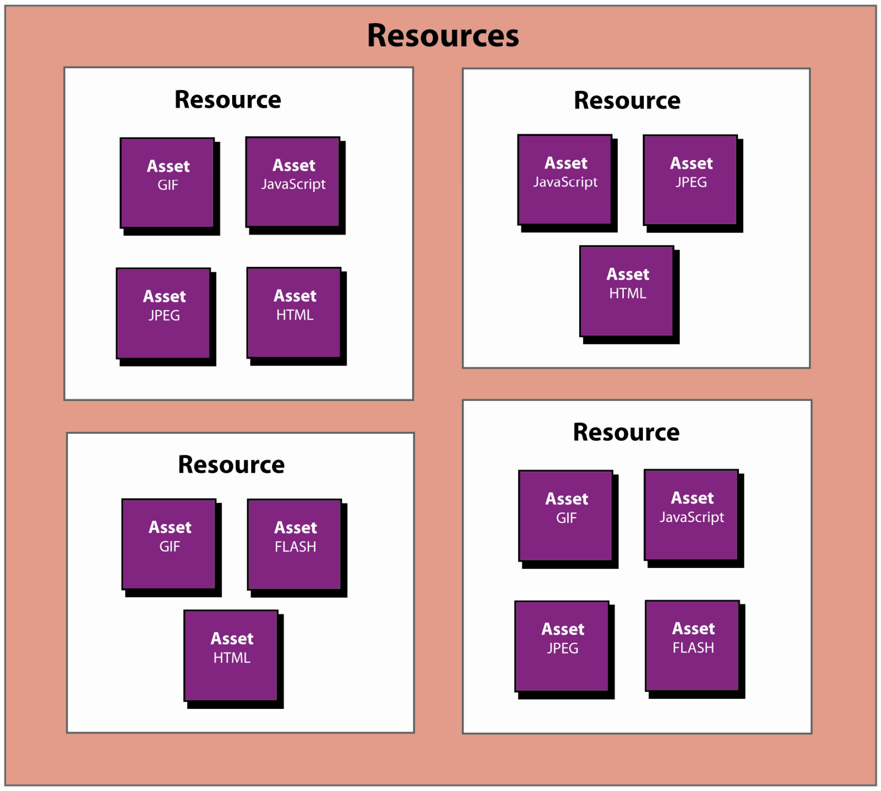


Figure 3.3.3a: Conceptual Illustration of Manifest Resources

The Resource describes the physical makeup (inventory of components) of the resource as a whole. The components of the resource are listed as Files within the Resource.

3.3.4. Content

The content (i.e., physical files) component represents the actual files referenced in the resources component. These files may be local files that are actually contained within the content package, or they can be external files that are referenced by a Universal Resource Indicator (URI). All of the physical files included in the content package should be declared and referenced in the manifest when interchanging content packages. Including files in a content package that are not referenced in the manifest can lead to a wide range of problems when the content package is imported into, run or exported from a system. During one of these processes the physical files may be needed to complete the content package. Systems will use the manifest to determine the makeup of the content package. If physical files are not listed in the manifest then there is the chance of corrupting the content package at a later time.

3.4. Building Content Packages

This section presents the requirements for building SCORM Content Packages. The section describes the XML binding for the IMS Content Packaging Specification as applied to SCORM. There are some specific rules that have guided the creation of this XML binding:

- The XML binding will adhere to the XML 1.0 specification [6] of the W3C.
- The XML binding must maintain the definitional structure of the IMS Content Packaging Information Model.

Some of the requirements are also drawn from other various specifications and standards. The majority of the requirements are inherited by the requirements defined in IMS Content Packaging Specification. Some other specifications and standards are implicitly inherited based on the nature of the XML and other Internet technologies.

This section also defines the requirements for each of the SCORM Content Package Application Profiles:

- Resource Content Package Application Profile: A content package that only contains resources (i.e., no organization). This type of content package can be used for bundling a set of learning resources with no defined organization or content structure. These learning resources may or may not have relationships between each other.
- Content Aggregation Content Package Application Profile: A content package for bundling a set of learning resources and their intended static structure and sequencing requirements (i.e., the manifest contains 1 or more organizations of the learning resources).

Refer to Section 3.5: *The SCORM Content Package Application Profiles* for more information on SCORM Content Package Application Profiles.

3.4.1. Manifest File

The following section defines the requirements for building an `imsmanifest.xml` file. The manifest is a structured inventory of the content of the package. If the package is intended for delivery to an end user, then the manifest also contains information about how the content is organized. The `imsmanifest.xml` is, as the name implies, an XML file. This section defines the requirements for each element defined by the IMS Content Packaging Specification.

Some elements use the term smallest permitted maximum (SPM) in describing the multiplicity and/or data types. The SPM indicates that applications that process content packages shall process at least that number of elements or number of characters, but are free to support and exceed the limit.

The data types and the formats for the elements are defined by the data types prescribed by the XML Schema Part 2: Datatypes W3C Recommendation [13]. The ordering of the XML elements is as defined by the IMS Content Packaging XML Binding for the manifest.

The following table is used to describe the SCORM Content Packaging Application Profile (refer to Section 3.5: *The SCORM Content Package Application Profiles* for more details) requirements:

Table 3.4.1a: SCORM Content Packaging Application Profile Table Format

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	<*requirement*>
Resource	<*requirement*>

The *SCORM Content Packaging Application Profile* column describes the types of application profiles defined by SCORM. The *Manifest Multiplicity Requirements* column defines the multiplicity requirement for the XML elements in the `imsmanifest.xml` file.

Table 3.4.1b: Explanation of Content Packaging Application Profile Multiplicity Requirements

Multiplicity Requirement	Explanation
1 and only 1	The element must exist 1 and only 1 time within the parent element
0 or More	The element can exist 0 or More times within the parent element.
1 or More	The element must exist 1 or More times within the parent element.
0	The element is not permitted.
0 or 1	The element can exist 0 or 1 time within the parent element.

Table 3.4.1b defines the types of multiplicities that are used in this section. Each type is accompanied by a brief explanation.

3.4.1.1. <manifest> Element

The `<manifest>` element represents a reusable unit of instruction that encapsulates metadata, organizations and resource references [3]. The `<manifest>` element is the root element node in the `imsmanifest.xml` file. Subsequent occurrences of the `<manifest>` elements inside the root `<manifest>` are used to compartmentalize files, metadata and organization structure for aggregation, disaggregation and reuse. These child `<manifest>` elements are referred to as (sub)manifests.

ADL Note: The IMS Global Consortium, Inc., is working on a new version of the IMS Content Packaging Specification. One of the major issues that IMS is resolving deals with (sub)manifests, their use, requirements of use and XML syntax requirements. At this time, ADL recommends not to use (sub)manifests until completion of the IMS work. Any questions, concerns or further recommendations on (sub)manifests should be sent to ADL.

All namespace declarations should be declared inside the `<manifest>` element. This includes any namespaces that are considered extensions to IMS and ADL. Although this is not considered a requirement, based on the XML specifications, ADL considers this to be a "best practice" and urges vendors and tools to provide this information.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<manifest>`

SCORM Requirements: The manifest element is the root element node for an IMS Manifest. The root `<manifest>` element shall exist 1 and only 1 time.

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	1 and only 1

Data Type: The `<manifest>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<manifest>` element contains the following elements/attributes.

Attributes:

- `identifier` (mandatory): The attribute identifies the manifest. The `identifier` is unique within the Manifest [3]. The `identifier` attribute is typically provided by an author or authoring tool during the development of the manifest. XML Data Type: `xs:ID`.
- `version` (optional): The `version` attribute identifies the version of the Manifest [3]. It is used to distinguish between manifests with the same identifier. The value has an SPM of 20 characters. XML Data Type: `xs:string`.

- `xml:base` (optional): The `xml:base` attribute provides a relative path offset for the content file(s) contained in the manifest [3]. The usage of this element is defined in the XML Base [7] specification developed by the World Wide Web Consortium (W3C). The value has an SPM of 2000 characters. XML Data Type: `xs:anyURI`.

Elements:

- `<metadata>`
- `<organizations>`
- `<resources>`
- `<manifest>`
- `<imsss:sequencingCollection>`

Example:

```
<manifest identifier="SAMPLE1" version="1.3" xml:base="mycontent"
          xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
          xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
                              imscp_v1p1.xsd
                              http://www.adlnet.org/xsd/adlcp_v1p3
                              adlcp_v1p3.xsd">
   <!-- imsmanifest contents -->
</manifest>
```

Code Illustration 3-1: <manifest> Element

3.4.1.2. <metadata> Element

The `<metadata>` element contains metadata describing the manifest [3]. It contains relevant information that describes the content package (i.e., Content Aggregation) as a whole. The `<metadata>` element is considered the root node for metadata defined in a content package. This means that all metadata for a content package is defined as a child of the `<metadata>` element.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<metadata>`

SCORM Requirements: SCORM places a requirement that all `<manifest>` elements shall contain the following multiplicity requirements for the `<metadata>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	1 and only 1

Data Type: The `<metadata>` element is a parent element. Parent elements have no values associated with them. Parent element acts as "containers" for other elements/attributes. The `<metadata>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<schema>`
- `<schemaversion>`
- `{Metadata}`

Example:

```
<manifest identifier="SAMPLE1" version="1.3" xml:base="mycontent/"
          xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
          xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
                              imscp_v1p1.xsd
                              http://www.adlnet.org/xsd/adlcp_v1p3
                              adlcp_v1p3.xsd">
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
      <adlcp:location>packageMetadata.xml</adlcp:location>
   </metadata>
</manifest>
```

Code Illustration 3-2: <metadata> Element

3.4.1.3. <schema> Element

The `<schema>` element describes the schema that defines and controls the Manifest [3]. Since this element is a child of the metadata describing the package, the element is used to describe the schema that controls the requirements of the manifest.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<schema>`

SCORM Requirements: SCORM places a requirement that the `<schema>` element shall adhere to the following multiplicity requirements:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	1 and only 1

Data Type: The `<schema>` element is represented as a characterstring. XML Data Type: `xs:string.`

SCORM requires that the `<schema>` element contain the following restricted vocabulary token:

- **`ADL SCORM`**: This restricted token indicates that the Content Package is built in accordance with the requirements defined by SCORM.

Example:

```
<manifest>
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
   </metadata>
</manifest>
```

Code Illustration 3-3: <schema> Element

3.4.1.4. <schemaversion> Element

The `<schemaversion>` element describes the version of the above schema (`<schema>`) [3].

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<schemaversion>`

SCORM Requirements: SCORM places a requirement that the `<schemaversion>` element shall adhere to the following multiplicity requirements:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	1 and only 1

Data Type: The `<schemaversion>` element is represented as a characterstring. XML Data Type: `xs:string.`

SCORM requires that the `<schemaversion>` element contain the following restricted vocabulary token:

- **`2004 3rd Edition`**: This restricted token indicates that the Content Package is built in accordance with SCORM 2004 3rd Edition Content Aggregation Model.

Example:

```
<manifest>
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
   </metadata>
</manifest>
```

Code Illustration 3-4: <schemaversion> Element

3.4.1.5. {Metadata}

Metadata can be inserted into a manifest using an appropriate metadata scheme [3]. If using metadata to describe SCORM Content Model Components, ADL highly recommends, at a minimum, the use of the IEEE LOM metadata scheme. Organizations may use various other metadata schemes if necessary (e.g., work being performed by Dublin Core). This metadata describes the package as a whole. There are several mechanisms for inserting metadata in a manifest. Metadata can be inserted into a manifest by extensions to the XML (inline metadata). ADL also provides a namespaced element (refer to Section 3.4.1.5.2: *<location> Element*) to permit a reference to a stand-alone XML file. The `{Metadata}`, found as a child of the `<metadata>`, is optional (can appear 0 or More times using one of the mechanisms described). The example below illustrates the use of inline XML extensions of the LOM elements.

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or More
Resource	0 or More

3.4.1.5.1. Metadata using XML extensions

By definition, XML can be extended by introducing elements and attributes from other defined namespaces. Metadata can be inserted into the Content Package manifest by using this type of XML extension mechanism. There are several ways to add extension elements into an XML document:

1. Defining a namespace in an XML element with a specific prefix. In this method, the specific namespace must be defined using the `xmlns:<prefix>` syntax in the element. Typically, these prefixes are defined in the root node of an XML element. However, this is not a requirement. The prefix and namespace can be defined in any element, as long as the extended elements are not used prior to the declaration of the namespace (refer to *Code Illustration 3-5: Inline Metadata Example – Use of XML Prefix* for an example).

Example:

```
<manifest xmlns:lom = "http://ltsc.ieee.org/xsd/LOM">
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
      <lom:lom>
         <lom:general>
            <lom:title>
               <lom:string language="en-US">Title for the Package</lom:string>
            </lom:title>
         </lom:general>
         <lom:metaMetadata>
            <lom:metadataSchema>LOMv1.0</lom:metadataSchema>
            <lom:metadataSchema>ADLv1.0</lom:metadataSchema>
         </lom:metaMetadata>
      </lom:lom>
   </metadata>
</manifest>
```

Code Illustration 3-5: Inline Metadata Example - Use of XML Prefix

2. Defining a namespace in an XML element without a prefix. In this method, the namespace is defined at the point of use. By using this method, the syntax is stating that the element is from a particular namespace and every child element of this element is also from the namespace (refer to *Code Illustration 3-6: Inline Metadata Example – Use of XML Namespace* for an example).

Example:

```
<manifest>
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
      <lom xmlns="http://ltsc.ieee.org/xsd/LOM">
         <general>
            <title>
               <string language="en-US">Title for the Package</string>
            </title>
         </general>
         <metaMetadata>
            <metadataSchema>LOMv1.0</metadataSchema>
            <metadataSchema>ADLv1.0</metadataSchema>
         </metaMetadata>
      </lom>
   </metadata>
</manifest>
```

Code Illustration 3-6: Inline Metadata Example – Use of XML Namespace

3.4.1.5.2. <location> Element

The `<location>` element provides a means to describe the location where the metadata describing the SCORM Content Model Component may be found. This may be a URI. This is an ADL namespaced element extension to the IMS Content Packaging Specification. The metadata creator has two options for expressing metadata in a Content Package. The creator can either use the `<location>` element to express the location of the metadata record or place the metadata inline within the Manifest file, as described previously. This value is affected by the use of `xml:base` values. Refer to Section 3.4.4.1: *Handling the XML Base Attribute* for more information on `xml:base` usage requirements and guidance.

XML Namespace: `http://www.adlnet.org/xsd/adlcp_v1p3`

XML Namespace Prefix: `adlcp`

XML Binding Representation: `<location>`

SCORM Requirements: SCORM places a requirement that the `<adlcp:location>` element shall adhere to the following multiplicity requirements:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or More
Resource	0 or More

Data Type: The `<adlcp:location>` element is represented as a characterstring. The characterstring has an SPM of 2000 characters. XML Data Type: `xs:string`.

Attributes:

- None

Elements:

- None

Example:

```
<manifest>
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
      <adlcp:location>course/metadata/course.xml</adlcp:location>
   </metadata>
</manifest>
```

Code Illustration 3-7: <adlcp:location> Element

3.4.1.6. <organizations> Element

The `<organizations>` element describes one or more structures or organizations for the content package [3].

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<organizations>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<organizations>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	1 and only 1

SCORM places a requirement that when building a Resource Content Package, this element is required to be represented in the manifest as an empty element (i.e., `<organizations/>`). When building a Content Aggregation Content Package, this element is required to contain at least one `<organization>` sub-element.

Data Type: The `<organizations>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<organizations>` element contains the following elements/attributes:

Attributes:

- `default` (mandatory – for a Content Aggregation Content Package): The `default` attribute identifies the default organization to use. The value of this element must reference an `identifier` attribute of an `<organization>` element that is a direct descendent of the `<organizations>` element. XML Data Type: `xs:IDREF`.

Elements:

- `<organization>`

Example:

```
<organizations default="TOC1">
   <organization identifier="TOC1">
      <title>Introduction to SCORM for LMS Vendors</title>
      <!--organizations structure placed here -->
   </organization>
   <organization identifier="TOC2">
     <title>Introduction to SCORM for Content Vendors</title>
     <!--organizations structure placed here -->
   </organization>
</organizations>
```

Code Illustration 3-8: <organizations> Element

3.4.1.7. <organization> Element

The `<organization>` element describes a particular hierarchical organization [3]. The content organization is defined by the `<organization>` element. The content organization is a conceptual term. The content organization can be a lesson, module, course, chapter, etc. What a content organization defines is dependent on an organization's curricular taxonomy. The `<organization>` element represents an Activity in the terms of IMS SS.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<organization>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<organization>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 or More
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element (its parent) is required to be empty.

Data Type: The `<organization>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<organization>` element contains the following elements/attributes:

Attributes:

- `identifier` (mandatory): An identifier for the organization that is unique within the manifest file [3]. Typically this value is provided by an author or authoring tool. XML Data Type: `xs:ID`.

- `structure` (optional): Describes the shape of the organization [3]. The default value of the structure attribute, if not provided, shall be `hierarchical`. The value has an SPM of 200 characters. XML Data Type: `xs:string`.
- `adlseq:objectivesGlobalToSystem` (optional, default = true): This attribute indicates that any mapped global shared objectives defined in sequencing information (refer to Section 5.1.1: <*sequencing*> *Element*) are either global to the learner and one experience of a content organization (`false`) or global for the lifetime of the learner within the LMS (`true`) across all content organizations. XML Data Type: `xs:boolean`.

Elements:

- `<title>`
- `<item>`
- `<metadata>`
- `<imsss:sequencing>`

Example:

```
<organizations>
   <organization identifier="TOC1">
      <title> Introduction to SCORM for LMS Vendors </title>
      <item identifier="ITEM1" identifierref="RESOURCE1" isvisible="true">
         <title>SCORM Run-Time Environment Requirements</title>
      </item>
      <item identifier="ITEM2" identifierref="RESOURCE2" isvisible="true">
         <title>LMS Conformance Requirements</title>
      </item>
   </organization>
</organizations>
```

Code Illustration 3-9: <organization> Element

3.4.1.8. <title> Element

The `<title>` element describes the title of the organization [3]. This element could be used to help a learner decide which organization to choose. Depending on what the organization is describing, this title could be for a course, module, lesson, etc.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<title>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<title>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty.

Data Type: The `<title>` element is represented as a characterstring element. The characterstring has an SPM of 200 characters. XML Data Type: `xs:string`.

Example:

```
<organization identifier="TOC1">
   <title>Introduction to the SCORM</title>
</organization>
```

Code Illustration 3-10: Organization <title> Element

3.4.1.9. <item> Element

The `<item>` element is a node that describes the hierarchical structure of the organization [3]. The `<item>` element represents an Activity in the content organization. The `<item>` element describes a node within the organization's structure. The `<item>` element can be nested and repeated within other `<item>` elements to any number of levels. This structuring of `<item>` elements shapes the content organization and describes the relationships between parts of the learning content.

The `<item>` element can act as a container of other `<item>` elements or as a leaf node. If an `<item>` is a leaf node, then the `<item>` shall reference a `<resource>` element. If an `<item>` element is a parent element, the `<item>` itself is not permitted to reference a `<resource>` element (only leaf `<item>` elements are permitted to reference resources).

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<item>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<item>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 or More
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty.

Data Type: The `<item>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<item>` element contains the following elements/attributes:

Attributes:

- identifier (mandatory): An identifier attribute is an identifier, for the item, that is unique within the Manifest. XML Data Type: xs:ID.
- identifierref (optional): The identifierref attribute is a reference to an identifier in the resources section or a (sub)manifest [3].

  ADL Note: The IMS Global Consortium, Inc., is working on a new version of the IMS Content Packaging Specification. One of the major issues that IMS is resolving deals with (sub)manifests, their use, requirements of use and XML syntax requirements. At this time, ADL recommends not to use (sub)manifests until completion of the IMS work. Any questions, concerns or further recommendations on (sub)manifests should be sent to ADL.

  If no identifierref is supplied, it is assumed that there is no content associated with this entry in the organization. The value has an SPM of 2000 characters. XML Data Type: xs:string.
- isvisible (optional): The isvisible attribute indicates whether or not this item is displayed when the structure of the package is displayed or rendered. If not present, value is defaulted to be true [3]. The value only affects the item for which it is defined and not the children of the item or a resource associated with an item. XML Data Type: xs:boolean.
- parameters (optional): The parameters attribute contains the static parameters to be passed to the resource at launch time. The parameters attribute should only be used for <item> elements that reference <resource> elements. The value has an SPM of 1000 characters. XML Data Type: xs:string.

  The accepted syntax for the parameters attribute value shall be:

  - #<parameter>
  - <name>=<value>(&<name>=<value>)*(#<parameter>)
  - ?<name>=<value>(&<name>=<value>)*(#<parameter>)

  Where:

  - <parameter>, <name> and <value> is some implementation defined characterstring value
  - = is required to separate the <name> and <value> pair
  - & is required to separate multiple sets of <name> and <value> pairs
  - (&<name>=<value>)* indicates that 0 or more <name> and <value> pairs can be concatenated together

  The characters used in the parameters value may need to be URL encoded. RFC 3986 defines the requirements for encoding URLs.

Elements:

- <title>
- <item>

- `<metadata>`
- `<adlcp:timeLimitAction>`
- `<adlcp:dataFromLMS>`
- `<adlcp:completionThreshold>`
- `<imsss:sequencing>`
- `<adlnav:presentation>`

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true"
         parameters="?width=500&#038;length=300">
       <title>Content 1</title>
   </item>
</organization>
```

Code Illustration 3-11: <item> Element

3.4.1.10. <title> Element

The `<title>` element describes the title of the item [3].

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<title>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<title>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty. Consequently, no `<item>` or `<title>` will be provided.

Data Type: The `<title>` element is represented as a characterstring element. The characterstring has an SPM of 200 characters. XML Data Type: `xs:string`.

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true">
      <title>Content 1</title>
   </item>
</organization>
```

Code Illustration 3-12: Item <title> Element

3.4.1.11. <item> Element

The `<item>` element can be nested an arbitrarily number of levels. This is typically based on the content structure of the aggregation. The `<item>` element can appear 0 or More times as a child of an `<item>` element (refer to Section 3.4.1.9: *<item> Element* for more details on the `<item>` element.)

3.4.1.12. <metadata> Element

The `<metadata>` element contains metadata describing the item [3]. It contains relevant information that describes the `<item>` element (i.e., Activity) as a whole. The `<metadata>` element is considered the root node for metadata describing the activity. This means that all metadata for the activity is defined as a child of the `<metadata>` element.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<metadata>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<metadata>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or 1
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty.

Data Type: The `<metadata>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<metadata>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- {Metadata} – Refer to Section 3.4.1.5: *{Metadata}* for information on the inclusion of metadata.

ADL Note: This is different from the `<metadata>` element defined in Section 3.4.1.2: *<metadata> Element*. The IMS Content Packaging Specification only permits the `<schema>` and `<schemaversion>` elements on the `<metadata>` element defined as a child of the `<manifest>` element.

Example:

```
<organization>
   <item>
     <title>The organization title</title>
     <metadata>
        <adlcp:location>lesson1/lesson1MD.xml</adlcp:location>
     </metadata>
   </item>
</organization>
```

Code Illustration 3-13: Item <metadata> Element

3.4.1.13. <timeLimitAction> Element

The `<timeLimitAction>` element defines the action that should be taken when the maximum time allowed in the current attempt of the activity is exceeded. All time tracking and time limit actions are controlled by the SCO.

This element is an ADL defined extension to the IMS Content Packaging Specification. The element shall only appear, if needed, as a child of a leaf `<item>` element that references a SCO. Only those `<item>` elements that reference a SCO resource can contain the `<timeLimitAction>` element.

The LMS shall use the value of the `<timeLimitAction>` element, if provided, to initialize the `cmi.time_limit_action` data model element (refer to the SCORM RTE book [2]). If the content developer defines a time limit action, then the SCO is responsible for all behaviors based on the time out (if the time out occurs).

XML Namespace: `http://www.adlnet.org/xsd/adlcp_v1p3`

XML Namespace Prefix: `adlcp`

XML Binding Representation: `<timeLimitAction>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<timeLimitAction>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or 1
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty.

Data Type: The `<timeLimitAction>` element is represented as a characterstring. The characterstring is required to be one of following set of restricted characterstring tokens:

- **`exit,message`**: The learner should be forced to exit the SCO. The SCO should provide a message to the learner indicating that the maximum time allowed for the learner attempt was exceeded.

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

If this feature is used within the SCO, the SCO shall keep track of the time affecting this timeout period and provide the informative message indicating the timeout (if appropriate).

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true">
      <title>Content 1</title>
      <adlcp:timeLimitAction>exit,no message</adlcp:timeLimitAction>
   </item>
</organization>
```

Code Illustration 3-14: <timeLimitAction> Element

3.4.1.14. <dataFromLMS> Element

The `<dataFromLMS>` element provides initialization data expected by the resource (i.e., SCO) represented by the `<item>` after launch. This data is opaque to the LMS and only has functional meaning to the SCO. This element shall not be used for parameters that the SCO may need during the launch (query string parameters). If this type of functionality is required, then the developer should use the `parameters` attribute of the item referencing the SCO resource.

This element is an ADL defined extension to the IMS Content Packaging Specification. The element shall only appear, if needed, as a child of a leaf `<item>` element. Only those `<item>` elements that reference a SCO resource can contain the `<dataFromLMS>` element).

The LMS shall use the value of the `<dataFromLMS>` element, if provided, to initialize the `cmi.launch_data` data model element (refer to SCORM Run-Time Environment book [2]).

XML Namespace: `http://www.adlnet.org/xsd/adlcp_v1p3`

XML Namespace Prefix: `adlcp`

XML Binding Representation: `<dataFromLMS>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<dataFromLMS>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or 1
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty.

Data Type: The `<dataFromLMS>` element is represented as a characterstring element. The characterstring has an SPM of 4000 characters.

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true">
      <title>Content 1</title>
      <adlcp:dataFromLMS>Some SCO Information</adlcp:dataFromLMS>
   </item>
</organization>
```

Code Illustration 3-15: <dataFromLMS> Element

3.4.1.15. <completionThreshold> Element

The `<completionThreshold>` element defines a threshold value that can be used by the SCO resource referenced by the `<item>` for which the `<completionThreshold>` is defined. This element is an ADL defined extension to the IMS Content Packaging Specification. The element shall only appear, if needed, as a child of a leaf `<item>` element. Only those `<item>` elements that reference a SCO resource can contain the `<completionThreshold>` element).

The LMS shall use the value of the `<completionThreshold>` element, if provided, to initialize the `cmi.completion_threshold` data model element (refer to the SCORM RTE book [2]). This value can be used by the SCO to determine completion.

XML Namespace: `http://www.adlnet.org/xsd/adlcp_v1p3`

XML Namespace Prefix: `adlcp`

XML Binding Representation: `<completionThreshold>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<completionThreshold>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or 1
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty.

Data Type: The `<completionThreshold>` element is represented as a decimal value between the range of 0.0 and 1.0 element.

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true">
      <title>Content 1</title>
      <adlcp:completionThreshold>0.75</adlcp:completionThreshold>
   </item>
</organization>
```

Code Illustration 3-16: <completionThreshold> Element

3.4.1.16. <sequencing> Element

Refer to Section 5.1.1: *<sequencing> Element*.

3.4.1.17. <presentation> Element

Refer to Section 5.2.1: *<presentation> Element*.

3.4.1.18. <metadata> Element

The `<metadata>` element is metadata describing the organization [3]. It contains relevant information that describes the `<organization>` element (i.e., Content Organization) as a whole. The `<metadata>` element is considered the root node for metadata describing the content organization. This means that all metadata for the content organization is defined as a child of the `<metadata>` element.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<metadata>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for this `<metadata>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or 1
Resource	0

For Resource Content Packages, this element shall not appear. The `<organizations>` element is required to be empty.

Data Type: The `<metadata>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<metadata>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- {Metadata} – Refer to Section 3.4.1.5: *{Metadata}* for information on the inclusion of metadata.

<u>**ADL Note:**</u> This is different from the `<metadata>` element defined in Section 3.4.1.2: *<metadata> Element*. The IMS Content Packaging Specification only permits the `<schema>` and `<schemaversion>` elements on the `<metadata>` element defined as a child of the `<manifest>` element.

Example:

```
<organization>
   <title>Introduction to SCORM</title>
   <item identifier="ITEM1" identifierref="RESOURCE1" isvisible="true">
      <title>SCORM Run-Time Environment Requirements</title>
   </item>
   <metadata>
      <adlcp:location>activities/activity1MD.xml</adlcp:location>
   </metadata>
</organization>
```

Code Illustration 3-17: Organization <metadata> Element

3.4.1.19. <sequencing> Element

Refer to Section 5.1.1: *<sequencing> Element*.

3.4.1.20. <resources> Element

The `<resources>` element is a collection of references to resources. There is no assumption of order or hierarchy of the individual `<resource>` elements that the `<resources>` element contains [3].

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<resources>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<resources>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	1 and only 1
Resource	1 and only 1

Data Type: The `<resources>` element is a parent element. Parent elements have no values associated with them. Parent elements act as “containers” for other elements/attributes. The `<resources>` element contains the following elements/attributes:

Attributes:

- `xml:base` (optional)**:** The `xml:base` attribute provides a relative path offset for the content file(s) [3]. The usage of this element is defined in the XML Base [7] Working Draft from the W3C. The value has an SPM of 2000 characters. XML Data Type: `xs:anyURI.`

Elements:

- `<resource>`

Example:

```
<manifest>
   <metadata/>
   <organizations/>
   <resources>
      <resource identifier="RESOURCE1" adlcp:scormType="sco" type="webcontent"
                href="lesson1.htm">
         <file href="lesson1.htm"/>
      </resource>
      <resource identifier="RESOURCE2" adlcp:scormType="sco" type="webcontent"
                href="intro1.htm">
         <file href="intro1.htm"/>
      </resource>
      <resource identifier="RESOURCE3" adlcp:scormType="asset"
                type="webcontent" href="content1.htm">
         <file href="content1.htm"/>
      </resource>
      <resource identifier="RESOURCE4" adlcp:scormType="sco" type="webcontent"
                href="summary1.htm">
         <file href="summary1.htm"/>
       </resource>
   </resources>
</manifest>
```

Code Illustration 3-18: <resources> Element

3.4.1.21. <resource> Element

The `<resource>` element is a reference to a resource [3]. There are two primary types of resources defined within SCORM:

- SCOs
- Assets

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<resource>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<resource>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or More
Resource	0 or More

A leaf `<item>` element is required to reference a resource (SCO resource or Asset resource). If an `<item>` references a resource, this resource is subject to being identified for delivery and launch to the learner. If an `<item>` references a `<resource>`, then the `<resource>` element shall meet the following requirements:

- The `type` attribute should be set to `webcontent`.
- The `adlcp:scormType` shall be set to `sco` or `asset`.
- The `href` attribute shall be required.

Data Type: The `<resource>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<resource>` element contains the following elements/attributes:

Attributes:

- `identifier` (mandatory): The `identifier` attribute represents an identifier, of the resource, that is unique within the scope of its containing manifest file [3]. This identifier is typically provided by an author or authoring tool. XML Data Type: `xs:ID`.
- `type` (mandatory): The `type` attribute indicates the type of resource [3]. The value has an SPM of 1000 characters. XML Data Type: `xs:string`.
- `href` (optional): The `href` attribute is a reference a Uniform Resource Locator (URL) [3]. The `href` attribute represents the "entry point" or "launching point" of this resource. External fully qualified URLs are also permitted. The URL may also contain additional parameters (refer to Section 3.4.3.3: *Handling Additional Parameters as Part of URLs* for information regarding the handling of these parameters). This value is affected by the use of `xml:base` values. Refer to Section 3.4.4.1: *Handling the XML Base Attribute* for more information on `xml:base` usage requirements and guidance. The value has an SPM of 2000 characters. The SPM represents the length of the `href` with the values of any `xml:base` applied to it. XML Data Type: `xs:string`.
- `xml:base` (optional**):** The `xml:base` attribute provides a relative path offset for the files contained in the manifest. The usage of this element is defined in the XML Base Working Draft from the W3C. The value has an SPM of 2000 characters. XML Data Type: `xs:anyURI`.
- `adlcp:scormType` (mandatory): The `adlcp:scormType` attribute defines the type of SCORM resource. This is an ADL extension to the IMS Content Packaging Information Model. XML Data Type: `xs:string`. The character string is restricted and shall be one of the following characterstring tokens (**`sco`** or **`asset`**).

Where `sco` indicates that the resource is a SCO resource and `asset` indicates that the resource is an Asset resource.

Elements:

- `<metadata>`
- `<file>`
- `<dependency>`

Example:

```
<resources>
   <resource identifier="R_A2" type="webcontent" adlcp:scormType="sco"
             href="sco1.html">
      <file href="sco1.html"/>
   </resource>
   <resource identifier="R_A3" type="webcontent" adlcp:scormType="sco"
             href="sco3.html">
      <file href="sco3.html"/>
   </resource>
   <resource identifier="R_A5" type="webcontent" adlcp:scormType="asset"
             href="pics\distress_sigs_add.jpg">
      <file href="pics\distress_sigs_add.jpg"/>
   </resource>
</resources>
```

Code Illustration 3-19: <resource> Element

3.4.1.22. <metadata> Element

The `<metadata>` element is metadata describing the resource [3]. It contains relevant information that describes the `<resource>` element as a whole. The `<resource>` element may represent a SCO or an Asset. This depends on the value of the `adlcp:scormType` attribute. The `<metadata>` element is considered the root node for metadata describing the resource. This means that all metadata for the resource is defined as a child of the `<metadata>` element.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<metadata>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<metadata>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or 1
Resource	0 or 1

Data Type: The `<metadata>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<metadata>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- {Metadata}: Refer to Section 3.4.1.5: *{Metadata}* for information on the inclusion of metadata.

ADL Note: This is different from the `<metadata>` element defined in Section 3.4.1.2: *<metadata> Element*. The IMS Content Packaging Specification only permits the `<schema>` and `<schemaversion>` elements on the `<metadata>` element defined as a child of the `<manifest>` element.

Example:

```
<resources>
   <resource identifier="R_A2" type="webcontent" adlcp:scormType="sco"
             href="sco1.html">
      <file href="sco1.html"/>
      <metadata>
         <adlcp:location>resources/resource1MD.xml</adlcp:location>
      </metadata>
   </resource>
</resources>
```

Code Illustration 3-20: Resource <metadata> Element

3.4.1.23. <file> Element

The `<file>` element is a listing of files that a resource described by a `<resource>` element requires for delivery. This element is repeated as necessary for each file for a given resource. The element acts as an inventory system detailing the set of files used to build the resource. The `<file>` element represents files that are local to the content package. For all files that are required for delivery of the content package in a SCORM run-time environment and are local to the content package (physically located in the content package), a `<file>` element shall be used to represent the file relative to the resource in which it is used. If the resource identified by a `<resource>` element is local to the package, then the resource itself shall be identified as a `<file>` element. The launch location of the `<resource>` (`<resource>`'s href value) shall be used as the `href` of the file. The value of the `<file>` element's `href` attribute should not contain any parameters that may have been defined on the `<resource>` element's `href` attribute.

All of the physical files that are included in the content package should be referenced by a file element. Leaving these references to the physical files out of the manifest may cause a wide range of problems.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<file>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<file>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or More
Resource	0 or More

Data Type: The `<file>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<file>` element contains the following elements/attributes:

Attributes:

- `href` (mandatory): The `href` attribute identifies the location of the file [3]. This value is affected by the use of `xml:base` values. Some systems may treat this value as case sensitive, developers should be aware of this and ensure that values used match the resources being referenced. Refer to Section 3.4.4.1: *Handling the XML Base Attribute* for more information on `xml:base` usage requirements and guidance. The value has an SPM of 2000 characters. The SPM represents the length of the `href` with the values of any `xml:base` applied to it. XML Data Type: `xs:string`.

Elements:

- `<metadata>`

Example:

```
<resource identifier="R_A2" type="webcontent" adlcp:scormType="sco"
          href="sco1.html">
   <file href="assets/image1.gif"/>
   <file href="sco1.html"/>
   <file href="assets/common/APIWrapper.js"/>
</resource>
```

Code Illustration 3-21: <file> Element

3.4.1.24. <metadata> Element

The `<metadata>` element is metadata describing the file [3]. It contains relevant information that describes the `<file>` element (i.e., Asset) as a whole. The `<metadata>` element is considered the root node for metadata describing the Asset. This means that all metadata for the Asset is defined as a child of the `<metadata>` element.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<metadata>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<metadata>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or 1
Resource	0 or 1

Data Type: The `<metadata>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<metadata>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- {Metadata} – Refer to Section 3.4.1.5: *{Metadata}* for information on the inclusion of metadata.

<u>**ADL Note:**</u> This is different from the `<metadata>` element defined in Section 3.4.1.2: *<metadata> Element*. The IMS Content Packaging Specification only permits the `<schema>` and `<schemaversion>` elements on the `<metadata>` element defined as a child of the `<manifest>` element.

Example:

```
<resource identifier="R_A2" type="webcontent" adlcp:scormType="sco"
          href="sco1.html">
   <file href="assets/image1.gif">
      <metadata>
         <adlcp:location>assets/asset1.xml</adlcp:location>
      </metadata>
   </file>
   <file href="sco1.html"/>
   <file href="assets/common/APIWrapper.js"/>
</resource>
```

Code Illustration 3-22: File <metadata> Element

3.4.1.25. <dependency> Element

The `<dependency>` element identifies a resource whose files this resource (the resource in which the dependency is declared in) depends on [3]. The resource that the `<dependency>` references can act as a container for multiple files that the resource containing the `<dependency>` is reliant on.

XML Namespace: `http://www.imsglobal.org/xsd/imscp_v1p1`

XML Namespace Prefix: `imscp`

XML Binding Representation: `<dependency>`

SCORM Requirements: SCORM places a requirement that all manifests shall adhere to the following multiplicity requirements for the `<dependency>` element:

SCORM Content Packaging Application Profile	Manifest Multiplicity Requirements
Content Aggregation	0 or More
Resource	0 or More

Data Type: This element is represented as an empty element. The `<dependency>` element only contains attributes.

Attributes:

- `identifierref` (mandatory): The `identifierref` attribute references an `identifier` attribute of a `<resource>` (within the scope of the `<manifest>` element for which it is defined) and is used to resolve the ultimate location of the dependent resource. The `identifierref` is not permitted to reference a resource defined in a (sub)manifest. The value has an SPM of 2000 characters. XML Data Type: `xs:string`.

Elements:

- None

Example:

```
<resources>
   <resource identifier="R_A2" type="webcontent" adlcp:scormType="sco"
             href="sco1.html">
      <file href="sco1.html"/>
      <dependency identifierref="R_A5"/>
   </resource>
   <resource identifier="R_A5" type="webcontent" adlcp:scormType="asset"
             href="pics\distress_sigs_add.jpg">
      <file href="pics\distress_sigs_add.jpg"/>
   </resource>
</resources>
```

Code Illustration 3-23: <dependency> Element

3.4.1.26. <manifest> Element

Refer to Section 3.4.1.1: *<manifest> Element.*

ADL Note: The IMS Global Consortium Inc., is working on a new version of the IMS Content Packaging Specification. One of the major revisions that IMS is resolving deals with (sub)manifests, their use, requirements of use and XML syntax requirements. At this time, ADL recommends not to use (sub)manifests until completion of the IMS work. Any questions, concerns or further recommendations on (sub)manifests should be sent to ADL.

3.4.1.27. <sequencingCollection> Element

Refer to Section 5.1.12: *<sequencingCollection> Element*.

3.4.2. Content Package Manifest Extensions

The IMS Content Packaging Specification allows for communities to place their own namespaced elements throughout the manifest. SCORM carries this practice with a set of extensions. These sets of extensions are required to meet various requirements and to help in the application profiling of the various specifications and standards described in SCORM. The extensions are defined in three separate XML Schema Definition (XSD) files. These files are:

- `adlcp_v1p3.xsd`: Describes SCORM Content Packaging extensions. These extension elements and attributes are defined within the `http://www.adlnet.org/xsd/adlcp_v1p3` namespace. ADL reserves the namespace and the `adlcp:` prefix to indicate elements of that namespace. These extension elements are described in detail in Section 3.4.1: *Manifest File*.
- `adlseq_v1p3.xsd`: Describes SCORM Sequencing extensions. These extension elements and attributes are defined within the `http://www.adlnet.org/xsd/adlseq_v1p3` namespace. ADL reserves the namespace and the `adlseq:` prefix to indicate elements of that namespace. These extension elements are described in detail in Section 3.4.1: *Manifest File.*
- `adlnav_v1p3.xsd`: Describes SCORM Navigation extensions. These extension elements and attributes are defined within the `http://www.adlnet.org/xsd/adlnav_v1p3` namespace. ADL reserves the namespace and the `adlnav:` prefix to indicate elements of that namespace. These extension elements are described in detail in Section 3.4.1: *Manifest File.*

Extending the manifest with an organization-defined extension could cause a low degree of semantic interoperability. Organizations that do not recognize the extension may not know how to deal with the extension properly and may ignore the extensions. To keep a high degree of semantic interoperability and, if the need arises, to extend the manifest, ADL recommends that vertical communities work to a consensus on building a set of interoperable extensions.

3.4.3. Content Package Manifest Href Handling

An "href" is used to describe the location of a `<file>` or `<resource>` identified in the content package manifest. This location can either be an internal URL or an external URL. According to the IMS Content Packaging Version 1.1.3 Final Specification, the value of an href is to be constructed according to the rules expressed in RFC 3986 [8].

3.4.3.1. Handling the XML Base Attribute

The XML Base [7] is a construct used to explicitly specify the base URL of a document in resolving relative URLs in links to files in a Content Package. The URLs can be prefixed by an XML Base attribute. This allows an author or authoring tool to specify and/or offset the base directory so that it is not necessary to repeat the base directory in every use of that URL.

The default base directory is the location of the package. This concept is known as "relative to the package." The only way to explicitly override this default base directory is to reference a file with an absolute path that is external to the package. If the file is not external to the package, any XML Base value will simply offset the default base directory. The XML Base attribute can be either:

- A relative URL (describing the offset from the root of the package, e.g., Course/Lesson/), or
- External URL (external from the package, e.g., http://www.adlnet.gov/content/).

Trailing slashes are required to be at the end of any XML Base value. When referencing local files in the content package, the URL, including XML Base, shall not begin with a leading forward slash ("/"). As defined in RFC 3986, a path with a leading forward slash indicates the absolute path of that file. Using a leading slash denotes the root of the local host. With this in mind, the use of a leading forward slash is not permitted to minimize misinterpretation and increase portability.

The IMS Content Packaging XML Binding Specification allows for the use of the XML Base attribute in the `<manifest>`, `<resources>` and `<resource>` elements.

If the XML Base attribute is present in the `<manifest>` element, all URLs found within the child elements of the `<manifest>` element shall use the XML Base value to construct the actual `href` value. This includes the href values for the `<resource>` and `<file>` elements and the value held in the `<adlcp:location>` element.

```
<manifest xml:base="Course/">
   <organizations>
      <organization>
         <item identifier="ID1" identifierref="R_ID1"></item>
      </organization>
   </organizations>
   <resources>
      <resource identifier="R_ID1"
                href="Lesson01/Topics/index.htm"></resource>
      </resources>
</manifest>
```

Code Illustration 3-24: XML Base Defined on <manifest> Element

Because of the use of the XML Base attribute in the `<manifest>` element and an `href` exists within the child hierarchy of the `<manifest>` element, the actual `href` for the resource, shown in *Code Illustration 3-24: XML Base Defined on <manifest> Element*, is: **`Course/Lesson01/Topics/index.htm`**.

If the XML Base attribute is present in the `<resources>` element, all URLs found within the child elements of the `<resources>` element shall use the XML Base value to construct the actual `href` value.

```
<manifest>
   <organizations>
      <organization>
         <item identifier="ID1" identifierref="R_ID1"></item>
      </organization>
   </organizations>
   <resources xml:base="Course/Lesson01/">
      <resource identifier="R_ID1" href="Topics/index.htm"></resource>
   </resources>
</manifest>
```

Code Illustration 3-25: XML Base Defined on <resources> Element

Because of the use of the XML Base attribute in the `<resources>` element and an `href` exists within the child hierarchy of the `<resources>` element, the actual `href` for the resource, shown in *Code Illustration 3-25: XML Base Defined on <resources> Element*, is: **`Course/Lesson01/Topics/index.htm`**

If the XML Base attribute is present in the `<resource>` element, all URLs found within the child elements of the `<resource>` element shall use the XML Base value to construct the actual `href` value.

```
<manifest>
   <organizations>
      <organization>
         <item identifier="ID1" identifierref="R_ID1">
         </item>
      </organization>
   </organizations>
   <resources>
      <resource identifier="R_ID1" xml:base="Course/Lesson01/Topics/"
                href="index.htm">
      </resource>
   </resources>
</manifest>
```

Code Illustration 3-26: XML Base Defined on <resource> Element

Because of the use of the XML Base attribute in the `<resource>` element and an `href` exists within the child hierarchy of the `<resource>` element, the actual `href` for the resource, shown in *Code Illustration 3-26: XML Base Defined on <resource> Element*, is: **`Course/Lesson01/Topics/index.htm.`**

If a combination of XML Base attributes is used throughout the Manifest, the value of the XML Base attribute shall be appended in the order of the hierarchy of these elements to form the actual URL. The `<manifest>` element's XML Base value comes first, followed by the `<resources>` elements XML Base value, followed by the `<resource>` elements XML Base value, followed by the `href` attribute's value.

```
<manifest xml:base="Course/">
   <organizations>
      <organization>
         <item identifier="ID1" identifierref="R_ID1"></item>
      </organization>
   </organizations>
   <resources xml:base="Lesson01/">
      <resource identifier="R_ID1"
                href="index.htm" xml:base="Topics/">
      </resource>
   </resources>
</manifest>
```

Code Illustration 3-27: XML Base Defined on All Permitted Elements

Because of the use of the XML Base attributes in the `<manifest>`, `<resources>` and `<resource>` elements and an `href` exists as an attribute of the `<resource>` element, the actual `href` for the resource, shown in *Code Illustration 3-27: XML Base Defined on All Permitted Elements*, is: **`Course/Lesson01/Topics/index.htm`**.

When using the `xml:base` attribute, care should be taken in regards to the SPM defined for the `xml:base` and `href` attribute values. The `xml:base` attribute has an SPM of 2000 characters. This SPM applies not only to the actual value held by the `xml:base` attribute, but also impacts the overall SPM for the `href` attribute. The SPM of 2000 characters defined for the `href` attribute represents the SPM for the length of the `href` after all of the `xml:base` attributes have been applied to the `href` to form the absolute URL. So developers should keep this SPM in mind when using the `xml:base` attribute in conjunction with the href attribute. Violating the SPM may cause interoperability issues between different systems.

3.4.3.2. URL Encoding and Decoding

In some situations the URLs used in defining the location of the files or resource may need to be encoded. RFC 3986 defines the rules and requirements for encoding URLs. Sections 2.2 through 2.4 of RFC 3986 describe how and when to encode or decode URLs. Some characters have a "structural" purpose as delimiters in a URL and may not be escaped when they serve that purpose. Those characters are:

- "/" in the schema part of the URL, or as a separator in the path part of a URL
- ":" in the schema part of the URL
- "#" as the lead character for an anchor value
- "&" as the separator between parameters
- "?" as the separator between the path part of a URL and the parameters
- "=" as the separator between a parameter value and the parameter name
- "%" as an escape indicator.

If one of those characters exists in the URL, but not for that purpose, it must be encoded (or escaped). With this in mind, it would be incorrect to take a complete URL such as "`Course/Lesson/Module/Resources/bar.html`" and just escape it using an ECMAScript escape function, since that would change the path separator slashes that are required for interpretation of the URL into escaped characters. On the other hand, if the value of a parameter that is included as part of the URL contains a slash character, that character must be escaped. So escaping must be done *before* assembling the parts of the URL, by escaping the segments that need to be escaped and then assembling those parts with delimiters that should not be escaped.

Additionally, double encoding shall not be used when the URL is included in a manifest. Thus the value of an href attribute for a resource shall be a string that contains a valid URL in the exact format required to launch the resource in a Web browser.

Furthermore, if parameters are specified for an `<item>`, they shall be properly escaped for use in a URL. For example, suppose that the following parameters are needed for a particular SCO:

- `"?ratio=3/4&scale=100&label=Gilbert & Sullivan"`

The example above is not a valid parameter value because it contains illegal characters. The "/" in "3/4" needs to be escaped because it is not part of the URL and it is not used as a separator in the path. Also, the "&" in "Gilbert & Sullivan" needs to be escaped because it is not used as a separator between parameters. The correctly escaped equivalent to these parameters is:

- `"ratio=3%2F4&scale=100&label=Gilbert %26 Sullivan"`

However, the following is not correctly escaped because characters are double-escaped:

- `"ratio=3%252F4&scale=100&label=Gilbert %2526 Sullivan"`

Since manifests are implemented in XML, the XML rules for escaping must also be followed. For example, the "&" may appear in its literal form only when used within a comment, processing instruction or a CDATA section according to the XML 1.0 standard. If they are needed elsewhere, as in the above example, it must be escaped using either numeric character references or strings ("%26" or "&amp;"). The following would be the correct value to use in the parameters attribute of the item element:

- `"ratio=3%2F4%26scale=100%26label=Gilbert %26 Sullivan"`

It is permitted to escape other characters such as the "=" or the " " such as:

- `"ratio%3D3%2F4%26scale%3D100%26label%3DGilbert%20%26%20Sullivan"`

3.4.3.3. Handling Additional Parameters as Part of URLs

There may be situations where content objects require information at launch time in order for the content object to operate properly. This information is sometimes referred to as launch parameters (or query strings). There are currently two mechanisms for representing query strings in a Manifest.

- **Option 1: As part of the `<resource>` or `<file>` href attribute.** The content developer can place the query string as part of the href. An example of this is:

```
<resource href="foo.html?Topic=1">
   <!-- resource information -->
</resource>
```

Code Illustration 3-28: Parameters defined on Href Attribute

- **Option 2: Using the parameters attribute of the `<item>`.** The content developer also has the option of placing the query string or launch parameters in the parameters attribute of the `<item>` that references a `<resource>`. The `href` attribute in the `<resource>` element is the URL used to launch the resource, which may or may not resolve to a file in the package. The href in the `<file>` element specifies a file name and, if, required, the installation path relative to the package installation "root" directory. These are not redundant, because they are not necessarily the same. For example, the `href` in the `<resource>` element might very well be something like "`scos/foo.html#xyz`", while the corresponding `<file>` `href` is "`scos/foo.html`".

Also, note that the manifest is also an inventory of every file included in the package, including the launch file for a resource. In other words, the `<file>` element is an inventory entry. The `<resource>` element specifies how to use a particular set of files (or how to access an external resource), and the `<item>` element in an `<organization>` specifies how to use a resource in one or more places in a content organization.

If an `<item>` references a `<resource>`, it is required that the `<resource>` contains an href entry for launching the resource. The parameters attribute is defined as the static parameters to be passed to the resource at launch time. This allows for the ability to reference the same `<resource>` from different items for different purposes.

For example:

```
<manifest>
   <!-- For brevity, elements not in question or part of the -->
   <!-- example were kept simple and incomplete -->
   <organizations>
      <organization>
         <item identifier="I01" identifierref="R_I01"
               parameters="?Topic=1">
         </item>
      </organization>
   </organizations>
   <resources>
      <resource identifier="R_I01" href="foo.htm">
      </resource>
   </resources>
</manifest>
```

Code Illustration 3-29: Parameters defined with parameters Attribute

In option 1, each Resource would have to be repeated in the Manifest, with `parameters` defined in the `href` attribute.

Due to the number of ways to syntactically represent the launch parameters within the Manifest, the IMS Content Packaging Specification details an algorithm for constructing the href attribute of the resource element and the parameters consistently.

```
While first char of parameters is in “?&”
   Clear first char of parameters
If first char of parameters is “#”
   If URI contains “#”
      Discard parameters
   Else
      Append parameters to the URI
   Done processing URI
If the URI contains a “?”
   Append “&” to the URI
Else
   Append “?” to the URI
Append parameters to URI
Done processing the URI
```

Code Illustration 3-30: Pseudo Code for Defining Href

3.5. SCORM Content Package Application Profiles

SCORM Content Package Application Profiles describe how the IMS Content Packaging Specification will be applied within the overall context of SCORM. The application profiles provide practical guidance for implementers and define additional requirements imposed by SCORM to integrate other standards and specifications and ensure interoperability. The IMS Content Packaging Specification will be used as the basis for a SCORM Content Package. However, SCORM will impose additional requirements, above those defined by the IMS Content Packaging Specification, to ensure sufficient information is included in each package. This will enable SCORM conformant systems to import and export packages that can be used by other SCORM conformant systems.

SCORM introduces the Content Aggregation Model (Section 2.1: *The SCORM Content Model Components*) that defines a generalized framework for object based learning content. The components are Assets, SCOs and Content Organizations. There are currently two SCORM Content Package Application Profiles, which describe how to package Content Aggregation Model components, identified:

- Resource Content Packages
- Content Aggregation Content Packages.

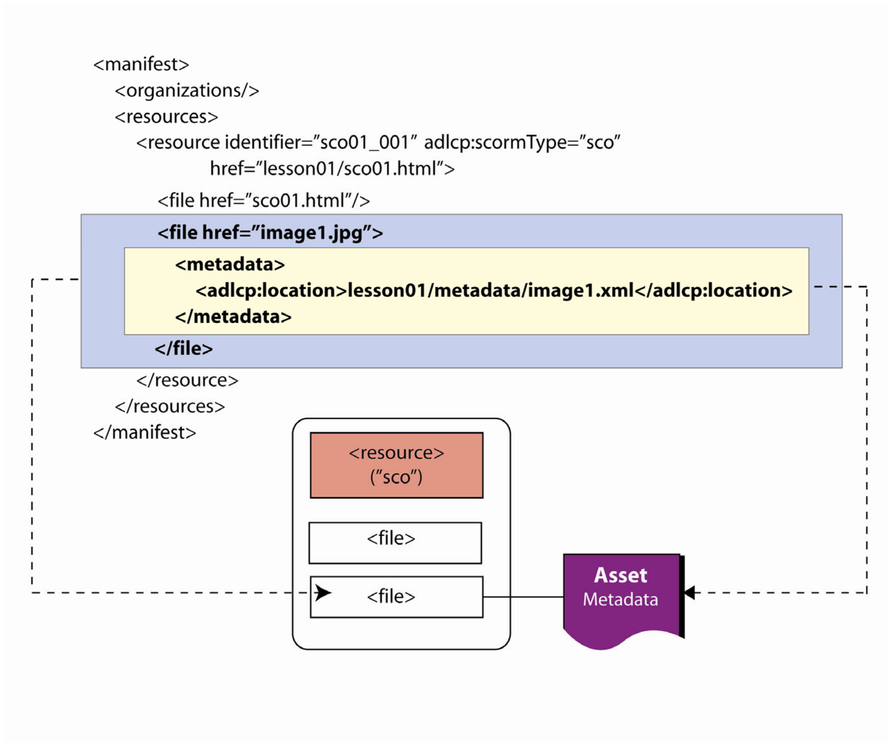
The following sections describe the application profiles, the constraints imposed by SCORM and a set of recommended best practices.

3.5.1. Resource Content Package

The SCORM Resource Content Package Application Profile defines a mechanism for packaging Assets and SCOs without having to provide any organization, learning context or curricular taxonomy. Packaging learning resources provides a common medium for exchange. The Resource Content Package Application Profile should be used for moving SCOs and Assets from system to system. Since there is no organization defined in a Resource Content Package, no logical content structure is defined. Since no structure is defined, this type of package cannot be delivered by an LMS to the learner. The SCORM Resource Content Package is merely a collection of learning resources that can be transferred between learning systems.

In many cases an Asset resource or a SCO resource will be comprised of a single file. However, there are cases where Assets and SCOs could be comprised of multiple files. The SCORM Resource Content Package Application Profile allows for packaging Assets and SCOs comprised of single or multiple files. Also, Assets and SCOs may be included locally in the package or may be referenced externally. Locally packaged files will be included as physical files within the overall package. When externally referenced, the Assets and SCOs will not be included as physical files within the package, but will instead be referenced by an URL.

The following figures depict several resource content packages. The examples show a sample `imsmanifest.xml` instance and how Assets and SCOs could be represented. Figure 3.5.1a shows an example of an Asset being represented as a `<file>` element in an `imsmanifest.xml` instance.



```
<manifest>
   <organizations/>
   <resources>
      <resource identifier="sco01_001" adlcp:scormType="sco"
               href="lesson01/sco01.html">
         <file href="sco01.html"/>
         <file href="image1.jpg">
            <metadata>
               <adlcp:location>lesson01/metadata/image1.xml</adlcp:location>
            </metadata>
         </file>
      </resource>
   </resources>
</manifest>
```

Figure3.5.1a: Example of an Asset Represented as a <file> Element

Figure 3.5.1b shows an example of an Asset being represented as a `<resource>` element (i.e., Asset resource) in an `imsmanifest.xml` instance.

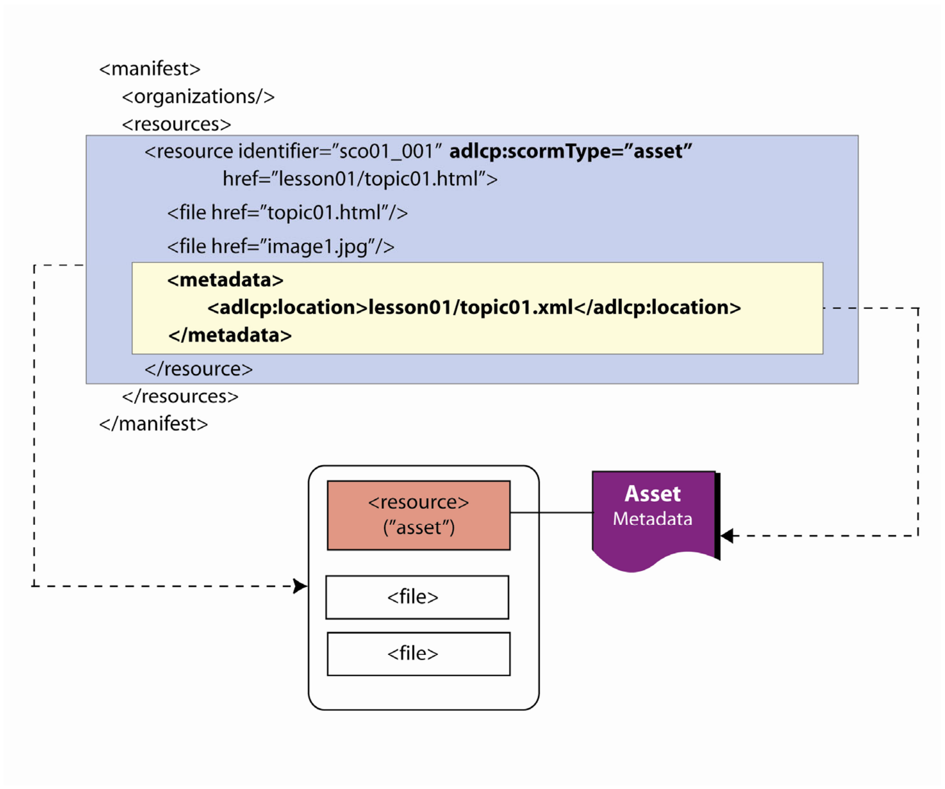
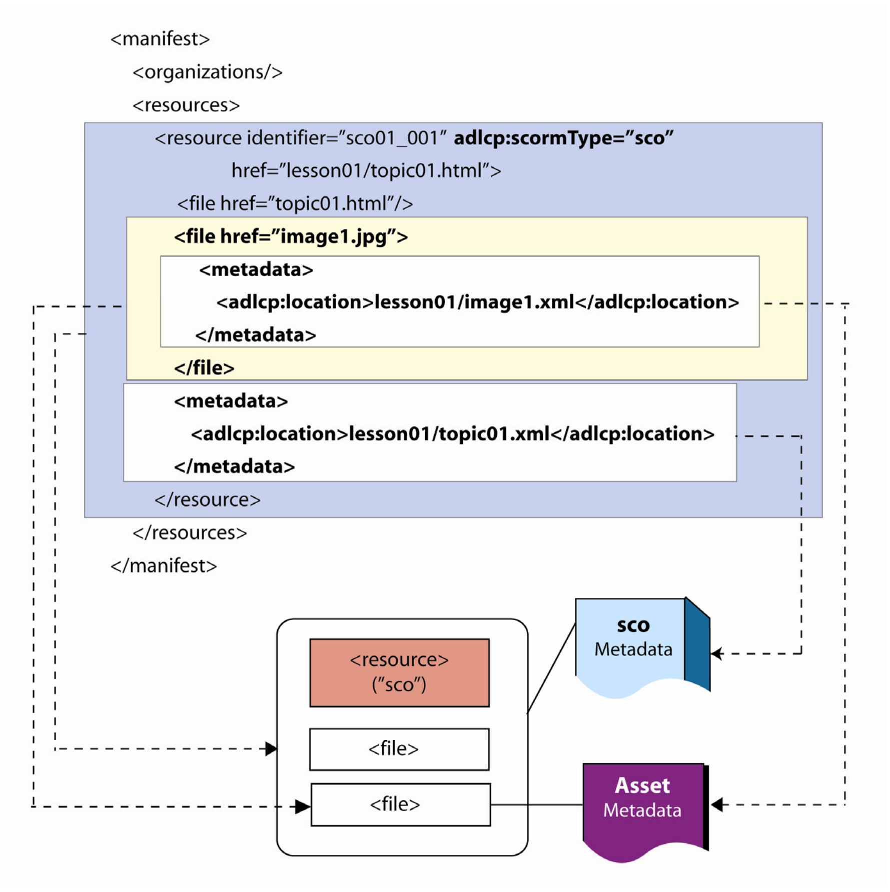

```
<manifest>
  <organizations/>
  <resources>
    <resource identifier="sco01_001" adlcp:scormType="asset"
           href="lesson01/topic01.html">
      <file href="topic01.html"/>
      <file href="image1.jpg"/>
      <metadata>
         <adlcp:location>lesson01/topic01.xml</adlcp:location>
      </metadata>
    </resource>
  </resources>
</manifest>
```

Figure 3.5.1b: Example of an Asset Represented as a <resource> Element

Figure 3.5.1c shows an example of a SCO being represented as a `<resource>` element in an `imsmanifest.xml` instance.



```
<manifest>
   <organizations/>
   <resources>
      <resource identifier="sco01_001" adlcp:scormType="sco"
               href="lesson01/topic01.html">
        <file href="topic01.html"/>
        <file href="image1.jpg">
          <metadata>
            <adlcp:location>lesson01/image1.xml</adlcp:location>
          </metadata>
        </file>
        <metadata>
          <adlcp:location>lesson01/topic01.xml</adlcp:location>
        </metadata>
      </resource>
   </resources>
</manifest>
```

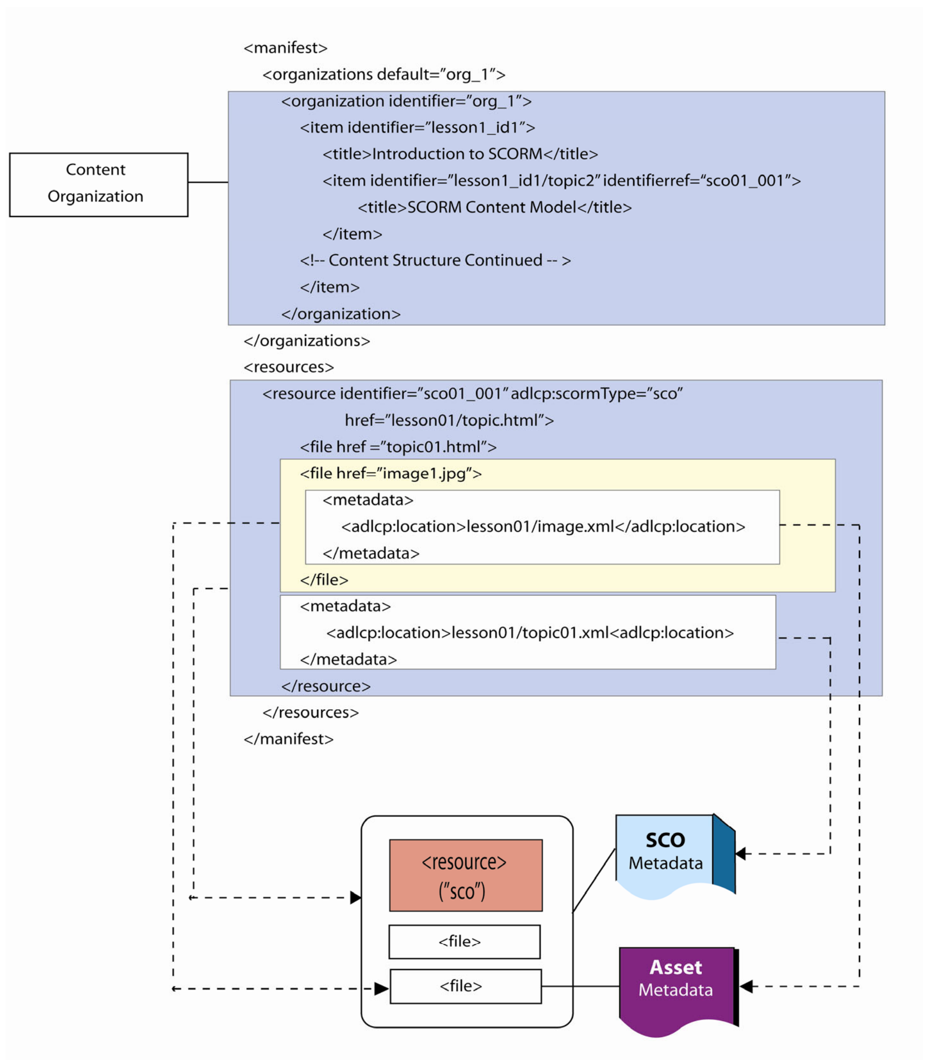
Figure 3.5.1c: Example of a SCO Represented as a <resource> Element

3.5.2. Content Aggregation Content Package

SCORM does not impose any requirements on the structure for content organizations. Individual content developers are free to aggregate content into any structure that provides value to them. The IMS Content Packaging Specification [3] provides a framework that includes most of the information that is needed by ADL, as well as logical places in which ADL extensions can be added to capture the rest of the information. Additionally, the IMS packaging model also provides a clean way to inventory and bundle all of the physical files required to deliver the learning resource, as well as to identify relationships between files that belong to one or more learning resources, including externally referenced resources that are not contained as physical files within a content package. The Content Aggregation Content Package Application Profile should be used to bundle learning resources and the content structure. This is the application profile that should be used to bundle complete courses, modules, lessons, etc. The Content Aggregation Content Package's main purpose is to be used to deliver content to an end user (i.e., typically through an LMS).

The IMS Content Packaging Specification also enables a separation of learning resources from the way those resources can be organized, allowing for one or more uses of the same learning resources within different contexts. SCORM defines a mechanism for packaging the files and providing the structure.

Figure 3.5.2a shows an example of a Content Organization being represented in an `imsmanifest.xml` instance.



```
<manifest>
   <organizations default="org_1">
      <organization identifier="org_1">
         <item identifier="lesson1_id1">
            <title>Introduction to SCORM</title>
            <item identifier="lesson1_id1/topic2" identifierref="sco01_001">
                  <title>SCORM Content Model</title>
            </item>
         <!-- Content Structure Continued -- >
         </item>
      </organization>
</organizations>
<resources>
   <resource identifier="sco01_001" adlcp:scormType="sco"
              href="lesson01/topic.html">
         <file href ="topic01.html">
         <file href="image1.jpg">
            <metadata>
               <adlcp:location>lesson01/image.xml</adlcp:location>
            </metadata>
         </file>
         <metadata>
            <adlcp:location>lesson01/topic01.xml<adlcp:location>
         </metadata>
      </resource>
   </resources>
</manifest>
```

Figure 3.5.2a: Example of a Content Aggregation Content Package and its Components

3.5.3. SCORM Content Package Application Profile Requirements

Table 3.5.3a defines the requirements for each of the aforementioned Content Package Application Profiles. Each of the profiles is listed with the corresponding requirements for each of the content packaging manifest elements/attributes.

- "M" indicates that the element/attribute is Mandatory.
- "O" indicates that the element/attribute is Optional.
- "NP" indicates that the element/attribute is Not Permitted.

Elements are indicated as they would in the XML Binding (i.e., using XML notation `<element_name>`). Attributes are indicated without any notation (e.g., 1.1 `identifier` is an attribute of the `<manifest>` element). The numbering system is based off of the IMS Content Packaging Specification.

Table 3.5.3a: SCORM Content Package Application Profile Manifest Element Requirements

No.	Elements	Resource Content Package	Content Aggregation Content Package
1	<manifest>	M	M
1.1	identifier	M	M
1.2	version	O	O
1.3	xml:base	O	O
1.4	<metadata>	M	M
1.4.1	<schema>	M	M
1.4.2	<schemaversion>	M	M
1.4.3	{Metadata}	O	O
1.5	<organizations>	M	M
1.5.1	default	NP	M
1.5.2	<organization>	NP	M
1.5.2.1	identifier	NP	M
1.5.2.2	structure	NP	O
1.5.2.3	adlseq:objectivesGlobalToSystem	NP	O
1.5.2.4	<title>	NP	M
1.5.2.5	<item>	NP	M
1.5.2.5.1	identifier	NP	M
1.5.2.5.2	identifierref	NP	O
1.5.2.5.3	<title>	NP	M
1.5.2.5.4	isvisible	NP	O
1.5.2.5.5	parameters	NP	O
1.5.2.5.6	<item>	NP	O
1.5.2.5.7	<metadata>	NP	O
1.5.2.5.7.1	{Metadata}	NP	O
1.5.2.5.8	<adlcp:timeLimitAction>	NP	O
1.5.2.5.9	<adlcp:dataFromLMS>	NP	O
1.5.2.5.10	<adlcp:completionThreshold>	NP	O
1.5.2.5.11	<imsss:sequencing>	NP	O
1.5.2.5.12	<adlnav:presentation>	NP	O

No.	Elements	Resource Content Package	Content Aggregation Content Package
1.5.2.6	<metadata>	NP	O
1.5.2.6.1	{Metadata}	NP	O
1.5.2.7	<imsss:sequencing>	NP	O
1.6	<resources>	M	M
1.6.1	xml:base	O	O
1.6.2	<resource>	O	O
1.6.2.1	identifier	M	M
1.6.2.2	type	M	M
1.6.2.3	href	O	O
1.6.2.4	adlcp:scormType	M	M
1.6.2.5	xml:base	O	O
1.6.2.6	<metadata>	O	O
1.6.2.6.1	{Metadata}	O	O
1.6.2.7	<file>	O	O
1.6.2.7.1	href	M	M
1.6.2.7.2	<metadata>	O	O
1.6.2.7.2.1	{Metadata}	O	O
1.6.2.8	<dependency>	O	O
1.6.2.8.1	identifierref	M	M
1.7	<manifest>	O	O
1.8	<imsss:sequencingCollection>	NP	O

3.6. Best Practices and Practical Guidelines

The following section describes a set of recommended best practices and practical guidelines for the development of content packages. These best practices are not considered conformance requirements.

3.6.1. Multiple Organizations for a Single Course

The content package allows for representations of multiple organizations for its resources. The same resources may be used in different content organizations adapted for different audiences. For example, one may find value in forcing novice users to progress through content in a linear manner without the ability to skip any instructional units, while advanced users may want to use the content as a refresher by selecting only the instructional units that they would like to experience. Multiple organizations can be used to structure a set of resources in different ways for different reasons. The utilization of multiple organization elements is ideal for the use case.

3.6.2. Using the <dependency> Element

Several learning resources, defined in a content package, may contain the same set of files. The files are represented as `<file>` elements in the manifest. The `<dependency>` element can be used to group these sets of files. The use of the `<dependency>` element in this scenario will alleviate the duplication of the `<file>` element for each set of files in each resource. In this scenario, a `<resource>` element can be used to gather the set of files. Once the `<resource>` is set up, all of the other resources that depend on the set of files can reference the resource by using the `<dependency>` element.

```
<manifest>
   <organizations>
      <organization>
         <item identifier="ID1" identifierref="R_ID1"></item>
         <item identifier="ID2" identifierref="R_ID2"></item>
      </organization>
   </organizations>
   <resources>
      <resource identifier="R_ID1" adlcp:scormType="sco" href="index_1.htm">
         <file href="index_1.htm"/>
         <file href="image1.jpg"/>
         <file href="image2.jpg"/>
         <file href="image3.jpg"/>
         <file href="apiWrapper.js"/>
      </resource>
      <resource identifier="R_ID2" adlcp:scormType="sco" href="index_1.htm">
         <file href="index.htm"/>
         <file href="image1.jpg"/>
         <file href="image2.jpg"/>
         <file href="image3.jpg"/>
         <file href="image4.gif"/>
         <file href="apiWrapper.js"/>
      </resource>
   </resources>
</manifest>
```

Code Illustration 3-31: Sharing of Files

In *Code Illustration 3-31: Sharing of Files*, the two defined resources both share a common set of files:

- `image1.jpg`
- `image2.jpg`
- `image3.jpg`
- `apiWrapper.js`

These sets of files are repeated, as `<file>` elements, for each resource. The method described above can be used to eliminate the repeating of these `<file>` elements.

```
<manifest>
   <organizations>
      <organization>
         <item identifier="ID1" identifierref="R_ID1"></item>
         <item identifier="ID2" identifierref="R_ID2"></item>
      </organization>
   </organizations>
   <resources>
      <resource identifier="R_ID1" adlcp:scormType="sco" href="index_1.htm">
         <file href="index_1.htm"/>
         <dependency identifierref="DEP_R_ID1"/>
      </resource>
      <resource identifier="R_ID2" adlcp:scormType="sco" href="index_1.htm">
         <file href="index.htm"/>
         <file href="image4.gif"/>
         <dependency identifierref="DEP_R_ID1"/>
      </resource>
      <resource identifier="DEP_R_ID1" adlcp:scormType="asset">
         <file href="image1.jpg"/>
         <file href="image2.jpg"/>
         <file href="image3.jpg"/>
         <file href="apiWrapper.js"/>
      </resource>
   </resources>
</manifest>
```

Code Illustration 3-32: Dependency Resource Example

In *Code Illustration 3-32: Dependency Resource Example*, a resource was created to hold the commonly used set of files. The resource was given a unique identifier, as required. The newly created resource is an asset. No `href` attribute was provided by the resource. The asset will never be launched by an LMS (no `<item>` identifierref references the resource). The resources that share these sets of files now contain a `<dependency>` element that references, using the `identifierref` attribute, the newly created resource.

This page intentionally left blank.

SECTION 4
Metadata

This page intentionally left blank.

4.1. Metadata Overview

Up to this point, SCORM has described the basic building blocks (SCORM Content Model Components) for content development. SCORM has also described how to bundle the building blocks into Content Aggregations and package those pieces for distribution from system to system. Once the SCORM Content Model Components have been built, it may be useful to describe those components in a consistent manner. Describing the components with metadata facilitates the search and discovery of the components across systems. An LMS could use the metadata to give the learner information about the content organization (i.e., course, lesson, module, etc.). Metadata can also be used at run-time to help in the decision of what content model component to deliver to the learner.

This section provides specific requirements and guidance for using metadata to describe SCORM Content Model Components. The metadata defined in this section is directly based on the IEEE 1484.12.1-2002 Learning Object Metadata (LOM) [11] standard and the IEEE 1484.12.3 Standard for Extensible Markup Language (XML) Binding for Learning Object Metadata Data Model [14]. The IEEE provides roughly 64 metadata elements that can be used to describe SCORM Content Model Components. SCORM strongly recommends the use of the IEEE LOM for describing SCORM Content Model Components

There are currently no requirements defined in SCORM for the creation of metadata and the association of that metadata to the specific SCORM Content Model Components. SCORM recognizes that the IEEE LOM is the defacto standard for metadata within the learning community and strongly recommends that LOM be used when describing SCORM Content Model Components. Although the Content Package Manifest provides a means for associating metadata instances to the SCORM Content Model Components, SCORM does not require such use. Organizations should work to define use cases and requirements around the creation of metadata and the association to SCORM Content Model Components. For example, the U.S. Department of Defense (DoD) provides the ADL Registry for registering content objects across the DoD and related organizations to enable search and discovery of registered content. The ADL Registry provides a specific application profile of LOM that defines additional requirements and practices for using IEEE LOM when registering content within the ADL Registry. For more information on the ADL Registry's use of IEEE LOM refer to https://adlregistry.dtic.mil. ADL encourages communities of practice and organizations to determine the set of elements to be used, additional requirements and practices around their use and any constraints.

The following section is broken up into five basic subsections each describing a different aspect of Metadata and how it relates to SCORM:

- Section 4.1: *Metadata Overview*. This section provides a general overview and background information on LOM and its relationship to SCORM.

- Section 4.2: *LOM Metadata Creation***.** This section defines requirements and guidance for creating metadata. The section provides the details on the requirements defined by IEEE and the details on building XML instances adhering to the IEEE LOM requirements.
- Section 4.3: *LOM XML Schema Validation Approaches***.** This section describes the validation approaches developed by IEEE and described in the IEEE Standard for XML Binding for LOM Data Model. The validation approaches provide different support for XML validation requirements depending on user needs.
- Section 4.4: *Metadata Extensions***.** This section describes the extension capabilities defined by IEEE Standard for XML Binding for Learning Object Metadata Data Model and SCORM. The section also discusses the pros and cons to creating extensions.
- Section 4.5: *Metadata and SCORM Content Model Components***.** This section describes one method for how metadata could be associated with the SCORM Content Model Components.

The purpose of metadata is to provide a common nomenclature enabling learning resources to be described in a common way. Metadata can be collected in catalogs, as well as directly packaged with the learning resource it describes. Learning resources that are described with metadata can be systematically searched for and retrieved for use and reuse.

SCORM describes how the IEEE LOM metadata element can be mapped to SCORM Content Model Components described in the Content Aggregation Model. This mapping of standardized elements and definitions from IEEE to the SCORM CAM provides the missing link between general specifications and specific content models.

4.2. LOM Metadata Creation

The following sections outline the LOM XML metadata elements. According to the IEEE, every LOM metadata element is optional. This implies that when building an XML metadata instance, the developer can optionally pick and choose which elements to use.

The IEEE LOM Information Model describes the set of data elements that are available to build metadata. The LOM Information Model is broken up into nine categories. These categories are based on the definitions found in the LOM Information Model. The nine categories of metadata elements are:

1. The *General* category can be used to describe general information about the SCORM Content Model Component as a whole.
2. The *Life Cycle* category can be used to describe features related to the history and current state of the SCORM Content Model Component and those who have affected the component during its evolution.
3. The *Meta-metadata* category can be used to describe information about the metadata record itself (rather than the SCORM Content Model Component that the record describes).
4. The *Technical* category can be used to describe technical requirements and characteristics of the SCORM Content Model Components.
5. The *Educational* category can be used to describe the educational and pedagogic characteristics of the SCORM Content Model Component.
6. The *Rights* category can be used to describe the intellectual property rights and conditions of use for the SCORM Content Model Component.
7. The *Relation* category can be used to describe features that define the relationship between this SCORM Content Model Component and other targeted components.
8. The *Annotation* category can be used to provide comments on the educational use of the SCORM Content Model Component and information on when and by whom the comments were created.
9. The *Classification* category can be used to describe where the SCORM Content Model Component falls within a particular classification system.

Some elements use the term smallest permitted maximum (SPM) in describing the multiplicity and/or data types. The SPM indicates that applications that process metadata shall process at least that number of elements or number of characters, but are free to support and exceed the limit.

Table 4.2a: Explanation of Multiplicity Requirements

Multiplicity Requirement	Explanation
1 and only 1	The element must exist 1 and only 1 time within the parent element
0 or More	The element can exist 0 or More times within the parent element.

1 or More	The element must exist 1 or More times within the parent element.
0 or 1	The element can exist 0 or 1 time within the parent element.

Table 4.2a defines the types of multiplicities that are used in this section. Each type is accompanied by a brief explanation. The table also has definitions for SPMs for those elements that have a multiplicity of more than 1. The SPM indicates the smallest number of elements that must be supported by a processing system.

4.2.1. <lom> Element

All metadata instances shall have `<lom>` as the root node. The root node begins to define the metadata used to describe the SCORM Content Model Component. When placed in a content package (refer to Section 4.5.1: *Associating Metadata with SCORM Components*), all metadata is placed within a `<imscp:metadata>` element (refer to Section 3.4.1: *Manifest File*) found in an `imsmanifest.xml` file. The `<lom>` root node encapsulates all of the categories described above. There is no implied order of the nine categories. The child elements can appear in any order.

All namespace declarations should be declared inside the `<lom>` element. This includes any namespaces that are considered extensions to the metadata. Although this is not considered a requirement, based on the XML specifications, ADL considers this to be a "best practice" and urges vendors and tools to provide this information.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Namespace Prefix: `lom`

XML Binding Representation: `<lom>`

Multiplicity Requirements: The `<lom>` element contains important elements that SCORM requires to describe all of the SCORM Content Model Components. If LOM metadata is being used to describe the SCORM Content Model Components then, the `<lom>` element shall appear 1 and only 1 time.

ADL Note: The multiplicity requirements for the `<lom>` element imply that the `<lom>` element (the root element of a LOM Instance) shall appear 1 and only 1 time for a given LOM Instance. More than one LOM Instance could be used to describe a given SCORM Content Model Component. In these cases, the `<lom>` element may appear 0 or More times as a direct descendant of the `<imscp:metadata>` element (refer to Section 3.4.1.2 *<metadata> Element* for more information).

Data Type: The `<lom>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<lom>` element contains the following child elements:

- `<general>`
- `<lifeCycle>`
- `<metaMetadata>`
- `<technical>`
- `<educational>`
- `<rights>`
- `<relation>`
- `<annotation>`
- `<classification>`

Example:

The example is used to illustrate the concepts described above. The nine category elements are represented as empty elements for simplicity.

```
<lom xmlns="http://ltsc.ieee.org/xsd/LOM">
   <general/>
   <classification/>
   <annotation/>
   <lifeCycle/>
   <technical/>
   <metaMetadata/>
   <educational/>
   <relation/>
   <rights/>
</lom>
```

Code Illustration 4-1: <lom> Element

4.2.2. <general> Element

The General category groups the general information that describes the resource as a whole. The resource in this case is the particular SCORM Content Model Component (Asset, SCO, Activity, Content Organization or Content Aggregation) being described. This general information is sometimes viewed as key information in that it is important for describing the particular component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<general>`

Multiplicity Requirements: The `<general>` element shall appear 0 or 1 time.

Data Type: The `<general>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<general>` element contains the following child elements:

- `<identifier>`
- `<title>`
- `<language>`
- `<description>`
- `<keyword>`
- `<coverage>`
- `<structure>`
- `<aggregationLevel>`

Example:

```
<lom>
   <general>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/content/CO_01</entry>
      </identifier>
      <title>
         <string language="en">Title for the learning object</string>
      </title>
      <language>en</language>
      <description>
         <string language="en">Textual description</string>
      </description>
      <keyword>
         <string language="en">learning object</string>
      </keyword>
      <coverage>
         <string language="en">Circa, 16th century France</string>
      </coverage>
      <structure>
         <source>LOMv1.0</source>
         <value>atomic</value>
      </structure>
      <aggregationLevel>
         <source>LOMv1.0</source>
         <value>2</value>
      </aggregationLevel>
   </general>
</lom>
```

Code Illustration 4-2: <general> Element

4.2.2.1. <identifier> Element

The `<identifier>` element represents a mechanism for assigning a globally unique label that identifies the SCORM Content Model Component. The notion of assigning a globally unique identifier to a component is important when dealing with multiple facets of learning content development (e.g., versioning, maintenance, etc.).

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<identifier>`

Multiplicity Requirements: The `<identifier>` element shall appear 0 or More times. The `<identifier>` element has an SPM of 10.

Data Type: The `<identifier>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<identifier>` element contains the following child elements:

- `<catalog>`
- `<entry>`

Example:

```
<lom>
   <general>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/content/CO_01</entry>
      </identifier>
   </general>
</lom>
```

Code Illustration 4-3: <identifier> Element

4.2.2.1.1. <catalog> Element

The `<catalog>` element represents the name or designator of the identification or cataloging scheme for the entry. There are a variety of cataloging systems available. SCORM does not require the use of any one particular cataloging system. Organizations are free to choose any cataloging scheme that meets their organizations practices or policies. Some types of cataloging systems are:

- Universal Resource Identifier (URI)
- Universal Resource Name (URN)
- Digital Object Identifier (DOI)
- International Standard Book Numbers (ISBN)
- International Standard Serial Numbers (ISSN).

The `<catalog>` element represents the scheme used to create and manage the entry.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<catalog>`

Multiplicity Requirements: The `<catalog>` element shall appear 0 or 1 time.

Data Type: The `<catalog>` element is represented as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <general>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/content/CO_01</entry>
      </identifier>
   </general>
</lom>
```

Code Illustration 4-4: <catalog> Element

4.2.2.1.2. <entry> Element

The `<entry>` element represents the value of the identifier within the identification or cataloging scheme (refer to the `<catalog>` element) that designates or identifies the learning object.

Identifiers can take on various formats. The IEEE requires that the actual identifier value be represented as a CharacterString. Organizations are free to choose any mechanism to create unique identifiers that meets their organizations practices or policies.

The following listing is a sampling of identifier values (entry):

Scheme (`<catalog>`)	Value (`<entry>`)
Universal Resource Name	urn:ADL: 1345-GFGC-23ED-3321
Universal Resource Identifier	http://www.adlnet.gov/content/C0_01
Handle Syntax	100.100/2345342256349543

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<entry>`

Multiplicity Requirements: The `<entry>` element shall appear 0 or 1 time.

Data Type: The `<entry>` element's value is represented as a CharacterString. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <general>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/content/CO_01</entry>
      </identifier>
   </general>
</lom>
```

Code Illustration 4-5: <entry> Element

4.2.2.2. <title> Element

The `<title>` element represents the name given to the learning object.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<title>`

Multiplicity Requirements: The `<title>` element shall appear 0 or 1 time.

Data Type: The `<title>` element is represented as a LangString element. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <general>
      <title>
         <string language="en">Sharable Content Object Reference Model</string>
      </title>
   </general>
</lom>
```

Code Illustration 4-6: <title> Element

4.2.2.3. <language> Element

The `<language>` element represents the primary human language or languages used within the SCORM Content Model Component to communicate to the intended user. The Language element can be repeated. This allows for the ability to describe components that are built to support multiple languages.

The value held by the `<language>` element shall be represented according to the following:

`Language = Langcode("-"Subcode)*`

`Langcode` – Represents a language code as defined by ISO 639:1988. This value is mandatory.

`Subcode` – Represents a country code from the code set defined by ISO 3166-1997. This value can be repeated and is optional.

The `<language>` element may also have a value of `none`, if the SCORM Content Model Component has no lingual content.

Examples:
- "en"
- "en-GB"

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<language>`

Multiplicity Requirements: The `<language>` element shall appear 0 or More times. The `<language>` element has an SPM of 10.

Data Type: The `<language>` element is represented as a CharacterString element. The CharacterString has an SPM of 100 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <general>
      <language>en</language>
      <language>fr</language>
   </general>
</lom>
```

Code Illustration 4-7: <language> Element

4.2.2.4. <description> Element

The `<description>` element represents a textual description of the SCORM Content Model Component being described by the metadata. The Description element allows for a narrative description of the component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<description>`

Multiplicity Requirements: The `<description>` element shall appear 0 or More times. The `<description>` element has an SPM of 10.

Data Type: The `<description>` element is represented as a LangString element. The LangString has an SPM of 2000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <general>
      <description>
         <string language="en">Textual description of the learning
object</string>
      </description>
   </general>
</lom>
```

Code Illustration 4-8: <description> Element

4.2.2.5. <keyword> Element

The `<keyword>` element shall be used to define common keywords or phrases that describe the learning object. When creating keyword(s), the creator should pick words or phrases that are very succinct and specific to the SCORM component. The Keyword element consists of one word or phrase. If more than one Keyword is desired, the creator should use multiple instances of the Keyword element.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<keyword>`

Multiplicity Requirements: The `<keyword>` element shall appear 0 or More times. The `<keyword>` element has an SPM of 10.

Data Type: The `<keyword>` element is represented as a LangString element. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <general>
      <keyword>
         <string language="en">learning object</string>
         <string language="nl">leerobject</string>
         <string language="fr">objet d'apprentissage</string>
      </keyword>
      <keyword>
         <string language="en">metadata</string>
         <string language="nl">metadata</string>
         <string language="fr">métadonnées</string>
      </keyword>
   </general>
</lom>
```

Code Illustration 4-9: <keyword> Element

4.2.2.6. <coverage> Element

The `<coverage>` element shall be used to describe the time, culture, geography or region to which the SCORM Content Model Component applies.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<coverage>`

Multiplicity Requirements: The `<coverage>` element shall appear 0 or More times. The `<coverage>` element has an SPM of 10.

Data Type: The `<coverage>` element is represented as a LangString element. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <general>
      <coverage>
         <string language="en">Circa, 16th century France</string>
      </coverage>
   </general>
</lom>
```

Code Illustration 4-10: <coverage> Element

4.2.2.7. <structure> Element

The `<structure>` element shall describe the underlying organizational structure of the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<structure>`

Multiplicity Requirements: The `<structure>` element shall appear 0 or 1 time.

Data Type: The `<structure>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`atomic`**: An object that is indivisible.
- **`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.

Example:

```
<lom>
   <general>
      <structure>
         <source>LOMv1.0</source>
         <value>atomic</value>
      </structure>
   </general>
</lom>
```

Code Illustration 4-11: <structure> Element

4.2.2.8. <aggregationLevel> Element

The `<aggregationLevel>` element shall describe the functional granularity of the learning object.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<aggregationLevel>`

Multiplicity Requirements: The `<aggregationLevel>` element shall appear 0 or 1 time.

Data Type: The `<aggregationLevel>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- `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.

Example:

```
<lom>
   <general>
      <aggregationLevel>
         <source>LOMv1.0</source>
         <value>2</value>
      </aggregationLevel>
   </general>
</lom>
```

Code Illustration 4-12: <aggregationLevel> Element

4.2.3. <lifeCycle> Element

The Life Cycle category groups the features related to the history and current state of the SCORM Content Model Component and those who have affected the component during its evolution. The typical types of information collected in this category include the status of the component (e.g., is the component in its final state or is it still in a draft format), a version identifier indicating the version of the component and a list of individuals/organizations that have affected the component in one manner or another.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<lifeCycle>`

Multiplicity Requirements: The `<lifeCycle>` element shall appear 0 or 1 time.

Data Type: The `<lifeCycle>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<lifeCycle>` element contains the following child elements:

- `<version>`
- `<status>`
- `<contribute>`

Example:

```
<lom>
   <lifeCycle>
      <version>
         <string language="en">1.0 alpha</string>
      </version>
      <status>
         <source>LOMv1.0</source>
         <value>final</value>
      </status>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>author</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Friday&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">A description for the date</string>
            </description>
         </date>
      </contribute>
   </lifeCycle>
</lom>
```

Code Illustration 4-13: <lifeCycle> Element

4.2.3.1. <version> Element

The <version> element shall describe the edition of the SCORM Content Model Component. A component may have several versions or editions during its lifetime. The <version> element allows for the description of the version of the component.

XML Namespace: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <version>

Multiplicity Requirements: The <version> element shall appear 0 or 1 time.

Data Type: The <version> element is represented as a LangString element. The LangString has an SPM of 50 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <lifeCycle>
      <version>
         <string language="en">1.0 alpha</string>
      </version>
   </lifeCycle>
</lom>
```

Code Illustration 4-14: <version> Element

4.2.3.2. <status> Element

The <status> element shall describe the completion status or condition of the SCORM Content Model Component. A component's status may change during its lifetime (draft, final, etc.). The <status> element allows for the description of the status of the component.

XML Namespace: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <status>

Multiplicity Requirements: The <status> element shall appear 0 or 1 time.

Data Type: The <status> element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **draft**: The component is in a draft state (as determined by the developer).
- **final**: The component is in a final state (as determined by the developer).
- **revised**: The component has been revised since the last version.
- **unavailable**: The status information is unavailable.

Example:

```
<lom>
   <lifeCycle>
      <status>
         <source>LOMv1.0</source>
         <value>final</value>
      </status>
   </lifeCycle>
</lom>
```

Code Illustration 4-15: <status> Element

4.2.3.3. <contribute> Element

The `<contribute>` element shall be used to describe those entities (i.e., people, organizations) that have contributed to the state of the SCORM Content Model Component during its lifecycle (e.g., creation, edits, reviews, publications, etc.). The Contribute element enables capturing of all those individuals or organizations involved.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<contribute>`

Multiplicity Requirements: The `<contribute>` element shall appear 0 or More times. The `<contribute>` element has an SPM of 30.

Data Type: The `<contribute>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<contribute>` element contains the following child elements:

- `<role>`
- `<entity>`
- `<date>`

Example:

```
<lom>
   <lifeCycle>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>author</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Friday&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">A description for the date</string>
            </description>
         </date>
      </contribute>
   </lifeCycle>
</lom>
```

Code Illustration 4-16: <contribute> Element

4.2.3.3.1. <role> Element

The `<role>` element defines the kind or type of contribution made by the contributor (identified by the Entity element). The IEEE has defined a set of typical roles that are involved with the lifecycle of the component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<role>`

Multiplicity Requirements: The `<role>` element shall appear 0 or 1 time.

Data Type: The `<role>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`author`**
- **`publisher`**
- **`unknown`**
- **`initiator`**
- **`terminator`**
- **`validator`**
- **`editor`**
- **`graphical designer`**
- **`technical implementer`**
- **`content provider`**
- **`technical validator`**
- **`educational validator`**
- **`script writer`**
- **`instructional designer`**
- **`subject matter expert`**

Example:

```
<lom>
   <lifeCycle>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>author</value>
         </role>
      </contribute>
   </lifeCycle>
</lom>
```

Code Illustration 4-17: <role> Element

4.2.3.3.2. <entity> Element

The `<entity>` element identifies the entity or entities that may have contributed during the development lifecycle of the SCORM Content Model Component. An entity can be an individual person, organization, etc. If more than one entity is listed, the entities shall be ordered as most relevant first.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<entity>`

Multiplicity Requirements: The `<entity>` element shall appear 0 or More times. The `<entity>` element has an SPM 40.

Data Type: The `<entity>` element is a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information). All entity values shall be represented in vCard [9] format. This allows systems to take the CharacterString represented by the `<entity>` element and process this CharacterString as a valid vCard.

Example:

```
<lom>
   <lifeCycle>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>author</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Author&#13;&#10;END:VCARD</entity>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Mary
Author&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">A description for the date</string>
            </description>
         </date>
      </contribute>
   </lifeCycle>
</lom>
```

Code Illustration 4-18: <entity> Element

4.2.3.3.3. <date> Element

The `<date>` element identifies the date of the contribution made by the entity.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<date>`

Multiplicity Requirements: The `<date>` element shall appear 0 or 1 time.

Data Type: The Date element is represented as a DateTime data type (refer to Section 4.2.11.4: *DateTime Data Type* for more information). The `<date>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<date>` element contains two elements, one that represents the actual date of the contribution (`<dateTime>`) and one that represents a textual description of the date (`<description>`):

- `<dateTime>`
- `<description>`

Example:

```
<lom>
   <lifeCycle>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>author</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Author&#13;&#10;END:VCARD</entity>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Mary
Author&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">This date represents the date the author
finished authoring the component.</string>
            </description>
         </date>
      </contribute>
   </lifeCycle>
</lom>
```

Code Illustration 4-19: LifeCycle Contribute <date> Element

4.2.4. <metaMetadata> Element

The Meta-Metadata category provides elements that describe the metadata record itself and not the SCORM Content Model Component the record is describing. This category describes how the metadata instance itself can be identified, who created the metadata instance, how, when and with what references.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<metaMetadata>`

Multiplicity Requirements: The `<metaMetadata>` element shall appear 0 or 1 time.

Data Type: The `<metaMetadata>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<metaMetadata>` element contains the following child elements:

- `<identifier>`
- `<contribute>`
- `<metadataSchema>`
- `<language>`

Example:

```
<lom>
   <metaMetadata>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/metadata/MDO_01</entry>
      </identifier>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>creator</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe Metadata
Creator&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">This date represents the date the creator
finished authoring the metadata.</string>
            </description>
         </date>
      </contribute>
      <metadataSchema>LOMv1.0</metadataSchema>
      <metadataSchema>ADLv1.0</metadataSchema>
      <language>en</language>
   </metaMetadata>
</lom>
```

Code Illustration 4-20: <metaMetadata> Element

4.2.4.1. <identifier> Element

The `<identifier>` element represents a mechanism for assigning a globally unique label that identifies the metadata record that describes the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<identifier>`

Multiplicity Requirements: The `<identifier>` element shall appear 0 or More times. The `<identifier>` element has an SPM of 10.

Data Type: The `<identifier>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<identifier>` element contains the following child elements:

- `<catalog>`
- `<entry>`

Example:

```
<lom>
   <metaMetadata>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/metadata/MDO_01</entry>
      </identifier>
   </metaMetadata>
</lom>
```

Code Illustration 4-21: Meta-Metadata <identifier> Element

4.2.4.1.1. <catalog> Element

The `<catalog>` element represents the name or designator of the identification or cataloging scheme for the entry. There are a variety of cataloging systems available. SCORM does not require the use of any one particular cataloging system. Organizations are free to choose any cataloging scheme that meets their organizations practices or policies. Some types of cataloging systems are:

- Universal Resource Identifier (URI)
- Universal Resource Name (URN)
- Digital Object Identifier (DOI)
- International Standard Book Numbers (ISBN)
- International Standard Serial Numbers (ISSN).

The `<catalog>` element represents the scheme used to create and manage the entry.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<catalog>`

Multiplicity Requirements: The `<catalog>` element shall appear 0 or 1 time.

Data Type: The `<catalog>` element is represented as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <metaMetadata>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/metadata/MDO_01</entry>
      </identifier>
   </metaMetadata>
</lom>
```

Code Illustration 4-22: Meta-Metadata Identifier <catalog> Element

4.2.4.1.2. <entry> Element

The `<entry>` element represents the value of the identifier within the identification or cataloging scheme (refer to the `<catalog>` element) that designates or identifies the metadata.

Identifiers can take on various formats. The IEEE requires that the actual identifier value be represented as a CharacterString. Organizations are free to choose any mechanism to create unique identifiers that meets their organizations practices or policies. It is recommended that a common scheme be chosen.

The following listing is a sampling of identifier values (entry):

Scheme (`<catalog>`)	Value (`<entry>`)
Universal Resource Name	urn:ADL: 1345-GFGC-23ED-3321
Universal Resource Identifier	http://www.adlnet.gov/content/C0_01
Handle Syntax	100.100/2345342256349543

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<entry>`

Multiplicity Requirements: The `<entry>` element shall appear 0 or 1 time.

Data Type: The `<entry>` element is represented as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <metaMetadata>
      <identifier>
         <catalog>URI</catalog>
         <entry>http://www.adlnet.gov/metadata/MDO_01</entry>
      </identifier>
   </metaMetadata>
</lom>
```

Code Illustration 4-23: Meta-Metadata Identifier <entry> Element

4.2.4.2. <contribute> Element

The `<contribute>` element shall be used to describe those entities (i.e., people, organizations) that have affected the state of the metadata (not the SCORM Content Model Component being described) instance during its development lifecycle. The `<contribute>` element enables capturing of all those individuals or organizations involved.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<contribute>`

Multiplicity Requirements: The `<contribute>` element shall appear 0 or More times. The `<contribute>` element has an SPM of 10.

Data Type: The `<contribute>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<contribute>` element contains the following child elements:

- `<role>`
- `<entity>`
- `<date>`

Example:

```
<lom>
   <metaMetadata>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>creator</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe Metadata
Creator&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">This date represents the date the creator
finished authoring the metadata.</string>
            </description>
         </date>
      </contribute>
   </metaMetadata>
</lom>
```

Code Illustration 4-24: Meta-Metadata <contribute> Element

4.2.4.2.1. <role> Element

The `<role>` element defines the kind or type of contribution made by the contributor (identified by the Entity element). The IEEE has defined a set of typical roles that are involved with the development lifecycle of the metadata instance.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<role>`

Multiplicity Requirements: The `<role>` element shall appear 0 or 1 time.

Data Type: The `<role>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`creator`**
- **`validator`**

Example:

```
<lom>
   <metaMetadata>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>creator</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe Metadata
Creator&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">This date represents the date the creator
finished authoring the metadata.</string>
            </description>
         </date>
      </contribute>
   </metaMetadata>
</lom>
```

Code Illustration 4-25: Meta-Metadata Contribute <role> Element

4.2.4.2.2. <entity> Element

The `<entity>` element identifies the entity or entities that may have contributed during the development lifecycle of the metadata instance. An entity can be an individual person, organization, etc. If more than one entity is listed, the entities shall be ordered as most relevant first.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<entity>`

Multiplicity Requirements: The `<entity>` element shall appear 0 or More times. The `<entity>` element has an SPM of 10.

Data Type: The `<entity>` element as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information). All entity values shall be represented in vCard [9] format. This allows systems to take the CharacterString represented by the `<entity>` element and process this CharacterString as a valid vCard.

Example:

```
<lom>
   <metaMetadata>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>creator</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe Metadata
Creator&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">This date represents the date the creator
finished authoring the metadata.</string>
            </description>
         </date>
      </contribute>
   </metaMetadata>
</lom>
```

Code Illustration 4-26: Meta-Metadata Contribute <entity> Element

4.2.4.2.3. <date> Element

The `<date>` element identifies the date of the contribution made by the entity.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<date>`

Multiplicity Requirements: The `<date>` element shall appear 0 or 1 time.

Data Type: The Date element is represented as a DateTime data type (refer to Section 4.2.11.4: *DateTime Data Type* for more information). The `<date>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<date>` element contains two elements, one that represents the actual date of the contribution (`<dateTime>`) and one that represents a textual description of the date (`<description>`):

- `<dateTime>`
- `<description>`

Example:

```
<lom>
   <metaMetadata>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>creator</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe Metadata 
Creator&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">This date represents the date the creator 
finished authoring the metadata.</string>
            </description>
         </date>
      </contribute>
   </metaMetadata>
</lom>
```

Code Illustration 4-27: Meta-Metadata Contribute <date> Element

4.2.4.3. <metadataSchema> Element

The `<metadataSchema>` element represents the name and version of the authoritative specification used to create the metadata instance. If multiple values are provided, then the metadata instance shall conform to multiple metadata schemas.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<metadataSchema>`

Multiplicity Requirements: The `<metadataSchema>` element shall appear 0 or More times. The `<metadataSchema>` element has an SPM of 10.

If metadata instances are created that conform to the IEEE LOM XML Binding and the `<metadataSchema>` element is present, the IEEE standard requires you to identify that it is conforming to the standard. This requires the `<metadataSchema>` element to contain the value of `LOMv1.0`.

Data Type: The `<metadataSchema>` element is represented as a CharacterString element. The CharacterString has an SPM of 30 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <metaMetadata>
      <metadataSchema>LOMv1.0</metadataSchema>
   </metaMetadata>
</lom>
```

Code Illustration 4-28: <metadataSchema> Element

4.2.4.4. <language> Element

The `<language>` element represents the language of the metadata instance (i.e., the language of all values found in LangStrings). This value represents the default language for all LangStrings. If a value for this data element is not present in a metadata instance, then there is no default language for LangString values. If this value is provided, it is not necessary to indicate a language value for LangString elements.

The value held by the `<language>` element shall be represented according to the following:

`Language = Langcode("-"Subcode)*`

`Langcode` – Represents a language code as defined by ISO 639:1988. This value is mandatory.

`Subcode` – Represents a country code from the code set defined by ISO 3166-1997. This value can be repeated and is optional.

Examples:

- "en"
- "en-GB"

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<language>`

Multiplicity Requirements: The `<language>` element shall appear 0 or 1 time.

Data Type: The `<language>` element is represented as a CharacterString element. The CharacterString has an SPM of 100 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <metaMetadata>
      <language>en</language>
   </metaMetadata>
</lom>
```

Code Illustration 4-29: Meta-Metadata <language> Element

4.2.5. <technical> Element

The Technical category describes all of the technical characteristics and requirements of the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<technical>`

Multiplicity Requirements: The `<technical>` element shall appear 0 or 1 time.

Data Type: The `<technical>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<technical>` element contains the following child elements:

- `<format>`
- `<size>`
- `<location>`
- `<requirement>`
- `<installationRemarks>`
- `<otherPlatformRequirements>`
- `<duration>`

Example:

```
<lom>
   <technical>
      <format>text/html</format>
      <size>1024</size>
      <location>Lesson01/Module01/Resources/SCO01.htm</location>
      <requirement>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>ms-internet explorer</value>
            </name>
            <minimumVersion>5.0</minimumVersion>
            <maximumVersion>6.0</maximumVersion>
         </orComposite>
      </requirement>
      <installationRemarks>
         <string language="en">This activity requires the client browser to
have a Macromedia Flash plugin installed.</string>
      </installationRemarks>
      <otherPlatformRequirements>
         <string language="en">Sound card, Min. RAM: 16Mb, Video card and
display: at least 800 X 600 pixels x 256 colors</string>
      </otherPlatformRequirements>
      <duration>
         <duration>P5Y</duration>
         <description>
            <string language="en">Length of time to play simulation</string>
         </description>
      </duration>
   </technical>
</lom>
```

Code Illustration 4-30: <technical> Element

4.2.5.1. <format> Element

The `<format>` element represents the technical datatype(s) of all of the components used in the makeup of the SCORM Content Model Component. This element is used to identify any potential software needs to access and use the component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<format>`

Multiplicity Requirements: The `<format>` element shall appear 0 or More times. The `<format>` element has an SPM of 40.

Data Type: The `<format>` element is represented as a CharacterString element. The CharacterString has an SPM of 500 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information). The CharacterString shall be a MIME type based on IANA registration (refer to RFC 2048:1996) or the string literal, "non-digital".

Example:

```
<lom>
   <technical>
      <format>video/mpeg</format>
      <format>text/html</format>
   </technical>
</lom>
```

Code Illustration 4-31: <format> Element

4.2.5.2. <size> Element

The `<size>` element represents the size of the digital SCORM Content Model Component in bytes. The size is represented as a decimal value (radix 10). Only the digits "0" through "9" should be used. This data element shall refer to the actual size of the SCORM Component. If the component is compressed, then this data element shall refer to the uncompressed size.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<size>`

Multiplicity Requirements: The `<size>` element shall appear 0 or 1 time.

Data Type: The `<size>` element is represented as a CharacterString element. The CharacterString has an SPM of 30 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <technical>
      <size>345</size>
   </technical>
</lom>
```

Code Illustration 4-32: <size> Element

4.2.5.3. <location> Element

The `<location>` element is a string that specifies the location of the SCORM Content Model Component described by the metadata.

This element could be used to describe one or more alternate locations where the SCORM Content Model Component can also be found, in addition to the content package itself (e.g., a known fixed location).

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<location>`

Multiplicity Requirements: The `<location>` element shall appear 0 or More times. The `<location>` element has an SPM of 10.

Data Type: The `<location>` element is represented as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <technical>
      <location>http://www.adlnet.gov/content/Asset.jpg</location>
   </technical>
</lom>
```

Code Illustration 4-33: <location> Element

4.2.5.4. <requirement> Element

The `<requirement>` element expresses the technical capabilities necessary for using the SCORM Content Model Component. The `<requirement>` element is repeatable. If multiple requirements are needed, then all of the requirements are required (logical connector is an AND).

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<requirement>`

Multiplicity Requirements: The `<requirement>` element shall appear 0 or More times. The `<requirement>` element has an SPM of 40.

Data Type: The `<requirement>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<requirement>` element contains the following child elements:

- `<orComposite>`

Example:

```
<lom>
   <technical>
      <requirement>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>ms-internet explorer</value>
            </name>
            <minimumVersion>5.0</minimumVersion>
            <maximumVersion>6.0</maximumVersion>
         </orComposite>
      </requirement>
   </technical>
</lom>
```

Code Illustration 4-34: <requirement> Element

4.2.5.4.1. <orComposite> Element

The `<orComposite>` element represents a single requirement. Multiple `<orComposite>` elements are connected with a logical connecter of "`or`".

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<orComposite>`

Multiplicity Requirements: The `<orComposite>` element shall appear 0 or More times. The `<orComposite>` element has an SPM of 40.

Data Type: The `<orComposite>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<orComposite>` element contains the following child elements:

- `<type>`
- `<name>`
- `<minimumVersion>`
- `<maximumVersion>`

Example: The following metadata indicates that the component being described will execute in either of the following Web browsers:

- Microsoft Internet Explorer
  - Minimum Version: 5.0
  - Maximum Version 6.0

or

- Netscape Communicator
  - Minimum Version: 4.7.9
  - Maximum Version 5.0

```
<lom>
   <technical>
      <requirement>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>ms-internet explorer</value>
            </name>
            <minimumVersion>5.0</minimumVersion>
            <maximumVersion>6.0</maximumVersion>
         </orComposite>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>netscape communicator</value>
            </name>
            <minimumVersion>4.7.9</minimumVersion>
            <maximumVersion>5.0</maximumVersion>
         </orComposite>
      </requirement>
   </technical>
</lom>
```

Code Illustration 4-35: <orComposite> Element

4.2.5.4.1.1. <type> Element

The `<type>` element represents the technology required to use the SCORM Content Model Component (e.g., hardware, software, network, etc.).

If the metadata instance contains a `<type>` element then a corresponding `<name>` element shall exist to describe more details about the type.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<type>`

Multiplicity Requirements: The `<type>` element shall appear 0 or 1 time.

Data Type: The `<type>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`operating system`**
- **`browser`**

Example:

```
<lom>
   <technical>
      <requirement>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>ms-internet explorer</value>
            </name>
            <minimumVersion>5.0</minimumVersion>
            <maximumVersion>6.0</maximumVersion>
         </orComposite>
      </requirement>
   </technical>
</lom>
```

Code Illustration 4-36: <type> Element

4.2.5.4.1.2. <name> Element

The `<name>` element represents the required technology to use the SCORM Content Model Component. The value used for the Name element depends on the value identified by the Value element.

If the metadata instance contains a `<name>` element then a corresponding `<type>` element shall exist to describe more details about the required technology.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<name>`

Multiplicity Requirements: The `<name>` element shall appear 0 or 1 time.

Data Type: The `<name>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

If Requirement.OrComposite.Type = “`operating system`”:

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

If Requirement.OrComposite.Type = “`browser`”:

- **`any`**
- **`netscape communicator`**

- **ms-internet explorer**
- **opera**
- **amaya**

Example:

```
<lom>
   <technical>
      <requirement>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>ms-internet explorer</value>
            </name>
            <minimumVersion>5.0</minimumVersion>
            <maximumVersion>6.0</maximumVersion>
         </orComposite>
      </requirement>
   </technical>
</lom>
```

Code Illustration 4-37: <name> Element

4.2.5.4.1.3. <minimumVersion> Element

The `<minimumVersion>` element represents the lowest possible version of the required technology to use the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<minimumVersion>`

Multiplicity Requirements: The `<minimumVersion>` element shall appear 0 or 1 time.

Data Type: The `<minimumVersion>` element is represented as a CharacterString. The CharacterString has an SPM of 30 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <technical>
      <requirement>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>ms-internet explorer</value>
            </name>
            <minimumVersion>5.0</minimumVersion>
            <maximumVersion>6.0</maximumVersion>
         </orComposite>
      </requirement>
   </technical>
</lom>
```

Code Illustration 4-38: <minimumVersion> Element

4.2.5.4.1.4. <maximumVersion> Element

The `<maximumVersion>` element represents the highest possible version of the required technology to use the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<maximumVersion>`

Multiplicity Requirements: The `<maximumVersion>` element shall appear 0 or 1 time.

Data Type: The `<maximumVersion>` element is represented as a CharacterString. The CharacterString has an SPM of 30 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <technical>
      <requirement>
         <orComposite>
            <type>
               <source>LOMv1.0</source>
               <value>browser</value>
            </type>
            <name>
               <source>LOMv1.0</source>
               <value>ms-internet explorer</value>
            </name>
            <minimumVersion>5.0</minimumVersion>
            <maximumVersion>6.0</maximumVersion>
         </orComposite>
      </requirement>
   </technical>
</lom>
```

Code Illustration 4-39: <maximumVersion> Element

4.2.5.5. <installationRemarks> Element

The `<installationRemarks>` element is used to represent any specific instructions on how to install the SCORM Content Model Component. This element could be used to describe to the user (e.g., LMS, Content Developer, Authoring Tool) of the component any particular instructions for use. It may be used to describe, in more detail, the technical requirements for the SCORM component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<installationRemarks>`

Multiplicity Requirements: The `<installationRemarks>` element shall appear 0 or 1 time.

Data Type: The `<installationRemarks>` element is represented as a LangString. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <technical>
      <installationRemarks>
         <string language="en">This activity requires the client browser to
have a Macromedia Flash plugin installed.</string>
      </installationRemarks>
   </technical>
</lom>
```

Code Illustration 4-40: <installationRemarks> Element

4.2.5.6. <otherPlatformRequirements> Element

The `<otherPlatformRequirements>` element is used to represent information about other software and hardware requirements of the SCORM Content Model Component. This element should be used to describe requirements that cannot be represented or expressed with the other Technical elements.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<otherPlatformRequirements>`

Multiplicity Requirements: The `<otherPlatformRequirements>` element shall appear 0 or 1 time.

Data Type: The `<otherPlatformRequirements>` element is represented as a LangString. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <technical>
      <otherPlatformRequirements>
         <string language="en">Sound card, Min. RAM: 16Mb, Video card and
display: at least 800 X 600 pixels x 256 colors</string>
      </otherPlatformRequirements>
   </technical>
</lom>
```

Code Illustration 4-41: <otherPlatformRequirements> Element

4.2.5.7. <duration> Element

The `<duration>` element represents the time a continuous SCORM Content Model Component takes when played at intended speed. This element is useful for sounds, movies, simulations and the like.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<duration>`

Multiplicity Requirements: The `<duration>` element shall appear 0 or 1 time.

Data Type: The `<duration>` element is represented as a Duration data type (refer to Section 4.2.11.5: *Duration Data Type* for more information).

Example:

```
<lom>
   <technical>
      <duration>
         <!-- Movie will play for 1 hour and 30 minutes -->
         <duration>PT1H30M</duration>
         <description>
            <string language="en">Length of time to play movie</string>
         </description>
      </duration>
   </technical>
</lom>
```

Code Illustration 4-42: <duration> Element

4.2.6. <educational> Element

The Educational category describes the key educational or pedagogic characteristics of the SCORM Content Model Component. This category allows for the description of the educational characteristics and is typically used by teachers, managers, authors and learners.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<educational>`

Multiplicity Requirements: The `<educational>` element shall appear 0 or More times. The `<educational>` element has an SPM of 100.

Data Type: The `<educational>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<educational>` element contains the following child elements:

- `<interactivityType>`
- `<learningResourceType>`
- `<interactivityLevel>`
- `<semanticDensity>`
- `<intendedEndUserRole>`
- `<context>`
- `<typicalAgeRange>`
- `<difficulty>`
- `<typicalLearningTime>`
- `<description>`
- `<language>`

Example:

```
<lom>
   <educational>
      <interactivityType>
         <source>LOMv1.0</source>
         <value>mixed</value>
      </interactivityType>
      <learningResourceType>
         <source>LOMv1.0</source>
         <value>figure</value>
      </learningResourceType>
      <learningResourceType>
         <source>LOMv1.0</source>
         <value>narrative text</value>
      </learningResourceType>
      <interactivityLevel>
         <source>LOMv1.0</source>
         <value>very low</value>
      </interactivityLevel>
      <semanticDensity>
         <source>LOMv1.0</source>
         <value>very low</value>
      </semanticDensity>
      <intendedEndUserRole>
         <source>LOMv1.0</source>
         <value>learner</value>
      </intendedEndUserRole>
      <context>
         <source>LOMv1.0</source>
         <value>training</value>
      </context>
      <typicalAgeRange>
         <string language="en">18-</string>
      </typicalAgeRange>
      <difficulty>
         <source>LOMv1.0</source>
         <value>easy</value>
      </difficulty>
      <typicalLearningTime>
         <duration>PT1H30M</duration>
         <description>
            <string language="en">Average length of time to experience the
activity.</string>
         </description>
      </typicalLearningTime>
      <language>en-US</language>
   </educational>
</lom>
```

Code Illustration 4-43: <educational> Element

4.2.6.1. <interactivityType> Element

The `<interactivityType>` element represents the dominant mode of learning supported by the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<interactivityType>`

Multiplicity Requirements: The `<interactivityType>` element shall appear 0 or 1 time.

Data Type: The `<interactivityType>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`active`**: Active learning (e.g., learning by doing) is supported by content that directly induces productive action by the learner.
- **`expositive`**: Expositive learning (e.g., passive learning) occurs when the learner's job mainly consists of absorbing the content exposed to them.
- **`mixed`**: A blend of active and expositive interactivity types.

Example:

```
<lom>
   <educational>
      <interactivityType>
         <source>LOMv1.0</source>
         <value>mixed</value>
      </interactivityType>
   </educational>
</lom>
```

Code Illustration 4-44: <interactivityType> Element

4.2.6.2. <learningResourceType> Element

The `<learningResourceType>` element represents the specific kind of the SCORM Content Model Component. This element is repeatable in order to fully describe the types of resources used in the component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<learningResourceType>`

Multiplicity Requirements: The `<learningResourceType>` element shall appear 0 or More times. The `<learningResourceType>` element has an SPM of 10.

Data Type: The `<learningResourceType>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`exercise`**
- **`simulation`**
- **`questionnaire`**
- **`diagram`**
- **`figure`**
- **`graph`**
- **`index`**

- **`slide`**
- **`table`**
- **`narrative text`**
- **`exam`**
- **`experiment`**
- **`problem statement`**
- **`self assessment`**
- **`lecture`**

Example:

```
<lom>
   <educational>
      <learningResourceType>
         <source>LOMv1.0</source>
         <value>narrative text</value>
      </learningResourceType>
      <learningResourceType>
         <source>LOMv1.0</source>
         <value>simulation</value>
      </learningResourceType>
   </educational>
</lom>
```

Code Illustration 4-45: <learningResourceType> Element

4.2.6.3. <interactivityLevel> Element

The `<interactivityLevel>` represents the degree of interactivity characterizing the SCORM Content Model Component. Interactivity in this context refers to the degree to which the learner can influence the aspect or behavior of the component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<interactivityLevel>`

Multiplicity Requirements: The `<interactivityLevel>` element shall appear 0 or 1 time.

Data Type: The `<interactivityLevel>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`very low`**
- **`low`**
- **`medium`**
- **`high`**
- **`very high`**

These values, inherently, are meaningful only in a context of a community practice. At this time, this scale is left to organizations to define.

Example:

```
<lom>
   <educational>
      <interactivityLevel>
         <source>LOMv1.0</source>
         <value>very low</value>
      </interactivityLevel>
   </educational>
</lom>
```

Code Illustration 4-46: <interactivityLevel> Element

4.2.6.4. <semanticDensity> Element

The `<semanticDensity>` represents the degree of conciseness of the SCORM Content Model Component. The semantic density of a SCORM component may be estimated in terms of its size, span or, in the case of self-timed resources such as audio or video, duration.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<semanticDensity>`

Multiplicity Requirements: The `<semanticDensity>` element shall appear 0 or 1 time.

Data Type: The `<semanticDensity>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`very low`**
- **`low`**
- **`medium`**
- **`high`**
- **`very high`**

These values, inherently, are meaningful only in a context of a community practice At this time, this scale is left to organizations to define.

Example:

```
<lom>
   <educational>
      <semanticDensity>
         <source>LOMv1.0</source>
         <value>very low</value>
      </semanticDensity>
   </educational>
</lom>
```

Code Illustration 4-47: <semanticDensity> Element

4.2.6.5. <intendedEndUserRole> Element

The `<intendedEndUserRole>` element represents the principal user(s) for which the SCORM Content Model Component was designed. If multiple elements are used, the most dominant role should be first.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<intendedEndUserRole>`

Multiplicity Requirements: The `<intendedEndUserRole>` element shall appear 0 or More times. The `<intendedEndUserRole>` element has an SPM of 10.

Data Type: The `<intendedEndUserRole>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`teacher`**
- **`author`**
- **`learner`**
- **`manager`**

Example:

```
<lom>
   <educational>
      <intendedEndUserRole>
         <source>LOMv1.0</source>
         <value>learner</value>
      </intendedEndUserRole>
   </educational>
</lom>
```

Code Illustration 4-48: <intendedEndUserRole> Element

4.2.6.6. <context> Element

The `<context>` element represents the principal environment within which the learning and use of the SCORM Content Model Component is intended to take place.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<context>`

Multiplicity Requirements: The `<context>` element shall appear 0 or More times. The `<context>` element has an SPM of 10.

Data Type: The `<context>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`school`**

- **higher education**
- **training**
- **other**

Example:

```
<lom>
   <educational>
      <context>
         <source>LOMv1.0</source>
         <value>training</value>
      </context>
   </educational>
</lom>
```

Code Illustration 4-49: <context> Element

4.2.6.7. <typicalAgeRange> Element

The `<typicalAgeRange>` element represents the age of the typical end user. This element shall refer to the developmental age, if that would be different from the chronological age. The IEEE Standard recommends that, when applicable, the value should be formatted as *minimum age – maximum age* or *minimum age* – (e.g., 18 – 25, or 18-). The values of this element do not necessarily have to be represented numerically (e.g., adults only, suitable for children over 7).

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<typicalAgeRange>`

Multiplicity Requirements: The `<typicalAgeRange>` element shall appear 0 or More times. The `<typicalAgeRange>` element has an SPM of 5.

Data Type: The `<typicalAgeRange>` element is represented as a LangString. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <educational>
      <typicalAgeRange>
         <string language="en">18-</string>
      </typicalAgeRange>
   </educational>
</lom>
```

Code Illustration 4-50: <typicalAgeRange> Element

4.2.6.8. <difficulty> Element

The `<difficulty>` element represents how hard it is to work with or through the SCORM Content Model Component for the typical intended target audience. The typical target audience can be characterized by the `<context>` and `<typicalAgeRange>` elements.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<difficulty>`

Multiplicity Requirements: The `<difficulty>` element shall appear 0 or 1 time.

Data Type: The `<difficulty>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`very easy`**
- **`easy`**
- **`medium`**
- **`difficult`**
- **`very difficult`**

These values, inherently, are meaningful only in a context of a community practice. At this time, this scale is left to organizations to define.

Example:

```
<lom>
   <educational>
      <difficulty>
         <source>LOMv1.0</source>
         <value>easy</value>
      </difficulty>
   </educational>
</lom>
```

Code Illustration 4-51: <difficulty> Element

4.2.6.9. <typicalLearningTime> Element

The `<typicalLearningTime>` element represents the approximate of typical time it takes to work with or through the SCORM Content Model Component for the typical intended target audience. The typical target audience can be characterized by the elements `<context>` and `<typicalAgeRange>`.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<typicalLearningTime>`

Multiplicity Requirements: The `<typicalLearningTime>` element shall appear 0 or 1 time.

Data Type: The `<typicalLearningTime>` element is represented as a Duration data type (refer to Section 4.2.11.5: *Duration Data Type* for more information).

Example:

```
<lom>
   <educational>
      <typicalLearningTime>
         <duration>PT1H30M</duration>
         <description>
            <string language="en">Average length of time to experience the
activity.</string>
         </description>
      </typicalLearningTime>
   </educational>
</lom>
```

Code Illustration 4-52: <typicalLearningTime> Element

4.2.6.10. <description> Element

The `<description>` element shall be used to comment on how the SCORM Content Model Component is to be used.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<description>`

Multiplicity Requirements: The `<description>` element shall appear 0 or More times. The `<description>` element has an SPM of 10.

Data Type: The `<description>` element is represented as a LangString. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <educational>
      <description>
         <string language="en">This course is designed for IT professionals
responsible for implementing Java</string>
      </description>
   </educational>
</lom>
```

Code Illustration 4-53: Educational <description> Element

4.2.6.11. <language> Element

The `<language>` element represents the human language used by the typical intended user of the SCORM Content Model Component. The typical target intended user can be characterized by the elements `<context>` and `<typicalAgeRange>`.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<language>`

Multiplicity Requirements: The `<language>` element shall appear 0 or More times. The `<language>` element has an SPM of 10.

Data Type: The `<language>` element is represented as a CharacterString. The CharacterString has an SPM of 100 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <educational>
      <language>en-US</language>
   </educational>
</lom>
```

Code Illustration 4-54: Educational <language> Element

4.2.7. <rights> Element

The Rights category describes the intellectual property rights and conditions of use for the SCORM Content Model Component. This element shall be used to describe any and all digital rights of the SCORM Component (cost for use, copyright, etc.).

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<rights>`

Multiplicity Requirements: The `<rights>` element shall appear 0 or 1 time.

Data Type: The `<rights>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<rights>` element contains the following child elements:

- `<cost>`
- `<copyrightAndOtherRestrictions>`
- `<description>`

Example:

```
<lom>
   <rights>
      <cost>
         <source>LOMv1.0</source>
         <value>yes</value>
      </cost>
      <copyrightAndOtherRestrictions>
         <source>LOMv1.0</source>
         <value>yes</value>
      </copyrightAndOtherRestrictions>
      <description>
         <string language="en">For additional information or questions
regarding copyright, distribution and reproduction, contact Joe Developer at
joe_developer@someorganization.org</string>
      </description>
   </rights>
</lom>
```

Code Illustration 4-55: <rights> Element

4.2.7.1. <cost> Element

The `<cost>` element represents whether the SCORM Content Model Component requires some sort of payment.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<cost>`

Multiplicity Requirements: The `<cost>` element shall appear 0 or 1 time.

Data Type: The `<cost>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`yes`**
- **`no`**

If the `<cost>` element is set to `yes`, the `<description>` element can be used to describe addition details dealing with the cost.

Example:

```
<lom>
   <rights>
      <cost>
         <source>LOMv1.0</source>
         <value>yes</value>
      </cost>
      <description>
         <string language="en">Contact joe_developer@someorg.org for cost 
information.</string>
      </description>
   </rights>
</lom>
```

Code Illustration 4-56: <cost> Element

4.2.7.2. <copyrightAndOtherRestrictions> Element

The `<copyrightAndOtherRestrictions>` element describes whether copyright or other restrictions apply to the use of the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<copyrightAndOtherRestrictions>`

Multiplicity Requirements: The `<copyrightAndOtherRestrictions>` element shall appear 0 or 1 time.

Data Type: The `<copyrightAndOtherRestrictions>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

- **`yes`**
- **`no`**

If the `<copyrightAndOtherRestrictions>` element is set to `yes`, the `<description>` element can be used to describe addition details dealing with the copyright and other restrictions.

Example:

```
<lom>
   <rights>
      <copyrightAndOtherRestrictions>
         <source>LOMv1.0</source>
         <value>yes</value>
      </copyrightAndOtherRestrictions>
      <description>
         <string language="en">Contact joe_developer@someorg.org for copyright
information.</string>
      </description>
   </rights>
</lom>
```

Code Illustration 4-57: <copyrightAndOtherRestrictions> Element

4.2.7.3. <description> Element

The `<description>` element allows for comments on the conditions of use of the SCORM Content Model Component. This element can be used to describe any cost, copyright or other restrictions about the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<description>`

Multiplicity Requirements: The `<description>` element shall appear 0 or 1 time.

Data Type: The `<description>` element is represented as a LangString. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <rights>
      <copyrightAndOtherRestrictions>
         <source>LOMv1.0</source>
         <value>yes</value>
      </copyrightAndOtherRestrictions>
      <description>
         <string language="en">For additional information or questions
regarding copyright, distribution and reproduction, contact Joe Developer at
joe_developer@someorganization.org</string>
      </description>
   </rights>
</lom>
```

Code Illustration 4-58: Rights <description> Element

4.2.8. <relation> Element

The Relation category defines the relationship between the SCORM Content Model Component and other components, if any. The Relation element is allowed to be repeated. To define multiple relationships, one could create several instances of this category. If there is more than one target component, then each target shall have a new relationship instance.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<relation>`

Multiplicity Requirements: The `<relation>` element shall appear 0 or More times. The `<relation>` element has an SPM of 100.

Data Type: The `<relation>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<relation>` element contains the following child elements:

- `<kind>`
- `<resource>`

Example:

```
<lom>
   <relation>
      <kind>
         <source>LOMv1.0</source>
         <value>isbasedon</value>
      </kind>
      <resource>
         <identifier>
            <catalog>URN</catalog>
            <entry>urn:ADL:1234-45FD</entry>
         </identifier>
         <description>
            <string language="en">Microsoft MSCE</string>
         </description>
      </resource>
   </relation>
</lom>
```

Code Illustration 4-59: <relation> Element

4.2.8.1. <kind> Element

The `<kind>` element describes the nature of the relationship between the SCORM Content Model Component and the target component identified by the Resource.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<kind>`

Multiplicity Requirements: The `<kind>` element shall appear 0 or 1 time.

Data Type: The `<kind>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

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

Example:

```
<lom>
   <relation>
      <kind>
         <source>LOMv1.0</source>
         <value>ispartof</value>
      </kind>
      <resource>
         <identifier>
            <catalog>URN</catalog>
            <entry>urn:ADL:1234-45FD-3324</entry>
         </identifier>
         <description>
            <string language="en">ADL Course: Microsoft MSCE</string>
         </description>
      </resource>
   </relation>
</lom>
```

Code Illustration 4-60: <kind> Element

4.2.8.2. <resource> Element

The `<resource>` element describes the target SCORM Content Model Component that this relationship references.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<resource>`

Multiplicity Requirements: The `<resource>` element shall appear 0 or 1 time.

Data Type: The `<resource>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<resource>` element contains the following child elements:

- `<identifier>`
- `<description>`

Example:

```
<lom>
   <relation>
      <kind>
         <source>LOMv1.0</source>
         <value>isbasedon</value>
      </kind>
      <resource>
         <identifier>
            <catalog>URN</catalog>
            <entry>urn:ADL:1234-45FD</entry>
         </identifier>
         <description>
            <string language="en">ADL Course: Microsoft MSCE</string>
         </description>
      </resource>
   </relation>
</lom>
```

Code Illustration 4-61: <resource> Element

4.2.8.2.1. <identifier> Element

The `<identifier>` element represents a mechanism for assigning a globally unique label that identifies the target SCORM Content Model Component

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<identifier>`

Multiplicity Requirements: The `<identifier>` element shall appear 0 or More times. The `<identifier>` element has an SPM of 10.

Data Type: The `<identifier>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<identifier>` element contains the following child elements:

- `<catalog>`
- `<entry>`

Example:

```
<lom>
   <relation>
      <kind>
         <source>LOMv1.0</source>
         <value>isbasedon</value>
      </kind>
      <resource>
         <identifier>
            <catalog>URN</catalog>
            <entry>urn:ADL:1234-45FD</entry>
         </identifier>
         <description>
            <string language="en">ADL Course: Microsoft MSCE</string>
         </description>
      </resource>
   </relation>
</lom>
```

Code Illustration 4-62: Relation Resource <identifier> Element

4.2.8.2.1.1. <catalog> Element

The `<catalog>` element represents the name or designator of the identification or cataloging scheme for the entry. There are a variety of cataloging systems available. SCORM does not require the use of any one particular cataloging system. Organizations are free to choose any cataloging scheme that meets their organizations practices or policies. Some types of cataloging systems are:

- Universal Resource Identifier (URI)
- Universal Resource Name (URN)
- Digital Object Identifier (DOI)
- International Standard Book Numbers (ISBN)
- International Standard Serial Numbers (ISSN).

The `<catalog>` element represents the scheme used to create and manage the entry.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<catalog>`

Multiplicity Requirements: The `<catalog>` element shall appear 0 or 1 time.

Data Type: The `<catalog>` element is represented as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <relation>
      <kind>
         <source>LOMv1.0</source>
         <value>isbasedon</value>
      </kind>
      <resource>
         <identifier>
            <catalog>URN</catalog>
            <entry>urn:ADL:1234-45FD</entry>
         </identifier>
         <description>
            <string language="en">ADL Course: Microsoft MSCE</string>
         </description>
      </resource>
   </relation>
</lom>
```

Code Illustration 4-63: Relation Resource Identifier <catalog> Element

4.2.8.2.1.2. <entry> Element

The `<entry>` element represents the value of the identifier within the identification or cataloging scheme (refer to the `<catalog>` element) that designates or identifies the target SCORM Content Model Component.

Identifiers can take on various formats. The IEEE requires that the actual identifier value be represented as a CharacterString. Organizations are free to choose any mechanism to create unique identifiers that meets their organizations practices or policies. It is recommended that a common scheme be chosen.

The following listing is a sampling of identifier values (entry):

Scheme (`<catalog>`)	Value (`<entry>`)
Universal Resource Name	urn:ADL: 1345-GFGC-23ED-3321
Universal Resource Identifier	http://www.adlnet.gov/content/C0_01
Handle Syntax	100.100/2345342256349543

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<entry>`

Multiplicity Requirements: The `<entry>` element shall appear 0 or 1 time.

Data Type: The `<entry>` element is represented as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <relation>
      <kind>
         <source>LOMv1.0</source>
         <value>isbasedon</value>
      </kind>
      <resource>
         <identifier>
            <catalog>URN</catalog>
            <entry>urn:ADL:1234-45FD</entry>
         </identifier>
         <description>
            <string language="en">ADL Course: Microsoft MSCE</string>
         </description>
      </resource>
   </relation>
</lom>
```

Code Illustration 4-64: Relation Resource Identifier <entry> Element

4.2.8.2.2. <description> Element

The `<description>` element describes the target SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<description>`

Multiplicity Requirements: The `<description>` element shall appear 0 or More times. The `<description>` element has an SPM of 10.

Data Type: The `<description>` element is represented as a LangString. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <relation>
      <kind>
         <source>LOMv1.0</source>
         <value>isbasedon</value>
      </kind>
      <resource>
         <identifier>
            <catalog>URN</catalog>
            <entry>urn:ADL:1234-45FD</entry>
         </identifier>
         <description>
            <string language="en">ADL Course: Microsoft MSCE</string>
         </description>
      </resource>
   </relation>
</lom>
```

Code Illustration 4-65: Relation Resource <description> Element

4.2.9. <annotation> Element

The Annotation category provides comments on the educational use of the SCORM Content Model Component and information on when and by whom the comments were created. This category enables educators to share their assessments of SCORM Content Model Components, suggestions for use, etc.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<annotation>`

Multiplicity Requirements: The `<annotation>` element shall appear 0 or More times. The `<annotation>` element has an SPM of 30.

Data Type: The `<annotation>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<annotation>` element contains the following child elements:

- `<entity>`
- `<date>`
- `<description>`

Example:

```
<lom>
   <annotation>
      <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Author&#13;&#10;END:VCARD</entity>
      <date>
         <dateTime>2001-07-30T10:14:35.5+01:00</dateTime>
         <description>
            <string language="en">Date and time annotation was created</string>
         </description>
      </date>
      <description>Learners will need to understand the fundamentals of Windows
programming in order to grasp the concepts described in this
learning.</description>
   </annotation>
</lom>
```

Code Illustration 4-66: <annotation> Element

4.2.9.1. <entity> Element

The `<entity>` element identifies the entity or entities that that have created the annotation.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<entity>`

Multiplicity Requirements: The `<entity>` element shall appear 0 or 1 time.

Data Type: The `<entity>` element as a CharacterString element. The CharacterString has an SPM of 1000 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information). All entity values shall be represented in vCard [9] format. This allows systems to take the CharacterString represented by the `<entity>` element and process this CharacterString as a valid vCard.

Example:

```
<lom>
   <annotation>
      <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Author&#13;&#10;END:VCARD</entity>
      <date>
         <dateTime>2001-07-30T10:14:35.5+01:00</dateTime>
         <description>
            <string language="en">Date and time annotation was created</string>
         </description>
      </date>
      <description>Learners will need to understand the fundamentals of Windows
programming in order to grasp the concepts described in this
learning.</description>
   </annotation>
</lom>
```

Code Illustration 4-67: Annotation <entity> Element

4.2.9.2. <date> Element

The `<date>` element identifies the date the annotation was created.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<date>`

Multiplicity Requirements: The `<date>` element shall appear 0 or 1 time.

Data Type: The Date element is represented as a DateTime data type (refer to Section 4.2.11.4: *DateTime Data Type* for more information). The `<date>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<date>` element contains two elements, one that represents the actual date of the contribution (`<dateTime>`) and one that represents a textual description of the date (`<description>`):

- `<dateTime>`
- `<description>`

Example:

```
<lom>
   <annotation>
      <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Author&#13;&#10;END:VCARD</entity>
      <date>
         <dateTime>2001-07-30T10:14:35.5+01:00</dateTime>
         <description>
            <string language="en">Date and time annotation was created</string>
         </description>
      </date>
      <description>Learners will need to understand the fundamentals of Windows
programming in order to grasp the concepts described in this
learning.</description>
   </annotation>
</lom>
```

Code Illustration 4-68: Annotation <date> Element

4.2.9.3. <description> Element

The `<description>` element shall be used to represent the contents of the annotation.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<description>`

Multiplicity Requirements: The `<description>` element shall appear 0 or 1 time.

Data Type: The `<description>` element is represented as a LangString. The LangString has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <annotation>
      <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe
Author&#13;&#10;END:VCARD</entity>
      <date>
         <dateTime>2001-07-30T10:14:35.5+01:00</dateTime>
         <description>
            <string language="en">Date and time annotation was created</string>
         </description>
      </date>
      <description>
          <string>Learners will need to understand the fundamentals of Windows
programming in order to grasp the concepts described in this learning.</string>
       </description>
   </annotation>
</lom>
```

Code Illustration 4-69: Annotation <description> Element

4.2.10. <classification> Element

The Classification category describes where the SCORM Content Model Component falls within a particular classification system. Multiple Classification categories may be used to define multiple classifications. The Classification category is typically used to link to a controlled vocabulary or classification system.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<classification>`

Multiplicity Requirements: The `<classification>` element shall appear 0 or More times. The `<classification>` element has an SPM 40.

Data Type: The `<classification>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<classification>` element contains the following child elements:

- `<purpose>`
- `<taxonPath>`
- `<description>`
- `<keyword>`

Example:

```
<lom>
   <classification>
      <purpose>
         <source>LOMv1.0</source>
         <value>skill level</value>
      </purpose>
      <taxonPath>
         <source>
            <string language="en-US">ADL SCORM Concepts</string>
         </source>
         <taxon>
            <id>I</id>
            <entry>
               <string language="en-US">Content Aggregation Model</string>
            </entry>
         </taxon>
         <taxon>
            <id>I.A</id>
            <entry>
               <string language="en-US">Content Packaging Fundamentals</string>
            </entry>
         </taxon>
         <taxon>
            <id>I.A.3</id>
            <entry>
               <string language="en-US">Resource Fundamentals</string>
            </entry>
         </taxon>
         <taxon>
            <id>I.A.3.a</id>
            <entry>
               <string language="en-US">Packaging SCOs</string>
            </entry>
         </taxon>
      </taxonPath>
      <description>
         <string language="en-US">Describing and packaging SCOs in a SCORM 
Content Package</string>
      </description>
      <keyword>
         <string language="en-US">Packaging SCOs</string>
      </keyword>
   </classification>
</lom>
```

Code Illustration 4-70: <classification> Element

4.2.10.1. <purpose> Element

The `<purpose>` element defines the purpose for classifying the SCORM Content Model Component.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<purpose>`

Multiplicity Requirements: The `<purpose>` element shall appear 0 or 1 time.

Data Type: The `<purpose>` element is represented as a Vocabulary element (refer to Section 4.2.11.3: *Vocabulary Data Type* for more information).

Vocabulary Tokens: The IEEE LOM defines the following set of vocabulary tokens:

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

Example:

```
<lom>
   <classification>
      <purpose>
         <source>LOMv1.0</source>
         <value>skill level</value>
      </purpose>
   </classification>
</lom>
```

Code Illustration 4-71: <purpose> Element

4.2.10.2. <taxonPath> Element

The `<taxonPath>` element describes a taxonomic path in a specific classification system. Each succeeding level is a refinement in the definition of the proceeding level.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<taxonPath>`

Multiplicity Requirements: The `<taxonPath>` element shall appear 0 or More times. The `<taxonPath>` element has an SPM of 15.

Data Type: The `<taxonPath>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<taxonPath>` element contains the following child elements:

- `<source>`
- `<taxon>`

Example:

```
<lom>
   <classification>
      <purpose>
         <source>LOMv1.0</source>
         <value>skill level</value>
      </purpose>
      <taxonPath>
         <source>
            <string language="en-US">ADL SCORM Concepts</string>
         </source>
         <taxon>
            <id>I</id>
            <entry>
               <string language="en-US">Content Aggregation Model</string>
            </entry>
         </taxon>
         <taxon>
            <id>I.A</id>
            <entry>
               <string language="en-US">Content Packaging Fundamentals</string>
            </entry>
         </taxon>
         <taxon>
            <id>I.A.3</id>
            <entry>
               <string language="en-US">Resource Fundamentals</string>
            </entry>
         </taxon>
         <taxon>
            <id>I.A.3.a</id>
            <entry>
               <string language="en-US">Packaging SCOs</string>
            </entry>
         </taxon>
      </taxonPath>
   </classification>
</lom>
```

Code Illustration 4-72: <taxonPath> Element

4.2.10.2.1. <source> Element

The `<source>` element describes or names the classification system. This data element may use any recognized official taxonomy or any user-defined taxonomy.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<source>`

Multiplicity Requirements: The `<source>` element shall appear 0 or 1 time.

Data Type: The `<source>` element is represented as a LangString. The LangString element has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <classification>
      <purpose>
         <source>LOMv1.0</source>
         <value>skill level</value>
      </purpose>
      <taxonPath>
         <source>
            <string language="en-US">ADL SCORM Concepts</string>
         </source>
      </taxonPath>
   </classification>
</lom>
```

Code Illustration 4-73: Taxon Path <source> Element

4.2.10.2.2. <taxon> Element

The `<taxon>` element describes 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.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<taxon>`

Multiplicity Requirements: The `<taxon>` element shall appear 0 or More times. The `<taxon>` element has an SPM of 15.

Data Type: The `<taxon>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements. The `<taxonPath>` element contains the following child elements:

- `<id>`
- `<entry>`

Example:

```
<lom>
   <classification>
      <purpose>
         <source>LOMv1.0</source>
         <value>skill level</value>
      </purpose>
      <taxonPath>
         <source>
            <string language="en-US">ADL SCORM Concepts</string>
         </source>
         <taxon>
            <id>I</id>
            <entry>
               <string language="en-US">Content Aggregation Model</string>
            </entry>
         </taxon>
      </taxonPath>
   </classification>
</lom>
```

Code Illustration 4-74: <taxon> Element

4.2.10.2.2.1. <id> Element

The `<id>` element describes the identifier of the taxon. For example, the element can be a number or letter combination provided by the source of the taxonomy.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<id>`

Multiplicity Requirements: The `<id>` element shall appear 0 or 1 time.

Data Type: The `<id>` element is represented as a CharacterString element. The CharacterString has an SPM of 100 characters (refer to Section 4.2.11.1: *CharacterString Data Type* for more information).

Example:

```
<lom>
   <classification>
      <taxonPath>
         <source>
            <string language="en-US">ADL SCORM Concepts</string>
         </source>
         <taxon>
            <id>I</id>
            <entry>
               <string language="en-US">Content Aggregation Model</string>
            </entry>
         </taxon>
      </taxonPath>
   </classification>
</lom>
```

Code Illustration 4-75: <id> Element

4.2.10.2.2.2. <entry> Element

The <entry> element shall contain a textual label of the taxon.

XML Namespace: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <entry>

Multiplicity Requirements: The <entry> element shall appear 0 or 1 time.

Data Type: The <entry> element is represented as a LangString. The LangString element has an SPM of 500 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <classification>
      <purpose>
         <source>LOMv1.0</source>
         <value>skill level</value>
      </purpose>
      <taxonPath>
         <source>
            <string language="en-US">ADL SCORM Concepts</string>
         </source>
         <taxon>
            <id>I</id>
            <entry>
               <string language="en-US">Content Aggregation Model</string>
            </entry>
         </taxon>
      </taxonPath>
   </classification>
</lom>
```

Code Illustration 4-76: Taxon <entry> Element

4.2.10.3. <description> Element

The <description> element contains a description of the SCORM Content Model Component relative to the stated Purpose (<purpose>) of the specific classification, such as discipline, idea, skill level, educational objective, etc.

XML Namespace: http://ltsc.ieee.org/xsd/LOM

XML Binding Representation: <description>

Multiplicity Requirements: The <description> element shall appear 0 or 1 time.

Data Type: The <description> element is represented as a LangString. The LangString element has an SPM of 2000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <classification>
      <description>
         <string language="en-US">Describing and packaging SCOs in a SCORM 
Content Package</string>
      </description>
   </classification>
</lom>
```

Code Illustration 4-77: Classification <description> Element

4.2.10.4. <keyword>

The `<keyword>` element contains keywords and phrases descriptive of the SCORM Content Model Component relative to the stated Purpose (`<purpose>`) of this specific classification, such as discipline, idea, skill level, educational objective, etc.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Binding Representation: `<keyword>`

Multiplicity Requirements: The `<keyword>` element shall appear 0 or More times. The `<keyword>` element has an SPM of 40.

Data Type: The `<keyword>` element is represented as a LangString. The LangString element has an SPM of 1000 characters (refer to Section 4.2.11.2: *LangString Data Type* for more information).

Example:

```
<lom>
   <classification>
      <keyword>
         <string language="en-US">Packaging SCOs</string>
      </keyword>
   </classification>
</lom>
```

Code Illustration 4-78: Classification <keyword> Element

4.2.11. Common Data Types

The IEEE LOM contains several common data types. These data types are used to describe the makeup of the values held by the individual LOM elements. The following sections define the LOM data types and their characteristics.

4.2.11.1. CharacterString Data Type

The CharacterString is a data type used to capture a set of characters that are not interpreted in a language. The characters represented by this data type are those characters supported by ISO/IEC 10646-1:2000 [10]. The ISO 10646 standard provides a unified character coding standard for the communication and exchange of electronic information.

4.2.11.2. LangString Data Type

The LangString data type represents one or more characterstrings in which the characterstring's language is identified. A LangString value may include multiple semantically equivalent characterstrings, such as translations or alternative descriptions.

XML Namespace: `http://ltsc.ieee.org/xsd/LOM`

XML Namespace Prefix: `lom`

XML Binding Representation:

```
<string language="language-code">Textual characterstring represented
in the defined language</string>
```

Code Illustration 4-79: LangString language Attribute

The `<string>` element shall contain the actual phrase in a human language. The SPM length of the `<string>` element's value is determined by the parent element refer to the LangString elements defined in Section 1.2: *IEEE 1484.12.1-2002 Learning Object Metadata*).

Attribute:

- `language`- represents the human language of the contents of the `<string>` element. The language attribute is represented as a CharacterString with an SPM of 100 characters. The language attribute is optional. The `<language>` element of the `<metaMetadata>` element represents the default language of all LangStrings values. If the language attribute is not present in the individual `<string>` elements, the value held by the string shall be represented in the language defined by the `<language>` element of the `<metaMetadata>` element. The language attribute is important because without it language information is lost. The value space of the characterstring is from ISO-10646-1. This standard only consists of character codes for each independent characters. The standard consists of codes for characters that come from different languages. Since the several languages share the same characters which are represented by the same character codes, the language information is important.

Multiplicity: The `<string>` element shall occur 0 or More times within its parent element (with an SPM of 10 `<string>` elements).

Example:

```
<general>
   <keyword>
      <string language="en">metadata</string>
      <string language="fr"> métadonnées</string>
   </keyword>
</general>
```

Code Illustration 4-80: LangString Data Type

4.2.11.3. Vocabulary Data Type

There are certain elements that have a Vocabulary data type. A vocabulary is a recommended list of appropriate values. The vocabulary data type is represented as a source/value pair. This indicates that for each vocabulary there is a source (or owner) and then a value (actual vocabulary token). For those vocabularies defined by the LOM, the source is required to be "`LOMv1.0`".

Data Type: The Vocabulary Data Type is an aggregate data type made up of two elements:

- `<source>`: An indication of the source, or owner, of the vocabulary values. For those elements that require the use of LOM vocabularies, the `<source>` element shall have a value of "`LOMv1.0`". For those elements that are not mandated to use a LOM vocabulary, the `<source>` value may be set to any implementation defined CharacterString. The CharacterString shall have an SPM of 1000 characters.
- `<value>`: The actual value defined by the source. If the `<source>` is LOMv1.0, the value shall come from the list defined in the LOM. For those elements that are not mandated to use a LOM vocabulary, the `<value>` shall be defined by the `<source>`. The value of the `<value>` element shall have an SPM of 1000 characters.

Multiplicity: The `<source>` and `<value>` elements shall occur 1 and only 1 time within those parent elements that are of a Vocabulary Data Type.

Example:

```
<lom>
<!--LOM Vocabulary example-->
   <rights>
      <cost>
         <source>LOMv1.0</source>
         <value>yes</value>
      </cost>
   </rights>
<!-- Vocabulary Extension Example -->
   <educational>
      <learningResourceType>
         <source>Organization XXX</source>
         <value>simple questionnaire</value>
      </learningResourceType>
   </educational>
</lom>
```

Code Illustration 4-81: Vocabulary Data Type

4.2.11.4. DateTime Data Type

The DateTime data type is used to describe a point in time with at least an accuracy as small as one second.

Data Type: The DateTime data type is an aggregate data type made up of two elements:

- `<dateTime>`: CharacterString representation of the point in time. The CharacterString shall have an SPM of 200 characters.
- `<description>`: Represents a description of the point in time. The `<description>` element is a LangString data type (refer to Section 4.2.11.2: *LangString Data Type* for more information). The LangString shall have an SPM of 1000 characters.

Format: The format of the `<dateTime>` element shall be represented in accordance with ISO8601:2000:

`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>`.

Multiplicity: The `<dateTime>` and `<description>` element occurs 0 or 1 time within its parent element.

Example:

```
<lom>
   <lifeCycle>
      <contribute>
         <role>
            <source>LOMv1.0</source>
            <value>author</value>
         </role>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe Author&#13;&#10;END:VCARD</entity>
         <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Mary Author&#13;&#10;END:VCARD</entity>
         <date>
            <dateTime>2002-12-12</dateTime>
            <description>
               <string language="en">A description for the date</string>
            </description>
         </date>
      </contribute>
   </lifeCycle>
</lom>
```

Code Illustration 4-82: DataTime Data Type

4.2.11.5. Duration Data Type

The Duration data type is used to describe an interval in time with accuracy at least as small as one second.

Data Type: The Duration data type is an aggregate data type made up of two elements:

- `<duration>`: CharacterString representation of the time interval. The CharacterString shall have an SPM of 200 characters.
- `<description>`: Represents a description of the time interval. The `<description>` element is a LangString data type (refer to Section 4.2.11.2: *LangString Data Type* for more information). The LangString shall have an SPM of 1000 characters.

Format: The format of the `<duration>` element shall be represented in accordance with ISO8601:2000:

`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.

Multiplicity: The `<duration>` and `<description>` element occurs 0 or 1 time within its parent element.

Example:

```
<lom>
   <educational>
      <typicalLearningTime>
         <duration>PT1H30M</duration>
         <description>
            <string language="en">It takes this long to complete the 
course</string>
         </description>
      </typicalLearningTime>
   </educational>
</lom>
```

Code Illustration 4-83: Duration Data Type

4.2.11.6. VCard Data Type

The vCard data type is used to describe an entity (individual or organization). vCard automates the exchange of personal information typically found on a traditional business card. The vCard specification [9] defines a "virtual" electronic business card. vCards can store information such as a name, address, telephone number, e-mail address, etc.

The following elements shall be in valid vCard format:

- LifeCycle.Contribute.Entity
- Meta-Metadata.Contribute.Entity
- Annotation.Entity

Example:

```
<annotation>
   <entity>BEGIN:VCARD&#13;&#10;VERSION:2.1&#13;&#10;FN:Joe 
Friday&#13;&#10;TITLE:Area 
Adminstrator\,Assistant&#13;&#10;EMAIL\;TYPE=INTERN&#13;&#10;ET:jfriday@host.co
m&#13;&#10;END:VCARD</entity>
</annotation>
```

Code Illustration 4-84: VCard Data Type

4.3. LOM XML Schema Validation Approaches

The LOM XML Binding is fundamentally a collection of rules describing how to create metadata instances in XML. The XSD files are used to describe and enforce these rules. Sometimes there are certain rules that cannot be expressed in an XML Schema Definition. In these cases, these rules are described by the normative text found in the IEEE standard. The IEEE provides several validation approaches for the LOM XML Binding. These validation approaches were built to provide several alternative XML schemas. Each of the approaches created support and enforce a different set of binding rules. In all cases, the XSD files are not sufficient to validate LOM Metadata instances. LOM Metadata instances shall be conformant to the LOM XML Binding in all cases where:

```
    LOM XML Binding = "XML Schema Definition" + "Normative Standard"
```

The IEEE provides a set of driver schemas that support the different validation approaches. These approaches can be used in the validation of the LOM metadata instances. However, there may be additional validation steps needed based on the approach being used. The following sections describe the different validation approaches, schema drivers and their relationship to SCORM.

4.3.1. Strict Schema Validation Approach

The main goal of the strict schema validation approach is to enforce the strict requirements as defined by the LOM. The strict schema validation approach and its set of corresponding XSD files have the following characteristics:

- Supports uniqueness constraints. For those elements that are defined in the LOM as having a multiplicity requirement of 0 or 1, the strict schema validation approach enforces this constraint.
- LOM Vocabulary only. The strict schema validation approach only permits LOM metadata instances that use the LOM-defined vocabulary tokens.
- No extensions. The strict schema validation approach does not support extensions to the LOM.

Because this approach does not support extensions to LOM and the use of LOM vocabularies, the approach permits valid strictly conforming LOM metadata instances (as defined by IEEE).

If an organization defines policies not to extend the base elements defined by the LOM and enforce the use of the vocabularies defined by the LOM, then ADL recommends the use of the strict schema validation approach. It will ensure the most semantic interoperability of all of the validation approaches.

4.3.2. Custom Schema Validation Approach

The main goal of the custom schema validation approach is to support the ability to customize LOM metadata instances to support extensions to both vocabularies and data elements. The custom schema validation approach (and its set of corresponding XSD files) contains the following characteristics:

- Support uniqueness constraints. For those elements that are defined in the LOM as having a multiplicity requirement of 0 or 1, the custom schema validation approach enforces this constraint.
- Custom Vocabularies. This approach allows for the use of LOM defined vocabularies or the use of an organizations vocabularies. The validation approach also provides a "template" to assist in building an XSD file for use by tools. These XSD files could be used in support of validation of the LOM Metadata instance.
- Supports extensions to the LOM. This schema validation approach supports the ability to extend the data model set defined by LOM. This allows an organization to incorporate a different set (from another namespace) of elements in a LOM metadata instance.

Because of the fact that this approach supports extensions to LOM, the approach permits valid conforming LOM metadata instances (as defined by IEEE).

If an organization contains policies or practices that require a different set of vocabularies or a set of elements that shall be included in metadata, then ADL recommends the use of the custom schema validation approach. However, keep in mind that this will not allow semantic interoperability of metadata instances between different organizations. To keep a high degree of semantic interoperability, ADL recommends that vertical communities (e.g., health care industry) work to a consensus on building a set of interoperable vocabularies that can be applied to the custom validation approach.

For those elements in the LOM that are of type Vocabulary, SCORM only requires that certain elements use those vocabularies. For some of the elements, SCORM only makes it a recommended best practice to use the defined LOM vocabularies. If an organization needs to define a different set of vocabularies for SCORM best practice vocabulary elements, then ADL recommends the use of the custom schema validation approach and the guidance given for building an XSD file. This will permit tools to properly validate the metadata instances.

4.3.3. Loose Schema Validation Approach

The main goal of the loose schema validation approach is to relax some of the constraints defined by the other schema validation approaches. When using the strict or custom validation approach, an artificial attribute is introduced to help tools with the validation of the uniqueness constraints of the LOM. If an organization wants to avoid the introduction of this artificial element, then the loose schema validation approach can be used. Keep in mind that the loose schema validation approach (and corresponding loose schema driver) does not check uniqueness constraints and will permit non-conformant LOM metadata instances. Extra steps must to be taken for tools to enforce the overall LOM XML Binding rules.

The loose schema validation approach (and its set of corresponding XML Schema Definition files) contains the following characteristics:

- Does not support uniqueness constraints. The loose schema validation approach relaxes the uniqueness constraint checks by removing the introduction of the artificial attribute. Because of this, there are cases where non-conforming LOM metadata instances will validate against the loose schema validation approach. It is recommended that producers of LOM metadata instances guarantee that the instance produced is valid according to the IEEE LOM XML Binding when using the loose schema validation approach.
- No validation of vocabularies. The loose schema validation approach relaxes the schema enforcement of vocabulary source and value pair relationship constraints. The loose schema validation approach simplifies the schema validation process. However, the absence of the enforcement does not guarantee conforming LOM metadata instances. Applications will have to validate the vocabulary source and value pair relationship constraints by other means.
- Supports extensions to the LOM. The validation approach supports the ability to extend the data model set defined by LOM. This allows an organization to incorporate a different set (from another namespace) of elements in a LOM metadata instance.

The loose schema validation approach requires more processing, outside of validation tools, to verify that the LOM metadata instance is conforming to the requirements of IEEE. ADL recommends that one of the other schema validation approaches be used in lieu of the loose schema validation approach.

4.4. Metadata Extensions

In some cases, organizations may find that the core set of metadata elements defined by LOM is not adequate enough to describe SCORM Content Model Components. The organization may have a set of metadata extensions that are required to be used in describing these components. There are currently two types of extensions mechanism permitted within the LOM:

- XML element extensions. The first mechanism allows for the extension of the LOM data model elements. It is permissible to add additional elements to metadata instances. For example, if an organization has additional information regarding intellectual property rights and conditions or use for their SCORM Content Model Components, it is feasible for the organization to add elements to the Rights Category. There are currently several ongoing research and development activities dealing with digital rights management. It is feasible that a set of elements describing a more robust and rich sets of rights will be developed. This could ideally be used in describing SCORM Content Model Components by extending the Rights category.

- Vocabulary extensions. Some of the IEEE elements have a defined list of vocabulary values. However, this is just a recommendation and the metadata instances are not required to use those vocabularies. If an organization has the need to use a different set of vocabularies, then the organization has several alternatives. If the organization wants to enforce validation of the vocabularies, then ADL recommends the use of the custom schema validation approach (refer to Section 4.4.2: *Vocabulary Extension* for more information on building XML Schema Definition files for validation purposes). The strict schema validation approach cannot be used since it will only validate strict LOM vocabulary pairs.

Several words of caution when using extensions.

1. When creating extension elements, it is not permitted to define elements that contain the same semantics of the currently defined IEEE LOM elements.
2. Metadata that relies on the recommended values will have the highest degree of semantic interoperability (i.e., the likelihood that such metadata will be understood by other end users or systems is the highest). To keep a high degree of semantic interoperability, ADL recommends that if extensions are needed to meet the needs of a vertical community (e.g., health care industry), then the vertical communities should work to a consensus on building a set of interoperable extensions.

The LOM distinguishes between two types of conformance to the IEEE standard. If a LOM metadata instance does not contain any extensions, then the IEEE standard refers to this as a strictly conforming LOM metadata instance. If a LOM metadata instance contains extended data elements, then the IEEE standard refers to this as a conforming LOM metadata instance. SCORM supports both types of conformance and recommends LOM metadata instances to be strictly conforming (based on the cautions described above).

4.4.1. Data Element Extension

There may be situations where organizations have policies and practices for describing SCORM Content Model Components in ways the LOM does not support with its element set. For example, organizations may have a robust digital rights management scheme that is used to describe their learning content. The LOM permits its base scheme to be extended. As mentioned above, this has potential to decrease the semantic interoperability of the metadata and learning content.

If an organization wishes to provide its own extensions to the current LOM the following rules shall be adhered to:

- Extensions to the LOM base schema shall retain the value space and data type of data elements from the LOM base schema.

- Extensions shall not define data types or value spaces for aggregate data elements in the LOM base schema.
- Extended data elements should not replace data elements in the LOM base schema.

Element extensions are handled in a variety of ways based on the XML specification set. ADL recommends that if XML extensions are needed by an organization, then the organization shall provide an XSD file that can be used for XML Metadata instance validation.

4.4.2. Vocabulary Extension

The IEEE recommends the use of the vocabulary token set defined in the LOM. SCORM acknowledges this recommendation but further permits, if necessary, the creation and use of an organizations own vocabulary token set for those elements marked as Best Practice Vocabulary. As mentioned above, this has the potential to decrease the semantic interoperability of the metadata and learning content and should be used with caution. To keep a high degree of semantic interoperability, ADL recommends that vertical communities (e.g., health care industry) work to a consensus on building a set of interoperable vocabularies. If there is a need in a vertical industry to create new sets of vocabulary tokens, it may be appropriate to work with others in that vertical industry to create an agreed-upon vocabulary token set.

ADL recommends the use of the custom schema validation approach in order to represent the extended vocabularies. In order to create and use an organization-defined set of vocabulary tokens, the IEEE custom schema validation approach requires a similar XML binding as defined by the LOM XML Binding.

4.5. Metadata and SCORM Content Model Components

Thus far, the metadata sections of this document detailed the LOM information model, how this information model is bound to XML and ways to extend the LOM to potentially meet certain organizations policies or business needs. This section describes one way that organizations may choose to associate metadata to the SCORM Content Model Components (Content Aggregation, Content Organization, Activity, SCO and Asset). SCORM enables a way of associating metadata to the SCORM Content Model Components through the SCORM Content Package Manifest. Each of the SCORM Content Model Components are represented in the Manifest. The Manifest provides a means for associating the metadata (refer to Section 3.4: *Building Content Packages*) to these components.

- **Content Aggregation Metadata:** Content Aggregation Metadata describes the content aggregation (i.e., the content package) as a whole. The purpose of applying Content Aggregation Metadata is to enable discoverability of the Content Aggregation and to provide descriptive information about the Content Aggregation as a whole.
- **Content Organization Metadata:** Content Organization Metadata describes the Content Organization. The purpose of applying Content Organization Metadata is to enable discoverability within, for example, a content repository and to provide descriptive information about the content structure, as a whole, defined by the Content Organization.
- **Activity Metadata:** Activity Metadata describes an individual Activity. The purpose of applying Activity Metadata is to make the Activity accessible (enabling discovery) within a content repository. The metadata should describe the Activity as a whole.
- **SCO Metadata:** Metadata can be applied to SCOs to provide descriptive information about the content in the SCO independent of use. This metadata is used to facilitate reuse and discoverability of content.
- **Asset Metadata:** Metadata can be applied to Assets to provide descriptive information about the Assets independent of any usage or potential usage within courseware content. This metadata is used to facilitate reuse and discoverability, within, for example, a content repository during content creation.

As mentioned earlier defining metadata in a SCORM Content Package is not required by SCORM. Organizations or communities of practice are encouraged to identify their uses cases and requirements for using such a mechanism for associating metadata. Some systems may or may not support the metadata defined in the Content Package. For example, some LMSs may be built to look for metadata associated with a content package and use that metadata to populate an internal catalog of content. It is important to know that this type of behavior is not mandated by SCORM. It is important for content developers to understand the purpose of the metadata being added to the content package and how it may ultimately be used by systems. These use cases and requirements are not defined by SCORM.

4.5.1. Metadata Describing Content Aggregations

Content Aggregation level metadata shall be used to describe the package (i.e., Content Aggregation) as a whole. If metadata is provided to describe the Content Aggregation, then this metadata shall adhere to the requirements outlined earlier.

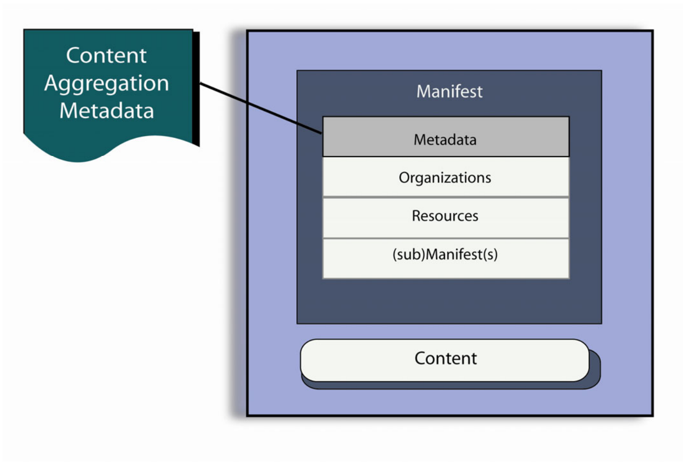


Figure 4.5.1a: Application of Metadata Describing a Content Aggregation

The following example illustrates the inclusion of metadata "inline" in the content package's manifest. This is one mechanism for applying metadata to a manifest. This mechanism can be used for the describing the rest of the SCORM Content Model Components within a manifest.

```
<manifest>
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
      <lom xmlns="http://ltsc.ieee.org/xsd/LOM">
         <general />
         <classification />
         <annotation />
         <lifeCycle />
         <technical />
         <metaMetadata />
         <educational />
         <relation />
         <rights />
      </lom>
   </metadata>
</manifest>
```

Code Illustration 4-85: Inline Namespace Metadata

The following example illustrates the use of the `<adlcp:location>` element to describe the metadata. The `<adlcp:location>` element describes the location of the metadata relative to the root of the package.

```
<manifest>
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
      <adlcp:location>contentAggregationMetadata.xml</adlcp:location>
   </metadata>
</manifest>
```

Code Illustration 4-86: Using <adlcp:location> to Reference Metadata

Both examples illustrate how the XML can be built to describe the Content Aggregation and where it is to be located in the manifest file.

4.5.2. Metadata Describing Content Organizations

Content Organization level metadata describes a Content Organization. This metadata is used to facilitate reuse and discoverability within a content repository or similar system. This is accomplished by providing descriptive information about the Content Organization. The metadata is information about a Content Organization as a whole. It describes what the Content Organization is for, who can use it, who controls it, etc., and information that can be searched externally such as the Content Organization title, description and version.

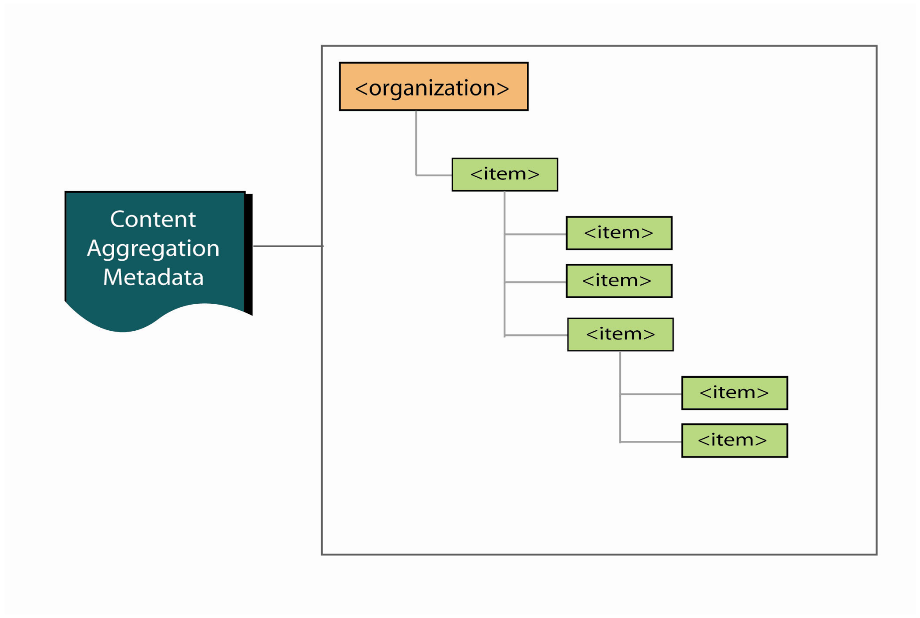


Figure 4.5.2a: Application of Metadata Describing a Content Organization

The metadata used to describe a Content Organization can be applied as "inline" metadata or referenced by the `<adlcp:location>` element. Specifically, it is applied to the `<organization>` elements in the package's manifest. For the sake of simplicity, the following examples will use the `<adlcp:location>` element to represent the reference to the metadata being described. It is important to note that this is done strictly for brevity. Inline metadata is permitted in all locations where the `<adlcp:location>` element is found.

```
<organizations>
   <organization>
      <title>Introduction to the SCORM</title>
      <item>…</item>
      <item>…</item>
      <metadata>
         <adlcp:location>contentOrganizationMetadata.xml</adlcp:location>
      </metadata>
   </organization>
</organizations>
```

Code Illustration 4-87: Organization Metadata Example

4.5.3. Metadata Describing Activities

Activity level metadata is metadata that describes activities (represented in `imsmanifest.xml` file as an `<item>` element). This metadata is used to facilitate reuse and discoverability within a content repository or similar system and to provide descriptive information about the activity. Activity metadata typically contains information about an activity as a whole that describes, in a context-sensitive manner, what it is for, who can use it, who controls it, etc.

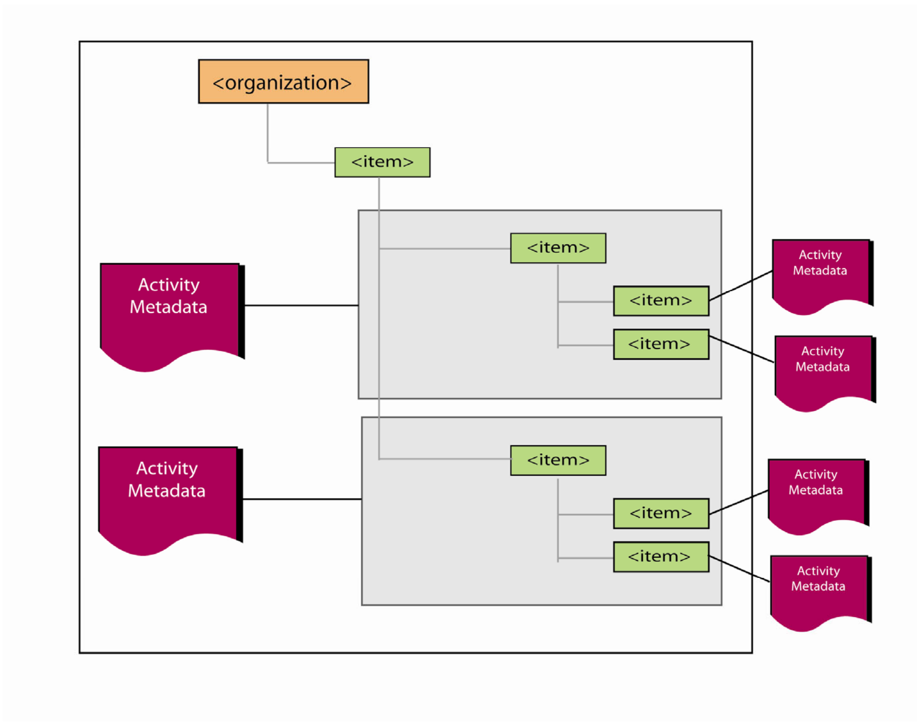


Figure 4.5.13a: Application of Metadata Describing an Activity

Activity Metadata is applied to the `<item>` elements in a content package's manifest.

```
<organizations>
   <organization>
      <title>Introduction to the SCORM</title>
      <item>
         <title>SCORM 101</title>
         <metadata>
            <adlcp:location>activityMetadata.xml</adlcp:location>
         </metadata>
      </item>
   </organization>
</organizations>
```

Code Illustration 4-88: Activity Metadata Example

4.5.4. Metadata Describing SCOs

Metadata can be associated with a SCO and should provide descriptive information about the learning resource independent of a particular context. This metadata is used to facilitate reuse and discoverability of such learning resources. SCO Metadata is metadata that describes a SCO that is not related to a specific Content Organization structure (i.e., context-independent metadata). The metadata contains information that can be searched externally such as content title, description, date of creation and version.

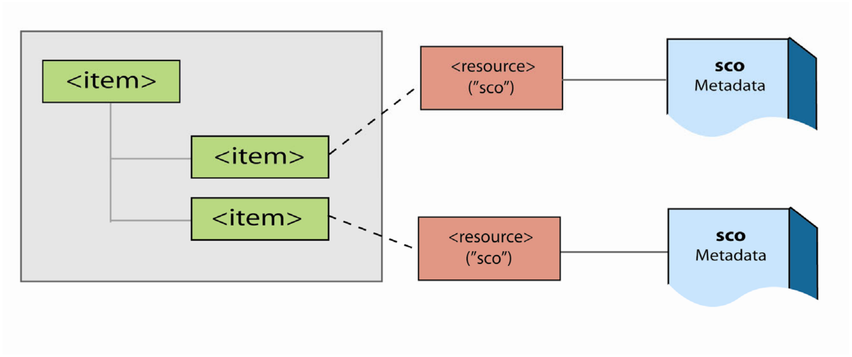


Figure 4.5.4a: Application of Metadata Describing a SCO

SCO Metadata is applied to `<resource>` elements with an `adlcp:scormType="sco"` attribute (i.e., SCO resource) in the content packages manifest.

```
<resources>
   <resource type="webcontent" adlcp:scormType="sco" href="sco1.htm">
      <metadata>
         <adlcp:location>SCOMetadata.xml</adlcp:location>
      </metadata>
   </resource>
</resources>
```

Code Illustration 4-89: Resource Metadata Example

4.5.5. Metadata Describing Assets

Metadata can be associated with Assets such as illustrations, documents or media streams. This Asset metadata should provide descriptive information about the Asset independent of learning content. This metadata is used to facilitate reuse and discoverability principally during learning content creation of such Assets. Asset metadata is metadata that describes Assets in a non-context-specific way that can be searched externally by title, description, date of creation and version and that can be used to create a searchable repository of sharable Assets.

Asset Metadata is applied to `<resource>` elements with an `adlcp:scormType="asset"` attribute (i.e., Asset resource) in the content packages manifest. Asset Metadata can also be applied to `<file>` elements found as child elements of `<resource>` elements.

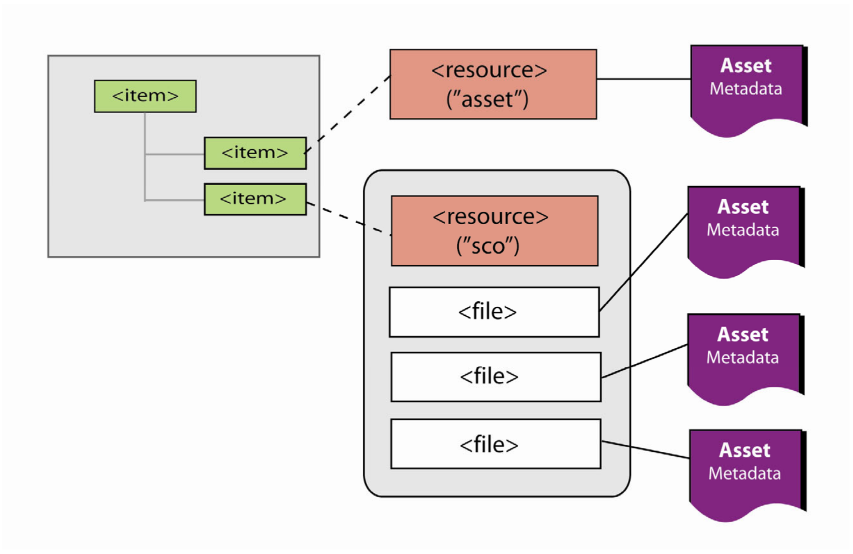


Figure 4.5.5a: Application of Metadata Describing Assets

The following example illustrates Asset Metadata, where the resource represents an asset.

```
<resources>
   <resource type="webcontent" adlcp:scormType="asset" href="asset.htm">
      <metadata>
         <adlcp:location>assetMetadata.xml</adlcp:location>
      </metadata>
   </resource>
</resources>
```

Code Illustration 4-90: Resource Asset Metadata Example

The following example illustrates Asset Metadata on a `<file>` element.

```
<resources>
   <resource type="webcontent" adlcp:scormType="asset" href="asset.htm">
      <file href="asset.htm">
         <metadata>
            <adlcp:location>assetMetadata.xml</adlcp:location>
         </metadata>
      </file>
   </resource>
</resources>
```

Code Illustration 4-91: File Asset Metadata Example

SECTION 5
SCORM® Sequencing and Presentation

This page intentionally left blank.

5.1. Sequencing and Presentation

This section describes how to encode specific sequencing strategies in XML. These sequencing strategies can then be placed in the IMS Manifest file to define sets of sequencing information for activities. There are two main ways of creating sequencing information:

- `<sequencing>` element. The `<sequencing>` element encapsulates all of the necessary sequencing information for a given activity.
- `<sequencingCollection>` element. The `<sequencingCollection>` element can be used to collect sets of sequencing information to be reused by several activities.

Activities are represented as `<item>` elements or `<organization>` elements within a manifest. The `<sequencing>` element can be placed as a child of an `<item>` or an `<organization>` element.

More details on sequencing information and strategies can be found in the SCORM SN book.

5.1.1. <sequencing> Element

Sequencing information is associated with items in a hierarchical structure by associating a single `<sequencing>` element with the hierarchical item. In the context of IMS Content Packages, this is done by including the `<sequencing>` element within either an `<item>` element or an `<organization>` element.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<sequencing>`

Data Type: The `<sequencing>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<sequencing>` element contains the following elements/attributes:

Attributes:

- `ID` (optional): The unique identifier assigned to this set of sequencing information. The `ID` attribute shall only occur on a `<sequencing>` element that is a child of a `<sequencingCollection>` element. The `ID` attribute is not permitted on a`<sequencing>` element that is a child of an `<item>` or `<organization>` element.

XML Data Type: `xs:ID`.

- `IDRef` (optional) – A reference to a unique identifier (i.e., `ID` attribute of a `<sequencing>` element) assigned to a set of sequencing information. The `IDRef` is used to link to reusable sequencing information defined somewhere in the same XML document. XML Data Type: `xs:IDREF`.

Elements:

- `<controlMode>`
- `<sequencingRules>`
- `<limitConditions>`
- `<auxiliaryResources>`
- `<rollupRules>`
- `<objectives>`
- `<randomizationControls>`
- `<deliveryControls>`
- `<adlseq:constrainedChoiceConsiderations>`
- `<adlseq:rollupConsiderations>`

Multiplicity: Occurs 1 or More times within the `<sequencingCollection>` element, if the `<sequencingCollection>` element is present. Occurs 0 or 1 time for each `<item>` or `<organization>` within an IMS content package.

Example:

```
<item identifier="INTRO" identifierref="RESOURCE_INTRO">
   <title>Photoshop Introduction</title>
   <imsss:sequencing>
      <imsss:limitConditions attemptLimit="1"/>
      <imsss:rollupRules rollupObjectiveSatisfied="false"/>
  </imsss:sequencing>
</item>
```

Code Illustration 5-1: <sequencing> Element

5.1.2. <controlMode> Element

The `<controlMode>` element is the container for the sequencing control mode information including descriptions of the types of sequencing behaviors specified for an activity [5]. This element captures information dealing with the types of sequencing requests that are permitted.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<controlMode>`

Data Type: The `<controlMode>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<controlMode>` element contains the following elements/attributes:

Attribute:

- `choice` (optional, default value = `true`) – Indicates that a choice sequencing request is permitted (or not permitted if value = `false`) to target the children of the activity [5]. XML Data Type: `xs:boolean`.
- `choiceExit` (optional, default value = `true`) – Indicates that an active child of this activity is permitted to terminate (or not permitted if value = `false`) if a choice sequencing request is processed [5]. XML Data Type: `xs:boolean`.
- `flow` (optional, default value = `false`) – Indicates the flow sequencing requests is permitted (or not permitted if value = `false`) to the children of this activity [5]. XML Data Type: `xs:boolean`.
- `forwardOnly` (optional, default value = `false`) – Indicates that backward targets (in terms of activity tree traversal) are not permitted (or are permitted if value = `false`) for the children of this activity [5]. XML Data Type: `xs:boolean`.
- `useCurrentAttemptObjectiveInfo` (optional, default value = `true`) – Indicates that the objective progress information for the children of the activity will only be used (or not used if value = `false`) in rule evaluations and rollup if that information was recorded during the current attempt on the activity [5]. XML Data Type: `xs:boolean`.
- `useCurrentAttemptProgressInfo` (optional, default value = `true`) – Indicates that the attempt progress information for the children of the activity will only be used (or not used if value = `false`) in rule evaluations and rollup if that information was recorded during the current attempt on the activity [5]. XML Data Type: `xs:boolean`.

Elements:

- None

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:controlMode choice="false" choiceExit="false"
             flow="true" forwardOnly = "true"/>
   </imsss:sequencing>
</item>
```

Code Illustration 5-2: <controlMode> Element

5.1.3. <sequencingRules> Element

The `<sequencingRules>` element is the container for a sequencing rule description. Each rule describes the sequencing behavior for an activity. Each activity may have an unlimited number of sequencing rules and within any grouping the rules are evaluated in the order in which they are listed [5].

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<sequencingRules>`

Data Type: The `<sequencingRules>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<sequencingRules>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<preConditionRule>`
- `<exitConditionRule>`
- `<postConditionRule>`

Multiplicity: Occurs 0 or More times in the `<sequencing>` element.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:sequencingRules>
         <imsss:preConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "satisfied"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "disabled"/>
         </imsss:preConditionRule>
      </imsss:sequencingRules>
   </imsss:sequencing>
</item>
```

Code Illustration 5-3: <sequencingRules> Element

5.1.3.1. <preConditionRule> Element

The `<preConditionRule>` element is the container for the description of 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 [5].

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<preConditionRule>`

Data Type: The `<preConditionRule>` element is a parent element. Parent elements have no values associated with them. Parent elements act as “containers” for other elements/attributes. The `<preConditionRule>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<ruleConditions>`
- `<ruleAction>`

Multiplicity: Occurs 0 or More times in the `<sequencingRules>` element.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:sequencingRules>
         <imsss:preConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "satisfied"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "disabled"/>
         </imsss:preConditionRule>
      </imsss:sequencingRules>
   </imsss:sequencing>
</item>
```

Code Illustration 5-4 <preConditionRule> Element

5.1.3.1.1. <ruleConditions> Element

The `<ruleConditions>` element is the container for the set of conditions that are to be applied either the pre-condition, post-condition and exit condition rules.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<ruleConditions>`

Data Type: The `<ruleConditions>` element is a parent element. Parent elements have no values associated with them. Parent elements act as “containers” for other elements/attributes. The `<ruleConditions>` element contains the following elements/attributes:

Attributes:

- `conditionCombination` (optional, default value = `all`): This attribute indicates how rule conditions (`<ruleCondition>`) are combined in evaluating the rule [5]. XML Data Type: `xs:token`.
  - **`all`**: The rule condition evaluates to true if and only if all of the individual rule conditions evaluates to true [5].
  - **`any`**: The rule condition evaluates to true if and only if any of the individual rule conditions evaluates to true [5].

Elements:

- `<ruleCondition>`

Multiplicity: Occurs 1 and only 1 time within the `<preConditionRule>`, `<exitConditionRule>` and `<postConditionRule>` elements.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:sequencingRules>
         <imsss:preConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "satisfied"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "disabled"/>
         </imsss:preConditionRule>
      </imsss:sequencingRules>
   </imsss:sequencing>
</item>
```

Code Illustration 5-5: <ruleConditions> Element

5.1.3.1.1.1. <ruleCondition>

The `<ruleCondition>` element represents the condition that is evaluated.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<ruleCondition>`

Data Type: The `<ruleCondition>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<ruleCondition>` element contains the following elements/attributes:

Attributes:

- `referencedObjective` (optional): This attribute represents the identifier of a local objective associated with the activity. The status of this objective will be used during the evaluation of the condition [5]. If provided, the value of the `referencedObjective` shall contain an `objectiveID` of either the `<primaryObjective>` or an `<objective>` element defined for the activity. The underlying data type for the `referencedObjective`, as defined by the IMS Simple Sequencing Specification, is a unique identifier. Since an empty characterstring does not provide sufficient semantic information to uniquely identify which objective is being referenced, then the `referencedObjective` attribute cannot be an empty characterstring and cannot contain all whitespace characters (which could be transcribed as an empty characterstring by an XML parser). If the `referencedObjective` is omitted, then the primary objective for the activity shall be assumed when the rule is evaluated, regardless of whether the primary objective is formally declared or not. XML Data Type: `xs:string`.

- `measureThreshold` (optional): The value used as a threshold during measure-based condition evaluations [5]. XML Data Type: `xs:decimal` (Range -1.0000 to 1.0000, with a precision of at least four decimal places).
- `operator` (optional, default value = `noOp`): The unary logical operator to be applied to the condition [5]. XML Data Type: `xs:token`.
    - **`not`**: The corresponding condition is negated in the rule evaluation [5].
    - **`noOp`**: The corresponding condition is used as is in rule evaluation [5].
- `condition` (required): This attribute represents the actual condition for the rule [5]. XML Data Type: `xs:token`. The following is a listing of the vocabulary tokens to be used for the `condition` attribute:
    - **`satisfied`**
    - **`objectiveStatusKnown`**
    - **`objectiveMeasureKnown`**
    - **`objectiveMeasureGreaterThan`**
    - **`objectiveMeasureLessThan`**
    - **`completed`**
    - **`activityProgressKnown`**
    - **`attempted`**
    - **`attemptLimitExceeded`**
    - **`timeLimitExceeded`**
    - **`outsideAvailableTimeRange`**
    - **`always`**

Elements:

- None

Multiplicity: Occurs 1 or More times within the `<ruleConditions>` element.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:sequencingRules>
         <imsss:preConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "satisfied"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "disabled"/>
         </imsss:preConditionRule>
      </imsss:sequencingRules>
   </imsss:sequencing>
</item>
```

Code Illustration 5-6: <ruleCondition> Element

5.1.3.1.2. <ruleAction> Element

The `<ruleAction>` element is the desired sequencing behavior if the rule evaluates to true. The set of rule actions vary depending on the type of condition (`<preConditionRule>`, `<postConditionRule>`, or `<exitConditionRule>`).

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<ruleAction>`

Data Type: The `<ruleAction>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<ruleAction>` element contains the following elements/attributes:

Attributes:

- `action` (required, if no action is defined the action is ignored): The action represents the desired sequencing behavior if the rule condition evaluates to true [5].
  - If `action` is defined in a `<preConditionRule>`, then the `action` attribute shall have one of the following values:
    - **`skip`**
    - **`disabled`**
    - **`hiddenFromChoice`**
    - **`stopForwardTraversal`**

- If action is defined in a `<postConditionRule>`, then the `action` attribute shall have one of the following values:
    - **`exitParent`**
    - **`exitAll`**
    - **`retry`**
    - **`retryAll`**
    - **`continue`**
    - **`previous`**
- If action is defined in a `<exitConditionRule>`, then the `action` attribute shall have one of the following values:
    - **`exit`**

Elements:

- None

Multiplicity: Occurs 1 and only 1 time within a `<preConditionRule>`, `<postConditionRule>` or `<exitConditionRule>` element.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:sequencingRules>
         <imsss:preConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "satisfied"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "disabled"/>
         </imsss:preConditionRule>
      </imsss:sequencingRules>
   </imsss:sequencing>
</item>
```

Code Illustration 5-7: <ruleAction> Element

5.1.3.2. <postConditionRule> Element

The `<postConditionRule>` element is the container for the description of actions that control sequencing decisions and delivery of a specific activity. Rules that include such actions are applied when the activity attempt terminates [5].

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<postConditionRule>`

Data Type: The `<postConditionRule>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<postConditionRule>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<ruleConditions>` (Refer to Section 5.1.3.1.1: *<ruleConditions> Element* for more details)
- `<ruleAction>` (Refer to Section 5.1.3.1.2: *<ruleAction> Element* for more details)

Multiplicity: Occurs 0 or More times in the `<sequencingRules>` element.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:sequencingRules>
         <imsss:postConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition="satisfied"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action="exitParent"/>
         </imsss:postConditionRule>
         <imsss:postConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "always"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "continue"/>
         </imsss:postConditionRule>
      </imsss:sequencingRules>
   </imsss:sequencing>
</item>
```

Code Illustration 5-8: <postConditionRule> Element

5.1.3.3. <exitConditionRule> Element

The `<exitConditionRule>` element is the container for the description of actions that control sequencing decisions and delivery of a specific activity. Rules that include such actions are applied after an activity attempt on a descendent activity terminates [5].

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<exitConditionRule>`

Data Type: The `<exitConditionRule>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<exitConditionRule>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<ruleConditions>` (Refer to Section 5.1.3.1.1: *<ruleConditions> Element* for more details
- `<ruleAction>` (Refer to Section 5.1.3.1.2: *<ruleAction> Element* for more details)

Multiplicity: Occurs 0 or More times in the `<sequencingRules>` element.

Example:

```
<item identifier="PRETEST1">
   <title>Module 1 --  Pretest</title>
   <item identifier="PRETEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="PRETEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <imsss:sequencing>
      <imsss:sequencingRules>
         <imsss:exitConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition="satisfied"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action="exit"/>
         </imsss:exitConditionRule>
      </imsss:sequencingRules>
   </imsss:sequencing>
</item>
```

Code Illustration 5-9: <exitConditionRule> Element

5.1.4. <limitConditions> Element

At this time, ADL recommends the use of the `<limitConditions>` element with caution. Various concerns have been raised about time tracking in order to evaluate these conditions. ADL plans to continue evaluating this element and its impact on sequencing, behaviors and strategies.

ADL supports the usage of only two current limit conditions. The limit condition deals with attempts on the activity and maximum time allowed in the attempt. This section describes that attempt limit and maximum time allowed in the attempt attribute and the requirements for use.

For more information on the other limit conditions, refer to the IMS Simple Sequencing Specification [5].

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<limitConditions>`

Data Type: The `<limitConditions>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<limitConditions>` element contains the following elements/attributes:

Attributes:

- `attemptLimit` (optional): This value indicates the maximum number of attempts for the activity [5]. XML Data Type: `xs:nonNegativeInteger`.
- `attemptAbsoluteDurationLimit` (optional): This value indicates the maximum time duration that the learner is permitted to spend on any single learner 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 [5]. This element is used to initialize the `cmi.max_time_allowed` (refer to the SCORM RTE Book [2]). Currently, the SCO is responsible for all time tracking and behaviors due to timing violations. XML Data Type: `xs:duration`.

Elements:

- None

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="INTRO" identifierref="RESOURCE_INTRO">
   <title>Photoshop Introduction</title>
   <imsss:sequencing>
      <imsss:limitConditions attemptLimit="1"/>
  </imsss:sequencing>
</item>
```

Code Illustration 5-10: <limitConditions> Element

5.1.5. <auxiliaryResources> Element

At this time, ADL recommends to use the `<auxiliaryResources>` element with caution. Various concerns have risen about defining the requirements on the usage of auxiliary resources (e.g., Can auxiliary resources be SCOs? Is sequencing information applied to auxiliary resources?). ADL plans to continue to evaluating this element and its impact on sequencing, behaviors and strategies.

For more information on the auxiliary resources, refer to the IMS SS [5].

5.1.6. <rollupRules> Element

The `<rollupRules>` element is the container for the set of rollup rules defined for the activity.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<rollupRules>`

Data Type: The `<rollupRules>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<rollupRules>` element contains the following elements/attributes:

Attributes:

- `rollupObjectiveSatisfied` (optional, default value = `true`): This attribute indicates that the objective's satisfied status associated with the activity is included in the rollup for its parent activity [5]. XML Data Type: `xs:boolean`.
- `rollupProgressCompletion` (optional, default value = `true`): This attribute indicates that the attempt's completion status associated with the activity is included in the rollup for its parent activity [5]. XML Data Type: `xs:boolean`.
- `objectiveMeasureWeight`(optional, default value = `1.0000`): This attribute indicates the weighting factor applied to the objectives normalized measure used during rollup for the parent activity [5]. XML Data Type: `xs:decimal` (Range 0.0000 to 1.0000 (precision to at least 4 significant decimal places).

Elements:

- `<rollupRule>`

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="MODULE2">
   <title>Module 2 -- Enhancing Images</title>
   <item identifier="PRETEST2">
      <title>Module 2 -- Pretest</title>
       <item identifier="PRETEST_QUESTION4" isvisible = "false"
             identifierref="RESOURCE_QUESTION4">
          <title>Question 4</title>
       </item>
       <item identifier="PRETEST_QUESTION5" isvisible = "false"
             identifierref="RESOURCE_QUESTION5">
          <title>Question 5</title>
       </item>
       <item identifier="PRETEST_QUESTION6" isvisible = "false"
             identifierref="RESOURCE_QUESTION6">
          <title>Question 6</title>
       </item>
       <imsss:sequencing>
          <imsss:rollupRules >
             <imsss:rollupRule childActivitySet = "all">
                <imsss:rollupConditions>
                   <imsss:rollupCondition condition = "attempted"/>
                </imsss:rollupConditions>
                <imsss:rollupAction action = "completed"/>
             </imsss:rollupRule>
          </imsss:rollupRules>
       </imsss:sequencing>
    </item>
</item>
```

Code Illustration 5-11: <rollupRules> Element

5.1.6.1. <rollupRule> Element

The `<rollupRule>` element is the container for each rollup rule that is to be applied to an activity. The general format for a rule can be expressed informally as “If child-activity set, condition set Then action”. Multiple conditions are permitted.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<rollupRule>`

Data Type: The `<rollupRule>` element is a parent element. Parent elements have no values associated with them. Parent elements act as “containers” for other elements/attributes. The `<rollupRule>` element contains the following elements/attributes:

Attributes:

- `childActivitySet` (optional, default value = `all`): This attribute indicates whose data values are used to evaluate the rollup condition [5]. XML Data Type: `xs:token`. The following is a list of vocabularies permitted to be used:
  - **`all`**

        - **any**
        - **none**
        - **atLeastCount**
        - **atLeastPercent**
    - `minimumCount` (optional, default value = `0`): The `minimumCount` attribute shall be used when the `childActivitySet` attribute is set to `atLeastCount`. The rollup rule condition evaluates to true if at least the number of children specified by this attribute have a rollup condition of true [5]. XML Data Type: `xs:nonNegativeInteger`.
    - `minimumPercent` (optional, default value = `0.0000`): The `minimumPercent` attribute shall be used when the `childActivitySet` attribute is set to `atLeastPercent`. The rollup rule condition evaluates to true if at least the percentage of children specified by this attribute have a rollup condition value of true [5].XML Data Type: `xs:decimal` (Range 0.0000 to 1.0000 (precision to at least 4 significant decimal places).

Elements:

- `<rollupConditions>`
- `<rollupAction>`

Multiplicity: Occurs 0 or More times in the `<rollupRules>` element.

Example:

```
<item identifier="MODULE2">
   <title>Module 2 -- Enhancing Images</title>
   <item identifier="PRETEST2">
      <title>Module 2 -- Pretest</title>
       <item identifier="PRETEST_QUESTION4" isvisible = "false"
             identifierref="RESOURCE_QUESTION4">
          <title>Question 4</title>
       </item>
       <item identifier="PRETEST_QUESTION5" isvisible = "false"
             identifierref="RESOURCE_QUESTION5">
          <title>Question 5</title>
       </item>
       <item identifier="PRETEST_QUESTION6" isvisible = "false"
             identifierref="RESOURCE_QUESTION6">
          <title>Question 6</title>
       </item>
       <imsss:sequencing>
          <imsss:rollupRules >
             <imsss:rollupRule childActivitySet = "all">
                <imsss:rollupConditions>
                   <imsss:rollupCondition condition = "attempted"/>
                </imsss:rollupConditions>
                <imsss:rollupAction action = "completed"/>
             </imsss:rollupRule>
          </imsss:rollupRules>
       </imsss:sequencing>
    </item>
</item>
```

Code Illustration 5-12: <rollupRule> Element

5.1.6.1.1. <rollupConditions> Element

The `<rollupConditions>` element is the container for the set of conditions that are applied within a single rollup rule.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<rollupConditions>`

Data Type: The `<rollupConditions>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<rollupConditions>` element contains the following elements/attributes:

Attributes:

- `conditionCombination` (optional, default value = `any`): This attribute indicates how the rollup conditions are to be combined. XML Data Type: `xs:token`. The attribute contains a value of one of the following tokens:
  - **`all`**
  - **`any`**

Elements:

- `<rollupCondition>`

Multiplicity: Occurs 1 and only 1 time in the `<rollupRule>` element.

Example:

```
<item identifier="MODULE2">
   <title>Module 2 -- Enhancing Images</title>
   <item identifier="PRETEST2">
      <title>Module 2 -- Pretest</title>
       <item identifier="PRETEST_QUESTION4" isvisible = "false"
             identifierref="RESOURCE_QUESTION4">
          <title>Question 4</title>
       </item>
       <item identifier="PRETEST_QUESTION5" isvisible = "false"
             identifierref="RESOURCE_QUESTION5">
          <title>Question 5</title>
       </item>
       <item identifier="PRETEST_QUESTION6" isvisible = "false"
             identifierref="RESOURCE_QUESTION6">
          <title>Question 6</title>
       </item>
       <imsss:sequencing>
          <imsss:rollupRules >
             <imsss:rollupRule childActivitySet = "all">
                <imsss:rollupConditions>
                   <imsss:rollupCondition condition = "attempted"/>
                </imsss:rollupConditions>
                <imsss:rollupAction action = "completed"/>
             </imsss:rollupRule>
          </imsss:rollupRules>
       </imsss:sequencing>
    </item>
</item>
```

Code Illustration 5-13: <rollupConditions> Element

5.1.6.1.2. <rollupCondition> Element

The `<rollupCondition>` element identifies a condition to be applied in the rollup rule [5].

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<rollupCondition>`

Data Type: The `<rollupCondition>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<rollupCondition>` element contains the following elements/attributes:

Attributes:

- `operator` (optional, default value = `noOp`): The unary logical operator to be applied to the individual condition [5]. XML Data Type: `xs:token`. The attribute contains a value of one of the following tokens:
  - **`not`**
  - **`noOp`**

- `condition` (required): Indicates the condition element for the rule. XML Data Type:`xs:token`. The attribute contains a value of one of the following tokens:
    - **`satisfied`**
    - **`objectiveStatusKnown`**
    - **`objectiveMeasureKnown`**
    - **`completed`**
    - **`activityProgressKnown`**
    - **`attempted`**
    - **`attemptLimitExceeded`**
    - **`timeLimitExceeded`**
    - **`outsideAvailableTimeRange`**

Elements:

- None

Multiplicity: Occurs 1 or More times in the `<rollupConditions>` element.

Example:

```
<item identifier="MODULE2">
   <title>Module 2 -- Enhancing Images</title>
   <item identifier="PRETEST2">
      <title>Module 2 -- Pretest</title>
       <item identifier="PRETEST_QUESTION4" isvisible = "false"
             identifierref="RESOURCE_QUESTION4">
          <title>Question 4</title>
       </item>
       <item identifier="PRETEST_QUESTION5" isvisible = "false"
             identifierref="RESOURCE_QUESTION5">
          <title>Question 5</title>
       </item>
       <item identifier="PRETEST_QUESTION6" isvisible = "false"
             identifierref="RESOURCE_QUESTION6">
          <title>Question 6</title>
       </item>
       <imsss:sequencing>
          <imsss:rollupRules >
             <imsss:rollupRule childActivitySet = "all">
                <imsss:rollupConditions>
                   <imsss:rollupCondition condition = "attempted"/>
                </imsss:rollupConditions>
                <imsss:rollupAction action = "completed"/>
             </imsss:rollupRule>
          </imsss:rollupRules>
       </imsss:sequencing>
    </item>
</item>
```

Code Illustration 5-14: <rollupCondition> Element

5.1.6.1.3. <rollupAction> Element

The `<rollupAction>` element identifies a condition to be applied in the rollup rule [5].

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<rollupAction>`

Data Type: The `<rollupAction>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<rollupAction>` element contains the following elements/attributes:

Attributes:

- `action` (required): This attribute indicates the desired rollup behavior if the rule evaluates to true. XML Data Type: `xs:token`. The attribute contains a value of one of the following tokens:
    - **`satisfied`**
    - **`notSatisfied`**
    - **`completed`**
    - **`incomplete`**

Elements:

- None

Multiplicity: Occurs 1 and only 1 time in the `<rollupRule>` element.

Example:

```
<item identifier="MODULE2">
   <title>Module 2 -- Enhancing Images</title>
   <item identifier="PRETEST2">
      <title>Module 2 -- Pretest</title>
       <item identifier="PRETEST_QUESTION4" isvisible = "false"
             identifierref="RESOURCE_QUESTION4">
          <title>Question 4</title>
       </item>
       <item identifier="PRETEST_QUESTION5" isvisible = "false"
             identifierref="RESOURCE_QUESTION5">
          <title>Question 5</title>
       </item>
       <item identifier="PRETEST_QUESTION6" isvisible = "false"
             identifierref="RESOURCE_QUESTION6">
          <title>Question 6</title>
       </item>
       <imsss:sequencing>
          <imsss:rollupRules >
             <imsss:rollupRule childActivitySet = "all">
                <imsss:rollupConditions>
                   <imsss:rollupCondition condition = "attempted"/>
                </imsss:rollupConditions>
                <imsss:rollupAction action = "completed"/>
             </imsss:rollupRule>
          </imsss:rollupRules>
       </imsss:sequencing>
    </item>
</item>
```

Code Illustration 5-15: <rollupAction> Element

5.1.7. <objectives> Element

The `<objectives>` element is the container for the set of objectives that are to be associated with an activity [5]. Each activity must have at least one primary objective and may have an unlimited number of objectives.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<objectives>`

Data Type: The `<objectives>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<objectives>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<primaryObjective>`
- `<objective>`

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="POSTTEST1">
   <title>Module 1 --  Posttest</title>
   <item identifier="POSTTEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="POSTTEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <item identifier="POSTTEST_QUESTION3" isvisible = "false"
         identifierref="RESOURCE_QUESTION3">
      <title>Question 3</title>
   </item>
   <imsss:sequencing>
      <imsss:objectives>
         <imsss:primaryObjective objectiveID = "PRIMARYOBJ"
                satisfiedByMeasure = "true">
            <imsss:minNormalizedMeasure>0.6</imsss:minNormalizedMeasure>
            <imsss:mapInfo targetObjectiveID = "obj_module_1"
                     readNormalizedMeasure = "false"
                     writeSatisfiedStatus = "true" />
         </imsss:primaryObjective>
      </imsss:objectives>
   </imsss:sequencing>
</item>
```

Code Illustration 5-16: <objectives> Element

5.1.7.1. <primaryObjective> Element

The `<primaryObjective>` element identifies the objective that contributes to the rollup associated with the activity [5]. If the `<objectives>` element is defined then the `<primaryObjective>` is mandatory. The element may be represented as an empty element (`<primaryObjective/>`).

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<primaryObjective>`

Data Type: The `<primaryObjective>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<primaryObjective>` element contains the following elements/attributes:

Attributes:

- `satisfiedByMeasure` (optional, default value = `false`): This attribute indicates that the `<minNormalizedMeasure>` shall be used (if value is set to true) in place of any other method to determine if the objective associated with the activity is satisfied [5]. XML Data Type: `xs:boolean`.

- `objectiveID` (optional): The identifier of the objective associated with the activity [5]. The underlying data type for the `objectiveID`, as defined by IMS SS, is a unique identifier. Since an empty characterstring does not provide sufficient semantic information to uniquely identify the objective, then the `objectiveID` attribute cannot be an empty characterstring and cannot contain all whitespace characters (which could be transcribed as an empty characterstring by an XML parser).

  If a `<primaryObjective>` contains an objective map (`<mapInfo>`), then the `objectiveID` attribute is required. If the `<primaryObjective>` does not contain an objective map, then the `objectiveID` attribute is optional.

  For a given set of objectives defined for an activity (i.e., one `<primaryObjective>` and multiple `<objective>` elements in a given `<objectives>` element), all defined `objectiveID` attributes shall be unique. LMSs shall use the value held by the `objectiveID` attribute to initialize the `cmi.objectives.n.id` data model element. XML Data Type: `xs:anyURI`.

Elements:

- `<minNormalizedMeasure>`
- `<mapInfo>`

Multiplicity: Occurs 1 and only 1 time in the `<objectives>` element.

Example:

```
<item identifier="POSTTEST1">
   <title>Module 1 --  Posttest</title>
   <item identifier="POSTTEST_QUESTION1" isvisible = "false"
            identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="POSTTEST_QUESTION2" isvisible = "false"
            identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <item identifier="POSTTEST_QUESTION3" isvisible = "false"
            identifierref="RESOURCE_QUESTION3">
      <title>Question 3</title>
   </item>
   <imsss:sequencing>
      <imsss:objectives>
         <imsss:primaryObjective objectiveID = "PRIMARYOBJ"
                satisfiedByMeasure = "true">
            <imsss:minNormalizedMeasure>0.6</imsss:minNormalizedMeasure>
            <imsss:mapInfo targetObjectiveID = "obj_module_1"
                      readNormalizedMeasure = "false"
                      writeSatisfiedStatus = "true" />
         </imsss:primaryObjective>
      </imsss:objectives>
   </imsss:sequencing>
</item>
```

Code Illustration 5-17: <primaryObjective> Element

5.1.7.1.1. <minNormalizedMeasure> Element

The `<minNormalizedMeasure>` element identifies minimum satisfaction measure for the objective [5]. The value is normalized between –1 and 1 (inclusive). If the primary objective (i.e., `<primaryObjective>` element) has `satisfiedByMeasure` equal to `true`, then the LMS shall use this value to initialize the `cmi.scaled_passing_score` (refer to the SCORM RTE Book[2]).

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<minNormalizedMeasure>`

Data Type: The `<minNormalizedMeasure>` element's value shall be of type `xs:decimal`. The default value if no value is provided is `1.0`.

Multiplicity: Occurs 0 or 1 time in the `<primaryObjective>` and `<objective>` elements.

Example:

```
<item identifier="POSTTEST1">
   <title>Module 1 --  Posttest</title>
   <item identifier="POSTTEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="POSTTEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <item identifier="POSTTEST_QUESTION3" isvisible = "false"
         identifierref="RESOURCE_QUESTION3">
      <title>Question 3</title>
   </item>
   <imsss:sequencing>
      <imsss:objectives>
         <imsss:primaryObjective objectiveID = "PRIMARYOBJ"
                satisfiedByMeasure = "true">
                <imsss:minNormalizedMeasure>0.6</imsss:minNormalizedMeasure>
                <imsss:mapInfo targetObjectiveID = "obj_module_1"
                      readNormalizedMeasure = "false"
                      writeSatisfiedStatus = "true" />
         </imsss:primaryObjective>
      </imsss:objectives>
   </imsss:sequencing>
</item>
```

Code Illustration 5-18: <minNormalizedMeasure> Element

5.1.7.1.2. <mapInfo> Element

The `<mapInfo>` element is the container for the objective map description. This defines the mapping of an activity's local objective information to and from a shared global objective. Each activity may have an unlimited number of objective maps**.**

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<mapInfo>`

Data Type: The `<mapInfo>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<mapInfo>` element contains the following elements/attributes:

Attributes:

- `targetObjectiveID` (required): The identifier of the global shared objective targeted for the mapping [5]. The underlying data type for the `targetObjectiveID`, as defined by IMS SS, is a unique identifier. Since an empty characterstring does not provide sufficient semantic information to uniquely identify which global shared objective is being targeted, then the `targetObjectiveID` attribute cannot be an empty characterstring and cannot contain all whitespace characters (which could be transcribed as an empty characterstring by an XML parser). XML Data Type: `xs:anyURI`.
- `readSatisfiedStatus` (optional, default value = `true`): This attribute indicates that the satisfaction status for the identified local objective should be retrieved (true or false) from the identified shared global objective when the progress for the local objective is undefined [5]. XML Data Type: `xs:boolean.`
- `readNormalizedMeasure` (optional, default value = `true`): This attribute indicates that the normalized measure for the identified local objective should be retrieved (true or false) from the identified shared global objective when the measure for the local objective is undefined [5].XML Data Type: `xs:boolean.`
- `writeSatisfiedStatus` (optional, default value = `false`): This attribute indicates that the satisfaction status for the identified local objective should be transferred (true or false) to the identified shared global objective upon termination (`Termination("")`) of the attempt on the activity. [5]. XML Data Type: `xs:boolean.`
- `writeNormalizedMeasure` (optional, default value = `false`): This attribute indicates that the normalized measure for the identified local objective should be transferred (true or false) to the identified shared global objective upon termination (`Termination("")`) of the attempt on the activity. [5]. XML Data Type: `xs:boolean.`

Elements:

- None

Multiplicity: Occurs 0 or More times on the `<primaryObjective>` and `<objective>` elements.

For Read Objective Maps (`readSatisfiedStatus` and `readNormalizedMeasure`):

- If multiple `<mapInfo>` elements exist for an objective (`<primaryObjective>` or `<objective>`), then only one `<mapInfo>` element shall have the `readSatisfiedStatus` attribute set to "true".
- If multiple `<mapInfo>` elements exist for an objective (`<primaryObjective>` or `<objective>`), then only one `<mapInfo>` element shall have the `readNormalizedMeasure` attribute set to "true".

For Write Objective Maps (`writeSatisfiedStatus` and `writeNormalizedMeasure`):

- For an activity, if multiple objectives (`<primaryObjective>` or `<objective>`) have `<mapInfo>` elements that share the same `targetObjectiveID`, then only one of the objectives shall have the `writeSatisfiedStatus` attribute set to "true".
- For any activity, if multiple objectives (`<primaryObjective>` or `<objective>`) have `<mapInfo>` elements that share the same `targetObjectiveID`, then only one of the objectives shall have the `writeNormalizedMeasure` attribute set to "true".

Example:

```
<item identifier="POSTTEST1">
   <title>Module 1 --  Posttest</title>
   <item identifier="POSTTEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="POSTTEST_QUESTION2" isvisible = "false"
         identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <item identifier="POSTTEST_QUESTION3" isvisible = "false"
         identifierref="RESOURCE_QUESTION3">
      <title>Question 3</title>
   </item>
   <imsss:sequencing>
      <imsss:objectives>
         <imsss:primaryObjective objectiveID = "PRIMARYOBJ"
                satisfiedByMeasure = "true">
            <imsss:minNormalizedMeasure>0.6</imsss:minNormalizedMeasure>
            <imsss:mapInfo targetObjectiveID = "obj_module_1"
                      readNormalizedMeasure = "false"
                      writeSatisfiedStatus = "true" />
         </imsss:primaryObjective>
      </imsss:objectives>
   </imsss:sequencing>
</item>
```

Code Illustration 5-19: <mapInfo> Element

5.1.7.2. <objective> Element

The `<objective>` element identifies objectives that do not contribute to rollup associated with the activity. This element can only exist if a `<primaryObjective>` has been defined.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<objective>`

Data Type: The `<objective>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<objective>` element contains the following elements/attributes:

Attributes:

- `satisfiedByMeasure` (optional, default value = `false`): This attribute indicates that the `<minNormalizedMeasure>` shall be used (if value is set to `true`) in place of any other method to determine if the objective associated with the activity is satisfied [5]. XML Data Type: `xs:boolean`.
- `objectiveID` (mandatory): The identifier of the objective associated with the activity [5]. The underlying data type for the `objectiveID`, as defined by the IMS Simple Sequencing Specification, is a unique identifier. Since an empty characterstring does not provide sufficient semantic information to uniquely identify the objective, then the o`bjectiveID` attribute cannot be an empty characterstring and cannot contain all whitespace characters (which could be transcribed as an empty characterstring by an XML parser).

  For a given set of objectives defined for an activity (i.e., one `<primaryObjective>` and multiple `<objective>` elements in a given `<objectives>` element), all defined `objectiveID` attributes shall be unique. LMSs shall use the value held by the `objectiveID` attribute to initialize the `cmi.objectives.n.id` data model element. XML Data Type: `xs:anyURI`.

Elements:

- `<minNormalizedMeasure>` (Refer to Section 5.1.7.1.1: *<minNormalizedMeasure> Element*)
- `<mapInfo>` (Refer to Section 5.1.7.1.2: *<mapInfo> Element*)

Multiplicity: Occurs 0 or More times in the `<objectives>` element.

Example:

```
<item identifier="POSTTEST1">
   <title>Module 1 --  Posttest</title>
   <item identifier="POSTTEST_QUESTION1" isvisible = "false"
         identifierref="RESOURCE_QUESTION1">
      <title>Question 1</title>
   </item>
   <item identifier="POSTTEST_QUESTION2" isvisible = "false"
            identifierref="RESOURCE_QUESTION2">
      <title>Question 2</title>
   </item>
   <item identifier="POSTTEST_QUESTION3" isvisible = "false"
            identifierref="RESOURCE_QUESTION3">
      <title>Question 3</title>
   </item>
   <imsss:sequencing>
      <imsss:objectives>
         <imsss:primaryObjective objectiveID = "PRIMARYOBJ" />
         <imsss:objective objectiveID="obj_module_1">
            <imsss:mapInfo targetObjectiveID="obj_module_1"
                   readSatisfiedStatus = "false"
                   readNormalizedMeasure = "false"
                   writeSatisfiedStatus = "true" />
         </imsss:objective>
      </imsss:objectives>
   </imsss:sequencing>
</item>
```

Code Illustration 5-20: <objective> Element

5.1.8. <randomizationControls> Element

The `<randomizationControls>` element is the container for the descriptions of how the children of an activity should be ordered during the sequencing process.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<randomizationControls>`

Data Type: The `<randomizationControls>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<randomizationControls>` element contains the following elements/attributes:

Attributes:

- `randomizationTiming` (optional, default = `never`): This attribute indicates when the ordering of the children of the activity should occur [5]. XML Data Type: xs:`token`. The attribute contains a value of one of the following tokens:
  - **`never`**
  - **`once`**
  - **`onEachNewAttempt`**
- `selectCount` (optional): This attribute indicates the number of child activities that must be selected from the set of child activities associated with the activity [5]. XML Data Type: `xs:nonNegativeInteger`.
- `reorderChildren` (optional, default value = `false`): This attribute indicates that the order of the child activities is randomized [5]. XML Data Type: `xs:boolean`.
- `selectionTiming` (optional, default = `never`): This attribute indicates when the selection should occur. XML Data Type: `xs:token`. The attribute contains a value of one of the following tokens:
  - **`never`**
  - **`once`**
  - **`onEachNewAttempt`**

Elements:

- None

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="POSTTEST1">
   <title>Module 1 --  Posttest</title>
      <item identifier="POSTTEST_QUESTION1" isvisible = "false"
            identifierref="RESOURCE_QUESTION1">
         <title>Question 1</title>
      </item>
      <item identifier="POSTTEST_QUESTION2" isvisible = "false"
            identifierref="RESOURCE_QUESTION2">
         <title>Question 2</title>
      </item>
      <item identifier="POSTTEST_QUESTION3" isvisible = "false"
            identifierref="RESOURCE_QUESTION3">
         <title>Question 3</title>
      </item>
   <imsss:sequencing>
      <imsss:randomizationControls selectCount="2"
                                   selectionTiming="onEachNewAttempt" />
   </imsss:sequencing>
</item>
```

Code Illustration 5-21: <randomizationControls> Element

5.1.9. <deliveryControls> Element

The `<deliveryControls>` element is the container for the descriptions of how the children of an activity should be ordered during the sequencing process.

XML Namespace: `http://www.imsglobal.org/xsd/imsss`

XML Namespace Prefix: `imsss`

XML Binding Representation: `<deliveryControls>`

Data Type: The `<deliveryControls>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<deliveryControls>` element contains the following elements/attributes:

Attributes:

- `tracked` (optional, default value = `true`): This attribute indicates that the objective progress information and activity/attempt progress information for the attempt should be recorded (true or false) and the data will contribute to the rollup for its parent activity [5]. XML Data Type: xs:boolean.
- `completionSetByContent` (optional, default value = `false`): This attribute indicates that the attempt completion status for the activity will be set by the SCO (true or false) [5]. XML Data Type: `xs:boolean`.
- `objectiveSetByContent` (optional, default value = `false`): This attribute indicates that the objective satisfied status for the activity's associated objective that contributes to rollup will be set by the SCO. XML Data Type: `xs:boolean`.

Elements:

- None

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="LESSON1" identifierref="RESOURCE_LESSON1">
   <title>Lesson 1 -- Interface</title>
      <imsss:sequencing>
         <imsss:deliveryControls tracked = "false"/>
      </imsss:sequencing>
</item>
```

Code Illustration 5-22: <deliveryControls> Element

5.1.10. <constrainedChoiceConsiderations> Element

The `<adlseq:constrainedChoiceConsiderations>` element is the container for the descriptions of how choice navigation requests should be constrained during the sequencing process. The constrained choice only applies to the activity for which it is defined.

XML Namespace: `http://www.adlnet.org/xsd/adlseq_v1p3`

XML Namespace Prefix: `adlseq`

XML Binding Representation: `<constrainedChoiceConsiderations>`

Data Type: The `<constrainedChoiceConsiderations>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<constrainedChoiceConsiderations>` element contains the following elements/attributes:

Attributes:

- `preventActivation` (optional, default value = `false`): This attribute indicates that attempts on children activities should not begin unless the current activity is the parent. XML Data Type: xs:boolean.
- `constrainChoice` (optional, default value = `false`): This attribute indicates that only activities which are logically "next" from the constrained activities can be targets of a choice navigation request. XML Data Type: `xs:boolean`.

Elements:

- None

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="Module1">
   <item identifier="EXAM1">
      <title>Module 1 -- Exam</title>
      <item identifier="QUESTION1" isvisible = "false"
            identifierref="RESOURCE_QUESTION1">
         <title>Question 1</title>
      </item>
      <item identifier="QUESTION2" isvisible = "false"
            identifierref="RESOURCE_QUESTION2">
         <title>Question 2</title>
      </item>
      <item identifier="QUESTION3" isvisible = "false"
            identifierref="RESOURCE_QUESTION3">
         <title>Question 3</title>
      </item>
      <imsss:sequencing>
         <imsss:controlMode choice="false" choiceExit ="false" flow="true"
                            forwardOnly="true"/>
         <imsss:rollupRules>
            <imsss:rollupRule childActivitySet="all">
               <imsss:rollupConditions>
                  <imsss:rollupCondition condition="attempted"/>
               </imsss:rollupConditions>
               <imsss:rollupAction action="completed"/>
            </imsss:rollupRule>
         </imsss:rollupRules>
         <imsss:objectives>
            <imsss:primaryObjective satisfiedByMeasure="true">
               <imsss:minNormalizedMeasure>0.6</imsss:minNormalizedMeasure>
            </imsss:primaryObjective>
         </imsss:objectives>
      </imsss:sequencing>
   </item>
   <imsss:sequencing>
      <imsss:controlMode flow = "true"/>
      <imsss:sequencingRules>
         <imsss:exitConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "completed"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "exit"/>
         </imsss:exitConditionRule>
      </imsss:sequencingRules>
      <adlseq:constrainedChoiceConsiderations  constrainChoice = "true" />
   </imsss:sequencing>
</item>
```

Code Illustration 5-23: <adlseq:constrainedChoiceConsiderations> Element

5.1.11. <rollupConsiderations> Element

The `<adlseq:rollupConsiderations>` element is the container for the descriptions of when an activity should be included in the rollup process.

XML Namespace: `http://www.adlnet.org/xsd/adlseq_v1p3`

XML Namespace Prefix: `adlseq`

XML Binding Representation: `<rollupConsiderations>`

Data Type: The `<rollupConsiderations>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<rollupConsiderations>` element contains the following elements/attributes:

Attributes:

- `requiredForSatisfied` (optional, default value = `always`): This attribute indicates the condition under which the activity is included in its parent's evaluation of a satisfied rollup rule. XML Data Type: xs:token.
- `requiredForNotSatisfied` (optional, default value = `always`): This attribute indicates the condition under which the activity is included in its parent's evaluation of a not satisfied rollup rule. XML Data Type: xs:token.
- `requiredForCompleted` (optional, default value = `always`): This attribute indicates the condition under which the activity is included in its parent's evaluation of a completed rollup rule. XML Data Type: xs:token.
- `requiredForIncomplete` (optional, default value = `always`): This attribute indicates the condition under which the activity is included in its parent's evaluation of a incomplete rollup rule. XML Data Type: xs:token.
- `measureSatisfactionIfActive` (optional, default value = `true`): This attribute indicates if the measure should be used to determine satisfaction during rollup when the activity is active. XML Data Type: `xs:boolean`.

Each of the attributes defined above can have a value of one of the following restricted tokens:

- **`always`** (default): The activity is always included in rollup rule processing.
- **`ifAttempted`**: The activity is included in the rollup processes if the activity was attempted.
- **`ifNotSkipped`**: The activity is included in the rollup processes if the activity was not skipped.
- **`ifNotSuspended`**: The activity is included in the rollup processes if the activity was not suspended.

Elements:

- None

Multiplicity: Occurs 0 or 1 time in the `<sequencing>` element.

Example:

```
<item identifier="POSTTEST3">
   <title>Module 3 -- Posttest</title>
   <item identifier="POSTTEST_QUESTION7" isvisible = "false"
         identifierref="RESOURCE_QUESTION7">
      <title>Question 7</title>
   </item>
   <item identifier="POSTTEST_QUESTION8" isvisible = "false"
         identifierref="RESOURCE_QUESTION8">
      <title>Question 8</title>
   </item>
   <item identifier="POSTTEST_QUESTION9" isvisible = "false"
         identifierref="RESOURCE_QUESTION9">
      <title>Question 9</title>
   </item>
   <imsss:sequencing>
      <imsss:controlMode choice = "false" choiceExit = "false" flow = "true"
                         forwardOnly = "true"/>
      <imsss:sequencingRules>
         <imsss:preConditionRule>
            <imsss:ruleConditions>
               <imsss:ruleCondition condition = "always"/>
            </imsss:ruleConditions>
            <imsss:ruleAction action = "hiddenFromChoice"/>
         </imsss:preConditionRule>
      </imsss:sequencingRules>
      <adlseq:rollupConsiderations measureSatisfactionIfActive = "false"
                requiredForCompleted = "ifNotSkipped" />
   </imsss:sequencing>
</item>
```

Code Illustration 5-24: <adlseq:rollupConsiderations> Element

5.1.12. <sequencingCollection> Element

There are situations where a set of sequencing information (those defined inside of a `<sequencing>` element) can be reused. The XML Binding of IMS SS defines an element, `<sequencingCollection>` that acts as a container for the set of sequencing information. The reuse happens when the `IDRef` attribute of the `<sequencing>` element references an `ID` attribute of a `<sequencing>` element that it is a child element of the `<sequencingCollection>`. If a `<sequencing>` element uses both the `IDRef` attribute and inline definition of sequencing information, any top-level element defined inline overrides any similar element defined in the referenced element.

Some characteristics and requirements of a sequencing collection that are important are:

1. If a `<sequencingCollection>` is desired, 1 and only 1 sequencing collection shall exist.
2. If a `<sequencing>` element is set up to reference a `<sequencing>` element defined in a sequencing collection, the `IDRef` attribute is mandatory and shall reference an "`ID`" specified by the `<sequencing>` element defined in the sequencing collection.
3. If a `<sequencing>` element is defined as a child of an `<item>` or `<organization>` element, then the `ID` attribute is not permitted.
4. For all `<sequencing>` elements defined in the sequencing collection, the `ID` attribute is mandatory and the `IDRef` attribute is not permitted. This requirement is in place to avoid the chaining of sequencing information (i.e., sequencing information that references other sequencing information). This requirement may be relaxed in the future; however, more use cases for the use of chaining sequencing information is required.

Attributes:

- None

Elements:

- `<sequencing>`, the sequencing element shall exist 1 or more times if a `<sequencingCollection>` is used to define a set of sequencing information.

Multiplicity: The `<sequencingCollection>` element shall exist 0 or 1 time as a child of the `<imscp:manifest>` element.

Example:

```xml
<manifest identifier="Manifest1">
   <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>2004 3rd Edition</schemaversion>
   </metadata>
   <organizations default="org1">
      <organization identifier="org1">
         <title>Organization Title</title>
         <item identifier="PRETEST1">
            <title>Module 1 -- Pretest</title>
            <item identifier="PRETEST_QUESTION1" isvisible = "false"
                  identifierref="RESOURCE_QUESTION1">
               <title>Question 1</title>
            </item>
            <item identifier="PRETEST_QUESTION2" isvisible = "false"
                  identifierref="RESOURCE_QUESTION2">
               <title>Question 2</title>
            </item>
            <imsss:sequencing IDRef = "pretest">
               <imsss:objectives>
                  <imsss:primaryObjective objectiveID = "PRIMARYOBJ"
                                          satisfiedByMeasure = "true">
                     <imsss:minNormalizedMeasure>0.6
                     </imsss:minNormalizedMeasure>
                     <imsss:mapInfo targetObjectiveID = "obj_module_1"
                                    readNormalizedMeasure = "false"
                                    writeSatisfiedStatus = "true"
                                    writeNormalizedMeasure = "true"  />
                  </imsss:primaryObjective>
               </imsss:objectives>
            </imsss:sequencing>
         </item>
      </organization>
   </organizations>
   <resources>
   <!-- A listing of resources -->
   </resources>
   <imsss:sequencingCollection>
      <imsss:sequencing ID = "pretest">
         <imsss:controlMode choice = "false" choiceExit = "false" flow = "true"
               forwardOnly = "true"/>
         <imsss:sequencingRules>
            <imsss:preConditionRule>
               <imsss:ruleConditions>
                  <imsss:ruleCondition condition = "satisfied"/>
               </imsss:ruleConditions>
               <imsss:ruleAction action = "skip"/>
            </imsss:preConditionRule>
         </imsss:sequencingRules>
      </imsss:sequencing>
   </imsss:sequencingCollection>
</manifest>
```

Code Illustration 5-25: <sequencingCollection> Element

5.2. Presentation/Navigation Information

SCORM defines presentation/navigation guidance to coincide with the IMS SS. ADL plans to continue gathering use cases and requirements from the ADL Community on presentation and navigation.

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

5.2.1. <presentation> Element

The `<presentation>` element is a container element that encapsulates presentation information for a given learning activity. This element is an ADL-defined extension to the IMS Content Packaging Specification. The element shall only appear, if needed, as a child of a leaf `<item>` element that references a resource.

XML Namespace: `http://www.adlnet.org/xsd/adlnav_v1p3`

XML Namespace Prefix: `adlnav`

XML Binding Representation: `<presentation>`

Data Type: The `<presentation>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<presentation>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<navigationInterface>`

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

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true">
      <title>Content 1</title>
      <adlnav:presentation>
         <adlnav:navigationInterface>
            <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
            <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
         </adlnav:navigationInterface>
     </adlnav:presentation>
   </item>
</organization>
```

Code Illustration 5-26: <adlnav:presentation> Element

5.2.1.1. <navigationInterface> Element

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

XML Namespace: `http://www.adlnet.org/xsd/adlnav_v1p3`

XML Namespace Prefix: `adlnav`

XML Binding Representation: `<navigationInterface>`

Data Type: The `<navigationInterface>` element is a parent element. Parent elements have no values associated with them. Parent elements act as "containers" for other elements/attributes. The `<navigationInterface>` element contains the following elements/attributes:

Attributes:

- None

Elements:

- `<hideLMSUI>`

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

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true">
      <title>Content 1</title>
      <adlnav:presentation>
         <adlnav:navigationInterface>
            <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
            <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
         </adlnav:navigationInterface>
     </adlnav:presentation>
   </item>
</organization>
```

Code Illustration 5-27: <adlnav:navigationInterface> Element

5.2.1.1.1. <hideLMSUI> Element

The `<hideLMSUI>` element indicates that the LMS should not provide user interface devices that enable the learner to trigger specific navigation events. These events are *Previous*, *Continue*, *Exit*, *Exit All*, *Abandon*, *Abandon All* and *Suspend All*.

XML Namespace: `http://www.adlnet.org/xsd/adlnav_v1p3`

XML Namespace Prefix: `adlnav`

XML Binding Representation: `<hideLMSUI>`

Data Type: The `<hideLMSUI>` element is represented as an xs:string. The string is a restricted vocabulary token. By default, if the values are not listed, then the LMS shall provide a user interface device to allow for the following navigation requests. The token shall be one of the following:

- **`previous`**: If specified, the LMS shall not display a “Previous” navigation device when this activity is the current activity.
- **`continue`**: If specified, the LMS shall not display a “Continue” navigation device when this activity is the current activity.
- **`exit`**: If specified, the LMS shall not display a “Exit” navigation device when this activity is the current activity.
- **`exitAll:`** If specified, the LMS shall not display a “Exit All” navigation device when a child of this activity cluster is the current activity
- **`abandon`**: If specified, the LMS shall not display a “Abandon” navigation device when this activity is the current activity.
- **`abandonAll`**: If specified, the LMS shall not display a “Abandon All” navigation device when a child of this activity cluster is the current activity
- **`suspendAll`**: If specified, the LMS shall not display a “Suspend All” navigation device when a child of this activity cluster is the current activity.

Attributes:

- None

Elements:

- None

Multiplicity: The `<hideLMSUI>` element shall occur 0 or More times within the `<navigationInterface>` element.

Example:

```
<organization>
   <item identifier="ITEM3" identifierref="RESOURCE3" isvisible="true">
      <title>Content 1</title>
      <adlnav:presentation>
         <adlnav:navigationInterface>
            <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
            <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
         </adlnav:navigationInterface>
     </adlnav:presentation>
   </item>
</organization>
```

Code Illustration 5-28: <adlnav:hideLMSUI> Element

5.3. Relationship to Content Packaging

The IMS Content Packaging Specification provides a structure for relating a learning activity to a content resource – the `<imscp:item>` element and its relationship to a `<imscp:resource>` element. Furthermore, `<imscp:item>` elements can be clustered into collections, with such collections contained in a parent `<imscp:organization>` element, as learning activities may be clustered together in a parent activity or activities. Therefore, IMS SS maps the concept of a learning activity to an `<imscp:item>` element, a collection of `<imscp:item>` elements within an `<imscp:organization>` element and to an `<imscp: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 `<imscp:organization>` element and each `<imscp:item>` element within it can have defined sequencing behaviors through the association of sequencing information:

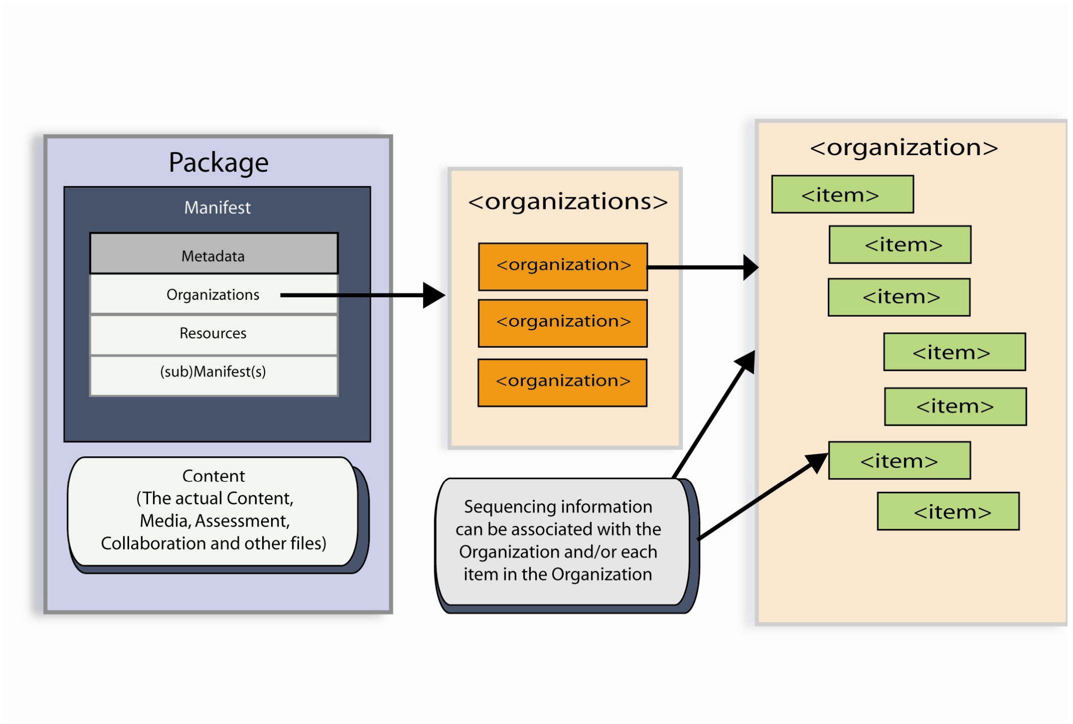


Figure 5.3a: Sequencing Rules and Content Packaging Structure Relationship

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

APPENDIX A
Acronym Listing

This page intentionally left blank.

Acronym Listing

ADL	Advanced Distributed Learning
AICC	Aviation Industry CBT Committee
API	Application Program Interface
ARIADNE	Alliance of Remote Instructional Authoring & Distribution Networks for Europe
CAM	Content Aggregation Model
CBT	Computer-Based Training
DOI	Digital Object Identifier
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
ISBN	International Standard Book Numbers
ISO	International Standards Organization
ISSN	International Standard Serial Numbers
LMS	Learning Management System
LOM	Learning Objects Metadata
LTSC	Learning Technology Standards Committee
PIF	Package Interchange File
RTE	Run-Time Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SN	Sequencing and Navigation
SPM	Smallest Permitted Maximum
SS	Simple Sequencing
URI	Universal Resource Indicator
URL	Universal Resource Locator
URN	Universal Resource Name
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition

This page intentionally left blank.

APPENDIX B
References

This page intentionally left blank.

References

1. *IEEE 1484.11.2 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication*. November 10, 2003
   Available at: http://www.ieee.org/

2. *SCORM 2004 3rd Edition Run-Time Environment Version 1.0,* Advanced Distributed Learning, October 20, 2006
   Available at: http://www.adlnet.gov/

3. *IMS Content Packaging Information Model*, *Version 1.1.4 Final Specification*. October, 2004
   Available at: http://www.imsglobal.org/

4. *Aviation Industry CBT Committee (AICC) Computer Managed Instruction Guidelines for Interoperability (CMI001) Version 3.5*. April 2, 2001
   Available at: http://www.aicc.org/

5. *IMS Simple Sequencing Behavior and Information Model* v1.0 Final Specification, IMS Global Learning Consortium, Inc., March 2003
   Available at: http://www.imsproject.org/.

6. *Extensible Markup Language (XML) 1.1* . 04 February 2004.
   Available at: http://www.w3.org/

7. *XML Base*. June 27, 2001
   Available at: http://www.w3.org/

8. *IETF RFC 3986:2005, Uniform Resource Identifier (URI): Generic Syntax*.
   Available at: http://www.ietf.org/

9. *IETF RFC 2426:1998, vCard MIME Directory Profile*

10. *ISO/IEC 10646-1:2000, Information technology—Universal multiple-octet coded character set –Part 1: Architecture and basic multilingual plane*.

11. *IEEE 1484.12.1-2002 Learning Object Metadata Standard*.
    Available at: http://www.ieee.org/

12. *IETF RFC 1951 DEFLATE Compressed Data Format Specification version 1.3*, May 1996
    Available at: http://www.ietf.org/

13. *XML Schema Part 2: Datatypes*, W3C Recommendation 02 May 2001,
    Available at: http://www.w3.org/

14. IEEE 1484.12.3 Standard for Extensible Markup Language (XML) Schema Binding for Learning Object Metadata
Available at: http://www.ieee.org/

15. *IMS Content Packaging Best Practice Guide*, Version 1.1.4 Final Specification, October 2004
Available at: http://www.imsglobal.org/

16. *SCORM 2004 3rd Edition Sequencing and Navigation Version 1.0,* Advanced Distributed Learning, October 20, 2006
Available at: http://www.adlnet.gov/

APPENDIX C
Document Revison History

This page intentionally left blank.

Document Revision History

SCORM Version	Release Date	Description of Change
1.3 Working Draft 1	22-Oct-2003	Initial draft. Changes include: • Updates to SCORM Content Packaging to include changes introduced by the IMS Content Packaging Specification Version 1.1.3. • Updates to the SCORM Metadata to include changes introduced by the standardization of IEEE 1484.12.1-2002 and IEEE 1484.12.3 Draft Standard for Extensible Markup Language (XML) Binding for Learning Object Metadata Data Model. • Updates to include IMS Simple Sequencing Version 1.0 support. • Inclusion of support for Navigation requirements due to inclusion of IMS Simple Sequencing Version 1.0.
CAM Version 1.3	30-Jan-2004	Updates include: • Clarification and changes to definitions of content aggregation vs. content organization. • Name change to SCORM Metadata Application Profiles: Package-level Metadata changed to Content Aggregation Metadata, Content Aggregation Metadata changed to Content Organization Metadata. • Updates to Section 3.3.2 Organizations to clarify definition of a content organization. • Added additional tables to Section 3.4.1 and Section 4.2 to help explain table formats.
CAM Version 1.3.1	22-Jul-2004	Updates include: • Refinements to grammar and style. • Updated Figure 2.1.3a to indicate activities and resources. • Numerous updates to the Code Illustrations to fix well-formedness issues during validation. Updated the use of the correct quotation marks for attribute values. Added missing elements that caused well-formedness errors. Changed element names that caused well-formedness errors. • Updated Section 3.4.1.6 <organizations> Element to delete the notion that the default attribute could reference an identifier of a manifest. The default attribute shall only reference <organization> elements that are direct descendants of the <organizations> element.

		• Added missing <adlcp:completionThreshold> element and the element's requirements. • Updated Section 3.4.1.23 to add more narrative text describing that the local resource must be identified using a <file> element. • Updated Figure 3.4.2.1a to change an illegal ID. The image had two item elements with the same ID (A002). • Added missing <adlcp:completionThreshold> element from Table 3.5.3a. • Removed Section 3.6.1 Packaging Multiple Courses and replaced it with Section 3.6.2 Multiple Organizations for a Single Course. More requirements gathering needs to take place on reasons for packaging multiple distinct "courses" before best practices are built. • Removed Figure 4.1a and text associated with the figure. • Changed the multiplicity requirements for the <metaMetadata> element for the Content Aggregation Application Profile from 0 or 1 to 1 and only 1. This matches the requirement defined in the SCORM 2004 Conformance Requirements Version 1.0 and the requirement defined for <metadataSchema> child element. • Changed an incorrect vocabulary reference in Section 4.2.8.1 <kind> Element from isbasisof to isbasisfor. • Updated requirements of the Content Aggregation Metadata Application Profile. The metadata is required and the only elements that are required are 3.0 Meta-Metadata and Metadata Schema. • Updated inconsistency found in Section 5.1.1 <sequencing>Element. The multiplicity was updated to 1 or more to match what was found in Section 5.1.12 <sequencingCollection> element. • Deleted sections 4.4.2.1 Source Declaration and 4.4.2.2 Vocabulary Declarations. This approach is just one way to extend LOM. Information on how to extend LOM will be discussed outside of SCORM. More guidance needs to be formulated (based on practice and IEEE work). • Changed requirement (in a Content Aggregation Content Package) for an <item> element from 0 or More to 1 or More, based on alignment with IMS Content Packaging. • Section 5.1.1: Added clarification of requirements for use of the ID and IDRef attributes. • Section 5.1.3.1.1.1: Updated to clarify that the referencedObjective attribute cannot be an empty characterstring or contain all whitespace. • Section 5.1.6.1.2: Updated a misspelled vocabulary token. The timeLimitExceeded vocabulary was missing an "i". • Section 5.1.7: Updated to indicate that the objectiveID attribute cannot be an empty characterstring or contain all whitespace. Updated to define the requirement that

		if the <primaryObjective> contains a objective map, then the objectiveID is required. Also updated text to indicate that the circumstances where the objectiveID attribute's values need to be unique. • Section 5.1.7.1.1: Updated to indicate that the LMS is responsible for using this element to initialize the cmi.scaled_passing_score RTE Data Model Element. • Section 5.1.7.1.2: Updated to indicate that the targetObjectiveID attribute cannot be an empty characterstring or contain all whitespace. • Section 5.1.7.1.2: Added more information regarding the requirements of Read Objective Maps and Write Objective Maps, specifically with multiple objective maps. • Section 5.1.7.2: Updated to indicate that the objectiveID attribute cannot be an empty characterstring or contain all whitespace. Also updated text to indicate that the circumstances where the objectiveID attribute's values need to be unique. • Updated Code Illustration 5-21 to show an example of Randomization Controls.
SCORM 2004 3rd Edition: CAM Version 1.0	20-Oct-2006	Updates included: • Updates to all images used in the document. Changes included a more common look-and-feel for the color and images used. • Changed all uses of meta-data to metadata. Removed the hyphen. • Updated Figure 3.2.a to change Physical Files to Content. • Removed language and requirements associated with (sub)manifests. New language indicates that ADL does not recommend the use of (sub)manifests until IMS completes work on the newer version of IMS Content Packaging. Various ADL Notes were added throughout the document. • Added more language describing the IMS Content Packaging recommendation that all files that are included in the content package should be declared or referenced in the manifest. • Updated the SPM for the <adlcp:dataFromLMS> element from 4096 to 4000. • Removed all language and requirements for the adlcp:persistState element. • Added language explaining that the SPM for the href attribute is the SPM for the value with the values of any xml:base applied to it. • Updated the identifierref attribute of the <dependency> element to state that the value cannot be a reference to a resource identifier in a (sub)manifest. • Added an ADL Note to Section 3.4.1.26 <manifest> Element stating that is recommended not to use (sub)manifest until IMS Content Packaging work completes.

		• Removed Section 3.4.2 (Sub)Manifests. All language and requirements to support and use (sub)manifests have been removed until further notice. • Updated Section 3.4.3.1 Handling the XML Base Attribute to include language about special care to the SPM defined in IMS for the href and xml:base attributes. • Updated Section 3.4.3.3 Handling the Parameters Attribute to include an update to the normative parameter construction algorithm. • Updated Table 3.5.3a to remove the <adlcp:persistState> element. • Updated Table 3.5.3a to change the requirements for items 1.4 <metadata>, 1.4.1 <schema>, 1.4.2 <schemaversion> from O (optional) to M (mandatory). The values in this table did not match those defined earlier. • Deleted Section 3.6.2 Packaging Learning Content for Reuse. This section dealt with using (sub)manifest to accomplish this goal. This was removed based on the other changes due to (sub)manifest updates. • Updated Section 4.2.1 <lom> Element to describe how to have multiple metadata instances describing the same SCORM Content Model Component. • Clarified that the <language> element (LOM.General.Language) can have a value of `none`, if the content has no lingual content. • Updated all metadata code illustrations that use VCard syntax. Removed the \n character in favor of the required &#13; and &#10; values. • Updated Section 4.2.4.2 <contribute> Element to fix the SPM value. Changed from 30 to 10 to match the IEEE XML Binding of LOM. • Updated Section 4.2.4.2 .2 <entity> Element to fix the SPM value. Changed from 40 to 10 to match the IEEE XML Binding of LOM. • Updated Section 4.2.4.4 <language> Element to add more clarification on the format of the value held by the language element. • Updated Section 4.2.8.2 <resource> Element to change the SPM from 0 or More to 0 or 1, to align with the IEEE XML Binding of LOM. • Updated Section 4.2.8.2.2 <description> Element to change the SPM from 0 or 1 to 0 or More (SPM: 10), to align with the IEEE XML Binding of LOM. • Updated Section 5.1 Sequencing and Presentation to remove that language dealing with sequencing information being placed on <item> elements that reference (sub)manifests. This was removed based on the other changes due to (sub)manifest updates. • Updated Section 5.1.7.1.1 <minNormalizedMeasure> Element to describe that an LMS shall use this value to initialize the cmi.scaled_passing_score data model element. • Updated Section 5.1.11 <rollupConsiderations>

		Element to change the default value of the measureSatisfactionIfActive attribute from false to true. • Updated Section 5.2.1 <presentation> Element to change SCO to resource. This covers the case that the resource is an Asset or SCO. • Updated the language found for the tokens used in the <hideLMSUI> element's value space. • Updated Section 5.2.1.1.1 <hideLMSUI> Element to include a new token value (suspendAll, abandonAll and exitAll). • Updated version of the document to define this version as SCORM 2004 3rd Edition. Also updated the internal version of this document to Version 1.0 • Updated, where necessary, throughout this document any information relevant to the change to SCORM 2004 3rd Edition. • Updated contact information for the ADL Co-Lab Hub • Editorial updates throughout the document (e.g., applied the SCORM registered trademark consistently throughout). • General updates to reflect changes made in other SCORM documents (i.e., Run-Time Environment and Sequencing and Navigation books). • Updated all relevant references to adlnet.org to adlnet.gov. • Updated the SCORM Content Model Components to separate out the 5 components into 5 sections. Several were combined into one. The 5 components and their respective sections are: Assets, SCOs, Activities, Content Organizations and Content Aggregations. • In support of the separation of the SCORM Content Model Components into separate sections, added Figure 2.1.3a and Figure 2.1.4a • Updated language for the identifier attribute of the <manifest>, <organization>, <item>, <resource> and <sequencing> elements to define that these identifiers shall be unique within the scope of the Manifest, regardless of case (i.e., identifiers are not case sensitive). • Changed the requirements for the use of the type attribute on a <resource> element. Changed the requirement so that the type attribute value should be set to webcontent for those <item> elements that reference <resource> elements. Prior to this change the type attribute had to be set to webcontent in order to be conformant. This requirement has been changed to a general practice and is no longer needed for conformance to SCORM. • Added language to the href attribute of a <resource> element to indicate that the URL may also contain additional parameters. • Updated the description of the <file> element for clarity. • Changed the title of Section 3.4.3.3 to Handling

		Additional Parameters as Part of URLs. • Removed all requirements for SCORM Metadata. The SCORM still recommends the use of IEEE LOM for defining metadata, defines SCORM Content Model Component metadata and describes ways of associating metadata to SCORM Content Model Components defined in a Manifest. However, the SCORM now references ADL Registry for metadata requirements associated with U.S. DoD requirements as defined in DoD Instruction 1322.26, Development, Management, and Delivery of Distance Learning, 6/16/2006. ADL also encourages other communities of practice to refer to ADL Registry for ways of implementing metadata requirements defined for particular communities of practice. • For those LOM elements that were mandatory for use in metadata instances describing SCORM Content Model Components within a Content Package Manifest, these requirements have now been relaxed to optional. No more mandatory requirements exist. For those that are required to adhere to U.S. DoD Instruction 1322.26, all requirements are now defined by the ADL Registry Metadata requirements. • Removed all notion of SCORM Best Practice and Restricted Vocabularies along with other SCORM relevant best practices with metadata in general. • Removed all notions of default values for sequencing elements where the XSDs did not define any. Elements/Attributes affected: referencedObjeective, measureThreshold, condition, attemptLimit, attemptAbsoluteDurationLimit and selectCount. • Updated Code Illustration 5-24 to place the <adlseq:rollupConsiderations> element in the appropriate location. • Changed the requirement for the value of the <schemaversion> element of a content package manifest. The value is now required to be 2004 3rd Edition. This value is used to indicate what version of SCORM that the content package is built in accordance with. • Added language to the <file> Element to clarify that for files that are local to a content package the <file> element's href attribute should be identical to the containing <resource> element's href attribute minus any parameters that may have been defined on the <resource> element's href attribute value. • Added language to indicate that some systems may treat href values as case sensitive and developers/tool makers should be aware of this when defining href values. • Updated language to map the requirements defined for the multiplicity of the <ruleConditions> element to that defined in the IMS provided XSD. The <ruleConditions> element shall be defined 1 and only 1 time within a <preConditionRule>,

		<postConditionRule> or <exitConditionRule> element. The requirement use to say 0 or 1. If one of these rules are going to be provided then a condition must be provided also. • Updated the language used for the referencedObjective attribute to indicate that if a value is provided then this value shall match an objectiveID of either a <primaryObjective> or one of the <objective> elements defined for the activity. • Updated language to map the requirements defined for the multiplicity of the <ruleCondition> element to that defined in the IMS provided XSD. The <ruleCondition> element shall be defined 1 or More times within a <ruleConditions> element. The requirement use to say 0 or More. If a <preConditionRule>, <postConditionRule> or an <exitConditionRule> is being provided then a condition must be provided also. • Changed the reference to IETF RFC 2396 to IETF RFC 3986 (newer version).