



# **Business Rules, Best Practices, and Examples for Army SCORM 2004 3<sup>rd</sup> Edition Conformant Courseware**

**Version 2.0**  
25 January 2012

757-878-5668  
DSN 826-5668  
SCORM@atsc.army.mil

## SUMMARY of CHANGE

BRBP Version	Release Date	Description of Change
1.7	02 September 2009	Update version
2.0	25 January 2012	Focus on SCORM

# Business Rules, Best Practices, and Examples for Army SCORM 2004 Conformant Courseware

<b>1. EXECUTIVE SUMMARY.....</b>	<b>7</b>
1.1 NOTATIONS USED .....	8
1.2 LIST OF ARMY BUSINESS RULES: .....	9
<b>2. INSTRUCTIONAL DESIGN.....</b>	<b>14</b>
2.1 SCORM CONCEPTS AND PROCESSES .....	14
2.1.1 Reusable Learning Objects (RLO).....	14
2.1.2 The "Page".....	14
2.1.3 SCORM Content Model Components .....	14
2.1.4 Army Classification of SCOs .....	16
2.2 COURSE MAP, LESSON FLOW DIAGRAM, AND CONTENT ORGANIZATION .....	17
2.2.1 Course Map .....	17
2.2.2 Content Organization .....	19
2.2.3 Lesson Flow Diagram.....	21
<b>3. CONTENT DESIGN SPECIFICS.....</b>	<b>23</b>
3.1.1 SCO Acronym .....	23
3.1.2 SCO Titles.....	24
3.1.3 Linking to Additional Learning Resources .....	26
3.2 LEARNING OBJECTIVES AND GLOBAL OBJECTIVES .....	26
3.3 COURSE TITLE.....	27
3.4 FILE NAMING CONVENTION .....	27
3.5 NAVIGATION .....	30
3.5.1 Navigation Help Feature .....	30
3.5.2 Intra-SCO or simple Navigation.....	30
3.6 TABLE OF CONTENTS (TOC).....	31
3.7 FOLDER STRUCTURE EXAMPLE.....	32
3.8 CHECKS-ON-LEARNING.....	32
3.9 COURSE MAP .....	33
3.10 LAUNCHABLE ASSETS .....	34
3.11 DESIGNING FOR REUSABILITY .....	34
3.12 EXTERNAL REFERENCES .....	34
<b>4. SEQUENCING AND NAVIGATION.....</b>	<b>37</b>
4.1 CLUSTER/LEAF.....	37
4.2 SEQUENCING STRATEGY .....	39
4.2.1 Rule-Based Sequencing.....	40
4.2.2 Global Objectives .....	41
4.2.3 Rollup Rules.....	43
4.2.3.1 Rollup Considerations.....	44
4.2.4 Attempt Limits .....	45
4.2.5 Randomization .....	45
4.2.6 Constrain Learner Choices.....	46
4.3 NAVIGATION STRATEGY .....	46
4.3.1 Control Mode.....	47
4.3.1.1 Choice Control Mode.....	47
4.3.1.2 Flow Control Mode.....	47
4.3.1.3 Forward Only Control Mode.....	49
4.3.2 Customization .....	49
4.3.3 LMS User Interface (UI) Buttons for Navigation .....	50
4.3.3.1 Hiding the LMS User Interface (UI) Buttons.....	51

4.4	NAVIGATION REQUESTS.....	53
<b>5.</b>	<b>COMMUNICATION WITH THE LMS (RUN-TIME ENVIRONMENT).....</b>	<b>54</b>
5.1	BEGIN COMMUNICATION .....	54
5.2	END COMMUNICATION.....	54
5.2.1	Retrieving Data.....	54
5.2.2	Storing Data.....	55
5.3	SCORM DATA MODEL.....	55
5.3.1	Bookmarking.....	55
5.3.2	Learner Entry.....	55
5.3.3	SCO Status.....	56
5.3.4	Learner Score .....	57
5.3.5	Mastery Score .....	57
5.3.6	Exit Status .....	57
5.3.7	Session Time .....	57
5.3.8	Total Time.....	58
5.3.9	Time Limit and Resulting Action.....	58
5.3.10	Data Storage Area .....	58
5.3.11	Test Item Data.....	59
<b>6.</b>	<b>METADATA .....</b>	<b>60</b>
6.1	ARMY METADATA REQUIREMENTS .....	60
6.1.1	Content Organization Metadata .....	61
6.1.2	SCO Metadata .....	61
6.2	DEVELOPING THE SCORM METADATA FILES .....	61
6.2.1	Catalog Identifier and Entry Identifier .....	61
6.2.2	Title of Learning Resource .....	61
6.2.3	Language of Learning Resource.....	61
6.2.4	Description of Learning Resource .....	61
6.2.5	Keywords .....	62
6.2.6	Type of Metadata .....	62
6.2.7	Version of Learning Resource .....	62
6.2.8	Status of Package Submittal.....	62
6.2.9	Proponent's Role.....	62
6.2.10	Name, Address, and E-mail of Proponent.....	62
6.2.11	Date of Submittal .....	62
6.2.12	Metadata Catalog and Entry Identifier.....	63
6.2.13	Metadata Schema.....	63
6.2.14	Language of Metadata File.....	63
6.2.15	File Formats.....	63
6.2.16	Cost of Learning Resource.....	63
6.2.17	Copyright and Other Restrictions .....	63
6.2.18	Classification .....	64
<b>7.</b>	<b>ARMY SCORM AND PLAYABILITY TESTING OF COURSEWARE .....</b>	<b>67</b>
7.1	SCORM REFERENCES AND TUTORIALS/ONLINE COURSES .....	67
7.2	SCORM TOOLS .....	67
<b>8.</b>	<b>GLOSSARY AND ACRONYMS.....</b>	<b>68</b>
<b>9.</b>	<b>PROGRAMMING EXAMPLES .....</b>	<b>76</b>
9.1	XML EXAMPLES ON THE MANIFEST .....	77
9.1.1	Course Title Example.....	77
9.1.2	SCO Example.....	78
9.1.3	Cluster Example.....	78
9.1.4	Launchable Asset Example .....	79

9.1.5	<i>ADL Extensions to the Manifest File</i> .....	80
9.1.6	<i>Screenshot of Manifest File</i> .....	81
9.1.7	<i>Manifest in Detail</i> .....	82
9.2	SEQUENCING AND NAVIGATION EXAMPLES .....	86
9.2.1	<i>Sequencing Overview</i> .....	86
9.2.2	<i>Control Mode Example</i> .....	87
9.2.2.1	Choice .....	88
9.2.2.2	Flow .....	89
9.2.2.3	Forward Only .....	89
9.2.2.4	Choice Exit .....	90
9.2.2.5	Constrained Choice Considerations .....	90
9.2.3	<i>Sequencing Impact on Graded Assessments</i> .....	91
9.2.3.1	Scaled Passing Score Example .....	91
9.2.3.2	Disable Graded Assessment When Completed Example .....	92
9.2.4	<i>Rule-Based Sequencing</i> .....	92
9.2.5	<i>Rollup Rules Examples</i> .....	93
9.2.5.1	Rollup Considerations Example .....	95
9.2.6	<i>Randomization Example</i> .....	95
9.2.7	<i>Delivery Controls Examples</i> .....	96
9.2.7.1	Tracked .....	96
9.2.7.2	Completion Set by Content .....	96
9.2.7.3	Objective Set By Content .....	96
9.2.8	<i>Hiding the LMS User Interface (UI) Buttons Example</i> .....	97
9.3	IMSSS:MAPINFO – SHARED DATA EXAMPLE .....	97
9.4	SEQUENCING COLLECTION EXAMPLE .....	99
9.5	COMMUNICATION WITH THE LMS (RUN-TIME ENVIRONMENT) EXAMPLES .....	99
9.5.1	<i>Begin Communication</i> .....	101
9.5.2	<i>End Communication</i> .....	102
9.6	DATA TRANSFER .....	107
9.6.1	<i>Retrieving Data</i> .....	107
9.6.2	<i>Storing Data</i> .....	107
9.6.3	<i>Commit</i> .....	107
9.7	STATE MANAGEMENT .....	108
9.7.1	<i>GetLastError</i> .....	108
9.8	MACROMEDIA FLASH SPECIFIC RUN-TIME EXAMPLE .....	108
9.9	BOOKMARKING EXAMPLE .....	109
9.10	LEARNER SCORING AND SUCCESS STATUS .....	111
9.10.1	<i>Mastery Score Example</i> .....	111
9.10.1.1	Initializing Learner's Success Status .....	112
9.10.2	<i>Confusing Success Status with Completion Status</i> .....	113
9.11	COMPLETION STATUS EXAMPLE .....	114
9.12	EXIT STATUS EXAMPLE .....	115
9.13	SESSION TIME EXAMPLE .....	115
9.14	TOTAL TIME EXAMPLE .....	117
9.15	DATA STORAGE AREA EXAMPLE .....	118
9.16	INTERACTIONS .....	118
9.16.1	<i>Overview</i> .....	118
9.16.2	<i>SCORM Required Interaction Data Fields</i> .....	120
9.16.3	<i>Army SCORM Required Interaction Data Fields</i> .....	121
9.16.4	<i>Not Required Interaction Data Fields</i> .....	125
9.16.5	<i>Test Item Data Collection Example</i> .....	125
9.17	COMMENTS FROM THE LEARNER EXAMPLE .....	126
9.18	EXTERNAL REFERENCES EXAMPLE .....	127
9.19	LEARNER DIRECTIONS PAGE .....	129
9.20	NAVIGATION REQUESTS EXAMPLES .....	130
9.20.1	<i>Navigation Request According to the Activity Tree</i> .....	130
9.21	GRADED ASSESSMENTS EXAMPLES .....	131

9.21.1	<i>Non-Resumable Learner Performance Test</i> .....	131
9.21.2	<i>Timed Non-Resumable Learner Performance Test</i> .....	133
9.21.2.1	Maximum Time Allowed (Time Limit) Example .....	134
9.21.2.2	Time Limit Action Example .....	135
9.21.3	<i>Resumed Learner Performance Test</i> .....	137
9.22	DIFFERENCES IN SCORM v1.2 DATA MODEL AND THE SCORM 2004 DATA MODEL .....	139
9.23	DATA TYPES FOR FORMATTING VALUES.....	141
9.24	METADATA EXAMPLES .....	142
9.24.1	<i>XML Binding for Separate Metadata Files</i> .....	142
9.24.2	<i>Relative Path Example</i> .....	143
9.24.3	<i>Content Organization Metadata Referenced on the Manifest</i> .....	143
9.24.4	<i>SCO Metadata Referenced on the Manifest</i> .....	144
9.24.5	<i>Impact of File Path Offset Option on Relative Paths in the Manifest</i> .....	144
9.25	DEVELOPING THE SCORM METADATA FILES.....	146
9.25.1	<i>Catalog and Entry Identifier</i> .....	147
9.25.2	<i>Title of Learning Resource</i> .....	148
9.25.3	<i>Language of Learning Resource</i> .....	148
9.25.4	<i>Description of Learning Resource</i> .....	149
9.25.5	<i>Keywords</i> .....	150
9.25.6	<i>Type of Metadata</i> .....	151
9.25.7	<i>Version of Learning Resource</i> .....	151
9.25.8	<i>Status of Package Submittal</i> .....	152
9.25.9	<i>Proponent's Role</i> .....	152
9.25.10	<i>Proponent's Name and Address</i> .....	153
9.25.11	<i>Date of Submittal</i> .....	153
9.25.12	<i>Meta-Metadata Catalog and Entry Identifier</i> .....	154
9.25.13	<i>Meta-Metadata Schema</i> .....	154
9.25.14	<i>Language of the Metadata File</i> .....	155
9.25.15	<i>File Formats</i> .....	155
9.25.16	<i>Cost of Learning Resource</i> .....	156
9.25.17	<i>Copyright and Other Restrictions</i> .....	156
9.25.18	<i>Classification</i> .....	157
9.26	SETTING UP THE SCORM METADATA SCHEMAS .....	160
9.27	PACKAGING AND DELIVERY .....	161
9.27.1	<i>Creating the Manifest File</i> .....	163
9.27.2	<i>How to Create a Content Package per the SCORM Implementation Guide (<a href="http://www.adlnet.gov">http://www.adlnet.gov</a>)</i> .....	164
9.27.3	<i>Final Courseware Packaging and Delivery Requirements</i> .....	166
<b>10.</b>	<b>TEST ITEM DATA (INTERACTIONS) TABLE.....</b>	<b>167</b>
<b>11.</b>	<b>METADATA TABLES.....</b>	<b>169</b>

# 1. Executive Summary

This document is intended for use by personnel (both government and contractor) involved in the design, development, and programming of all Sharable Content Object Reference Model (SCORM) 2004 conformant Courseware. Programmers from contractors and/or proponent agencies designing and developing dL courseware must thoroughly understand this document.

These Business Rules and Best Practices apply to any Army SCORM 2004 conformant courseware that will be served on a SCORM 2004 conformant Learning Management System (LMS). All Business Rules must be adhered to when developing above referenced courseware. Best Practices are included to assist in the development of SCORM 2004 conformant courseware and to aid in consistency.

Note: At the time of release of this document, the ADL Technical Working Group (TWG) acknowledges that the SCORM certification process for LMSs is inadequate to support interoperability of content between different LMSs. There is no guarantee that a course will behave in the same way across different ADL-certified LMSs. Therefore, your course should be tested in your target LMS.

There are some significant differences between the capabilities of SCORM v1.2 and SCORM 2004. One difference is the addition of sequencing and navigation in SCORM 2004. SCORM 2004 allows sequencing to be defined by the developer, thereby controlling the order in which the learner navigates through the courseware. SCORM v1.2 allowed the learner to freely navigate wherever he desired. Another difference is in lesson status (SCORM v1.2). In SCORM 2004, it is measured by both completion status and success status. Completion status only indicates whether the learner has completed a Sharable Content Object (SCO). Success status indicates whether the learner has mastered the SCO.



**Best Practice: Whether a student earns a passing score on a test, or simply satisfies viewing requirements, the developer should set success AND completion. In practice, for both function and form, it is advisable to set success and completion in unison. Functionally, setting both progress measures assists the LMS's timely and precise application of SCORM defaults while processing rollup rules. Relative to form, users seldom distinguish levels of completeness and will intuitively process a single electronic goal attained indicator. Finally in Army instructional design, single content items will never require a management system's independent processing of learner progress indicators.**

- **SCORM Version:** All references to SCORM will refer to the specific components of the SCORM specifications suite found on [ADL's download page](#).

## 1.1 Notations Used

The following notations are used throughout the document and explained below:

-  **Business Rule (Army):** An Army mandatory requirement for implementing the SCORM specification. These requirements must be met for SCORM 2004 courseware to be accepted by the Army.
-  **Best Practice:** Recommended SCORM 2004 practice for what works best based on lessons learned. Alternative approaches are acceptable as long as they comply with SCORM 2004 and Army SCORM 2004 requirements and can be implemented in the LMS appropriate for the course.
-  **ADL Reference:** References ADL SCORM 2004 2<sup>nd</sup> and 3<sup>rd</sup> Edition Documentation Suites (or the latest edition supported by the target LMS(s) at time of contract award) as presented by Advanced Distributed Learning Initiative. SCORM 2004 3rd Edition Documentation Suite can be found on the ADL Web site at <http://www.adlnet.gov/capabilities/scorm/scorm-2004-3rd#tab-resources>.
-  **ADL 3<sup>rd</sup> Edition Note:** Information specifically related to SCORM 3<sup>rd</sup> Edition.
-  **To Be Developed (TBD):** Certain portions of this document, or material referenced in this document, are still in development. These sections will be updated in future iterations of this document.
- **PROGRAMMER INFO:** Programming information separated from instructional design information.

This document provides Army Business Rules, Best Practices, definitions, explanations, and sample code for developing SCORM conformant courseware produced under the dL Program. **The Army Business Rules are mandatory requirements for Army SCORM courseware.** To promote interoperability and courseware consistency, the Best Practices defined in this document should be used to the maximum extent possible.

Changes to this document are anticipated to coincide with updates to the SCORM specification as well as through identification of additional lessons learned.

For SCORM technical support or to submit comments or suggestions for this document, send an e-mail to [scorm@army.mil](mailto:scorm@army.mil) or call 757-878-5668 (DSN 826-5668). For information about how to upload files to ATSC or accessing courseware on the Army developmental Saba server or Blackboard server, send e-mail to [Technical Standards and Specification Team](mailto:scorm@army.mil) or call 757-878-5668 (DSN 826-5668).

## **1.2 List of Army Business Rules:**

**Important Instructional Design note:** The same re-use dialogue presented for SCO information is presentable for any learning objects or chunks. Within the Business Rules words will be written out for “content”, however in the body of this document it will be noted that SCO will represent learning objects and chunks in general and not just SCORM learning objects. This position reflects the recognition that any learning object or chunk can be arbitrarily sized identically to SCOs for content that is not SCORM compliant. In other words, reusability traits can be duplicated for non-tracked content hosted on any content hosting platform when SCORM compliance is not required.

Section Title	Business Rule
All about SCOs	<p><b>2.1.3.4-1 Business Rule (Army SCORM):</b> Independent SCOs shall be able to stand alone as self-contained blocks of learning content. The learning content of an independent SCO shall not contain references (for example, hyperlinks, narration, etc.) to another SCO or aggregation. An independent SCO shall not contain references to a course map or transitional statements such as "The next lesson will explain...", etc.</p> <p>- OR -</p> <p><b>2.1.3.4-1 Business Rule (Army Content):</b> Independent and portable Chunks shall be able to stand alone as self-contained blocks of learning content. The learning content of an independent and portable chunk shall not contain references (for example, hyperlinks, narration, etc.) to other chunks or instructional material within the same *course*. An independent and portable chunk shall not contain references to a course map or transitional statements such as: The next lesson will explain..., etc.</p> <p><b>3.1.4-1 Business Rule (Army SCORM):</b> The SCO title contained within the SCO displayed to the learner must be the same as the SCO title that is displayed on the LMS Table of Contents.</p> <p>- OR -</p> <p><b>3.1.4-1 Business Rule (Army Content):</b> The topic title presented to the learner in instructional presentation must be the same as the Linked title that is displayed on the Table of Contents (or Menu).</p> <p><b>3.1.4-2 Business Rule (Army SCORM):</b> Hierarchical higher level instruction titles or words (course, phase, module, lesson, etc.) must not be contained within independent SCO content or titles. If an independent SCO contains an ELO, it must not reference its TLO within the content or title of the SCO. Independent SCO titles must not include the MOS, Skill Level, SQI, ASI, or SCO acronym within the actual SCO, navigation links, or the manifest. Independent SCO content and titles must not include sequencing numbers.</p>

Section Title	Business Rule
<p>All about SCOs (cont)</p> <p>File Naming Convention</p>	<p><b>3.1.5-1 Business Rule (Army SCORM):</b> Learning content files for all SCOs must be physically located within its SCORM content package (within the root folder or subfolder) and not referenced by an URL.</p> <p><b>9.20-1 Business Rule (Army SCORM):</b> A SCO cannot hyperlink to another SCO.</p> <p><b>3.4-1 Business Rule (Army Content):</b> Mandatory URL, file, and folder format and size:</p> <ul style="list-style-type: none"> <li>▪ A three character extension is recommended. A single period shall be used to separate the file name from its extension. Folder names shall not contain a period.</li> <li>▪ The path separator shall be either '/' or '\' depending upon the operating system (Microsoft® Windows, UNIX®) because the browser will resolve the correct path using the appropriate separator.</li> <li>▪ Long file names are allowed but must conform to ISO 9660 with Microsoft Joliet extension and the Army's file naming scheme. Joliet allows filenames up to 64 characters in length. Joliet allows spaces but the Army's naming scheme does not.</li> <li>▪ The Army's file naming scheme is extended to Government Furnished Information (GFI) for all files internal to the SCORM content package.</li> </ul> <p><b>3.4-2 Business Rule (Army Content):</b> Additional characters that shall not be used in Web-based courseware for naming files/folders: The following characters cause problems in a UNIX® Operating System environment:    ; , ! @ # \$ ( ) &lt; &gt; / \ " ' ` ~ { } [ ] = + &amp; ^ &lt;space&gt; &lt;tab&gt;</p> <p><b>3.4-3 Business Rule (Army Content):</b> All Uniform Resource Locators (URLs) within training content with unsafe or reserved characters shall be replaced with their hexadecimal equivalent.</p>
<p>Navigation Help Feature</p>	<p><b>3.5.1-1 Business Rule (Army SCORM):</b> A Navigation Help feature must be available at all times. It must explain the function of all internal navigation buttons available to the learner because the behavior of each button and button label is not determined by SCORM. If the LMS user interface buttons are enabled at points throughout the learning experience, the navigation help shall refer the learner to the LMS for further explanation of the LMS navigation features.</p>

Section Title	Business Rule
Bookmarking	<p><b>5.3.1-1 Business Rule (Army SCORM):</b>            SCOs shall provide a learner with the ability to bookmark their progress in a SCO whenever the learner leaves the SCO prematurely. The SCO must allow the learner the option of resuming their progress in that SCO from the bookmarked location. This bookmarking capability is not required for any proponent allowed exceptions; for example, with assessments, summaries, introductions, etc.</p>
Global Objectives	<p><b>4.1.2-1 Business Rule (Army SCORM):</b> The scope of all manifest global objectives shall be the manifest itself. Manifest global objectives will be used for sharing the sets of Objective Progress Information.</p>
Test Item Data (Interactions)	<p><b>9.16-1 Business Rule (Army SCORM):</b> Developers of SCO assessments must make use of the SCORM Interactions Data Model Element to record information about the learner's response for validation purposes. Usage of the Interactions Data Model Element must conform to <a href="#">Figure 9.16.1a</a>.</p>
Communication with the LMS	<p><b>9.5.2-1 Business Rule (Army SCORM):</b> Whenever the learner leaves a SCO, (for example a SCO navigation request, LMS navigation button, closing the browser window, pushing an 'Exit' button on the SCO page, etc.) the SCO must terminate communication with the LMS.</p> <p><b>9.10.1-1 Business Rule (Army SCORM):</b> Graded assessment SCOs shall ensure that an appropriate value for the success status, either 'passed' or 'failed' is set in the LMS. For graded assessment SCOs that pass a score to the LMS, a corresponding <code>&lt;imss:minNormalizedMeasure&gt;</code> must be delineated on the manifest. These SCOs shall make evaluations of the success status based upon the 'scaled_passing_score'.</p> <p><b>9.11-1 Business Rule (Army SCORM):</b> Once a learner receives credit for a SCO or course, the courseware shall not change that status for any reason.</p>

Section Title	Business Rule
Metadata	<p><b>6.1-1 Business Rule (Army SCORM):</b> Metadata is required for ALL Content Packages and Resource Packages, all manifest Organization level items including each SCO and all launchable assets. Non-launchable Assets and resource aggregations must have metadata only upon the government’s request. All metadata must be formatted in accordance with the instructions in the Metadata Section and <a href="#">Figure 9.25b</a>.</p> <p><b>9.24.1-1 Business Rule (Army Content):</b> Metadata must be in separate XML files and referenced from within the manifest file using the XML binding for separate metadata file references.</p> <p><b>9.26-1 Business Rule (Army Content):</b> The metadata schemas that must reside in the root folder must also be located in the folder(s) where the metadata resides. If a folder contains metadata, then the metadata schemas must be copied into this folder.</p>
Packaging and Delivery	<p><b>9.27-1 Business Rule (Army SCORM):</b> All SCOs (content and graded assessments) must be contained in a content package. SCORM packages must contain a manifest (imsmanifest.xml) file and all of the SCORM and extension schemas in the root of the package. All physical files required for the courseware must be referenced locally and contained within the content package and disclosed on the imsmanifest.xml file. Metadata is supplied as separate files and these files must be contained within the content package and disclosed on the manifest file.</p> <p><b>9.27.2-1 Business Rule (Army Content):</b> The Government requires a “Gold Copy” of all elearning content to be delivered using SCORM content packaging guidelines. The “Gold Copy” requirement includes all file based content that is consumable by the learner, Furthermore the files will be packaged into logical bundles and delivered as a PIF files. Other contract deliverables (for example, original source files, SCORM testing log files, answer keys, loading instructions, ALO XML file, etc.) must be delivered as separate files external to the courseware PIF file(s).</p>

Figure 1.1a

## **2. Instructional Design**

Following SCORM specifications and loading instructional content into an LMS will have limited repercussions in the Instructional Design of the material. This section will begin explanations that will help minimize influences and insure that the learner experience will not be degraded simply because of the specifications necessary for SCORM compliance.

### **2.1 SCORM Concepts and Processes**

Key concepts and processes are introduced in this section to assist the reader in understanding this document.

#### **2.1.1 Reusable Learning Objects (RLO)**

Learning content is generally produced as courses, modules, or lessons each full of resources such as animations, simulations, graphics, video, and audio. New technical standards were developed to open these resources to discovery and reuse. These standards allow for separation of the all the pieces of learning content making it available to other course developers. Other benefits include:

- Use existing content to create new courses for all government
- Provide doctrinally correct instruction that can be branded by other groups
- Utilize multiple delivery channels (Internet, intranet, print, and more)
- Provide for efficient and cost effective content revisions by updating content
- Improve course development time and efficiency
- Assemble new courses and other deliverables from existing content, in whole or in part

RLOs are small chunks of learning content that can be tagged with descriptive labels (metadata) and made discoverable by other developers.

#### **2.1.2 The "Page"**

The page concept has been used in computer based training for a number of years. Most authoring tools use the metaphor of a book as a basis for design layout. The learner has the look and feel of content displayed on a page with buttons to allow movement from page to page. This metaphor is repeated in both SCORM and non-SCORM Web-based training. SCOs and web-based chunks are routinely delivered as HTML pages. With intra-SCO navigation, a single HTML page can appear to be more than one page to the learner and within a web-based chunk a single HTML page can appear to be more that one page to the learner.

#### **2.1.3 SCORM Content Model Components**

The SCORM Content Model describes the components used to build a learning experience from learning resources. The Content Model also defines how these lower-level sharable, learning

resources are pulled together (aggregated) into higher-level units of instruction. The SCORM Content Model is made up of Assets, Sharable Content Objects (SCOs), and Content Organization.

Shareable Content Objects (SCOs) and assets are two types of Reusable Learning Objects used in SCORM.

- **Assets** – SCOs are made up of assets. An Asset can be as small as an individual graphic or as large as a set of web pages or a simulation or practical exercise. For example, a lesson (SCO) might contain a simulation (asset) and a glossary (asset) as well as a check on learning (asset). All of the graphics are also assets and are part of the lesson.

From a design perspective, an asset is any object that should be created for potential reuse. This will make it easier to build, maintain, and update content that contains the asset.

Assets can be effectively managed using content management or learning content management systems (LCMS). Identification of assets with appropriate metadata assists in their discovery and reuse. This metadata may be as simple as an identifying label, or it may include information on educational strategy, technical requirements, and digital rights. Assets are an important type of reusable learning object that plays a key role in the production process. Assets can be as large as a SCO and if desirable an asset launch instruction can be built into the Table of Contents identical to a SCO. Note that even though Launchable assets are launched from the LMS generated TOC, SCORM places no communication requirements on launchable assets and therefore launchable assets will not be tracked and launchable assets will not contribute to learner completion or mastery.

- **SCOs** – Sharable Content Objects are Reusable Learning Objects that can be launched and tracked by communicating with a learning management system (LMS). It is this communication with an LMS that makes a SCO a SCO. Without this, a SCO is just a collection of assets. SCOs should be developed to support specific instructional objectives. An entire course could be a SCO if there is no reason to break it into smaller pieces.

Learning content may be broken down into pieces to make it more reusable. For example, an assessment might be a SCO if it communicates with the LMS. If it needs to be part of a larger piece of content, then it might best be treated as an asset within a SCO and would not then need to communicate with the LMS; as its parent (the SCO) will perform that function.

There are limitations inherent in SCORM that can influence design and production decisions. SCORM specifications allow a SCO to communicate with an LMS but not with another SCO. This limitation supports the discovery and reuse of a SCO.

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. This restriction will sometimes result in the duplication of content (Special table of calculations needed in two objects) and sometimes result in the creation of a reusable asset (glossary or reference reading list)

Another important consideration for the instructional design team is the concept of dependence, both for a SCO, and for learning objects or chunks in general. To improve reusability, SCOs and chunks should be designed independent of learning context. To attain this freedom from context, SCOs are usually formed from the smallest part of learning content; for example, an animation depicting electrical troubleshooting instead of the complete lesson in electrical diagnostics. The complete lesson may have references too specific for other content developers to use. A SCO is considered dependent if the graphic design elements or the sequence and navigation design are tied to a specific context or content.

#### **2.1.4 Army Classification of SCOs**

The Army supports different classifications or purposes of SCOs relating to reusability. The same dialogue could be presented with respect to web-based non-SCORM learning objects or chunks but in this section readers must interpret references to SCOs as references to web-based content objects in general. Independent SCOs are intended to train the task without reference to a context (for example, MOS title); whereas dependent SCOs have fewer restrictions and can relate to context or provide a transition between tasks. Certain rules may apply to one type of SCO and not another.

Learners can access assets, such as remediation, help, or glossary, or references, and then return to his previous location in a SCO. Refer to the [Navigation Requests](#) section (4.3).

- Independent SCOs – are "independent," meaning those SCOs that are context-neutral and contain learning content and/or graded assessments. Context-neutral implies the absence of the MOS and removal of the course, module, and lesson titles. Army structure terms like course, phase, module, lesson, etc., are also removed. Independent SCOs are considered "reusable" and could be stored in a SCO repository to be available to other training developers for use in another course(s) with little or no modification required.

The independent SCO may refer to non-testable reference material via a URL. Refer to the [External References](#) section (3.12).

- The Dependent SCOs – are developed to establish the context of an otherwise context-neutral group of SCOs such as a Course or Module Introduction or a Course Overview SCO. Also included would be SCOs that would transition the learner between content. For example, a dependent SCO could be developed to transition from an introduction SCO to an independent SCO, or between two independent SCOs. Dependent SCOs are limited to a single use; they cannot be reused unless modified.

- Dependent SCOs such as introduction SCOs or transitional SCOs, which do not contain learning content, may reference learning objectives, course content, etc., for the learner to understand what is to be taught. This is acceptable to the Army. The Army considers these dependent SCOs as "single use" objects and not reusable.
- Dependent SCO files will be local files contained in the content package. Dependent SCOs can contain external references accessed by URLs. Refer to the [External References](#) section (3.12).
- Learner Performance Test SCO

Learner Performance Tests are excluded from rules of independent SCOs although they are not actually dependent SCOs either. The title of the test will reference the instruction covered in the test. Though the performance tests may be reusable, they do not follow all the rules of an independent SCO.

Hence, certain rules may apply to one type of SCO and not another.



**2.1.3.4-1 Business Rule (Army SCORM): Independent SCOs shall be able to stand alone as self-contained blocks of learning content. The learning content of an independent SCO shall not contain references (for example, hyperlinks, narration, etc.) to another SCO or aggregation. An independent SCO shall not contain references to a course map or transitional statements such as: The next lesson will explain..., etc.**

- OR -



**2.1.3.4-1 Business Rule (Army Content): Independent and portable Chunks shall be able to stand alone as self-contained blocks of learning content. The learning content of an independent and portable chunk shall not contain references (for example, hyperlinks, narration, etc.) to other chunks or instructional material within the same \*course\*. An independent and portable chunk shall not contain references to a course map or transitional statements such as: The next lesson will explain..., etc.**



**Best Practice: An independent SCO or reusable content should be able to be extracted from its current package into a new package and play independently within the LMS. It should not depend on other learning content.**

## 2.2 Course Map, Lesson Flow Diagram, and Content Organization

### 2.2.1 Course Map

A course map is a graphic portrayal of the order in which a course/courseware should be presented. It shows the recommended and/or alternative training sequence for presenting the material to be

learned based on the learning hierarchy. The course map shows the sequences and relationships among lessons in the instructional unit. The map (Figure 2.5.1a) shows how activities, or structured units of instruction, relate to each other.

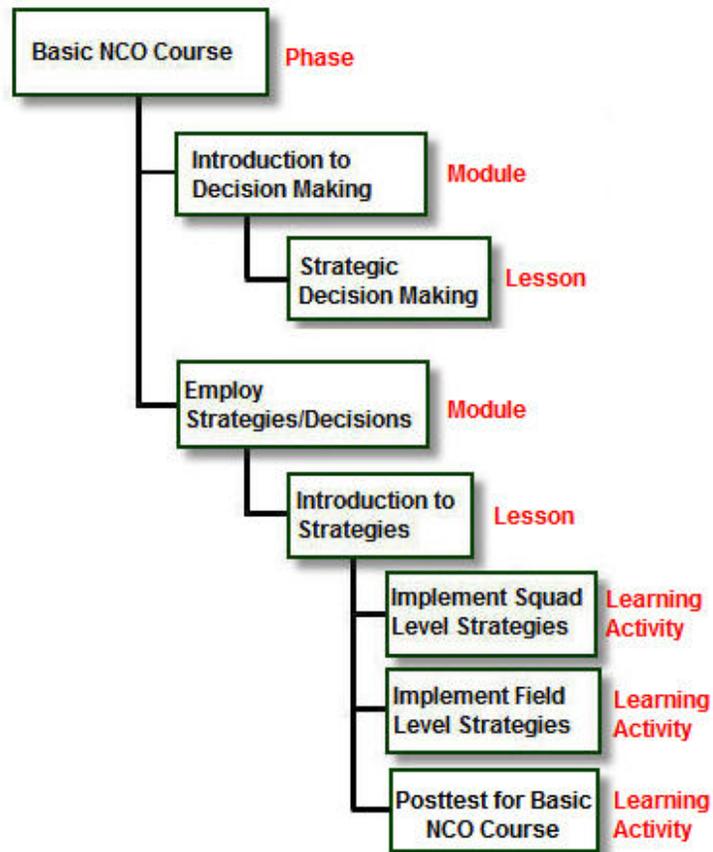


Figure 1.5.1a

Instruction	Army Classification	Structure
Basic Non-Commissioned Officer Course	Course Title	Phase
Introduction to Decision Making	Cluster	Module
Strategic Decision Making	Independent SCO	Learning Activity
Employ Strategies/Decisions	Cluster	Module
Introduction to Strategies	Dependent SCO	Lesson
Implement Squad Level Strategies	Independent SCO	Learning Activity
Implement Field Level Strategies	Independent SCO	Learning Activity
Posttest for Basic Non-Commissioned Officer's Course	Test SCO	Learning Activity

Figure 2.5.1b

### **2.2.2 Content Organization**

In SCORM objects are collected together into packages for transporting. Each package must have a manifest file provides the LMS with the content organization and lists the files that provide the instructional material. The Content Organization section provides the structure to the learning objects in SCORM and the Resources section identifies each file in the package. Each SCORM content package has an Organization section and a resource section. The Content Organization section of begins as a single activity and is expanded as needed. (Figure 2.5.2a and figure 2.5.2b)

Activities in a Content Organization may consist of sub-activities, which may themselves consist of other activities. There is no set limit to the number of levels of nesting for activities. While learning taxonomies may be associated with hierarchical levels of activities, (for example, course, phase, module, lesson, etc.), this is not a requirement. Activities at the bottom of the list will have an associated learning resource (either a SCO or asset resource) that consists of the files that execute the activity.

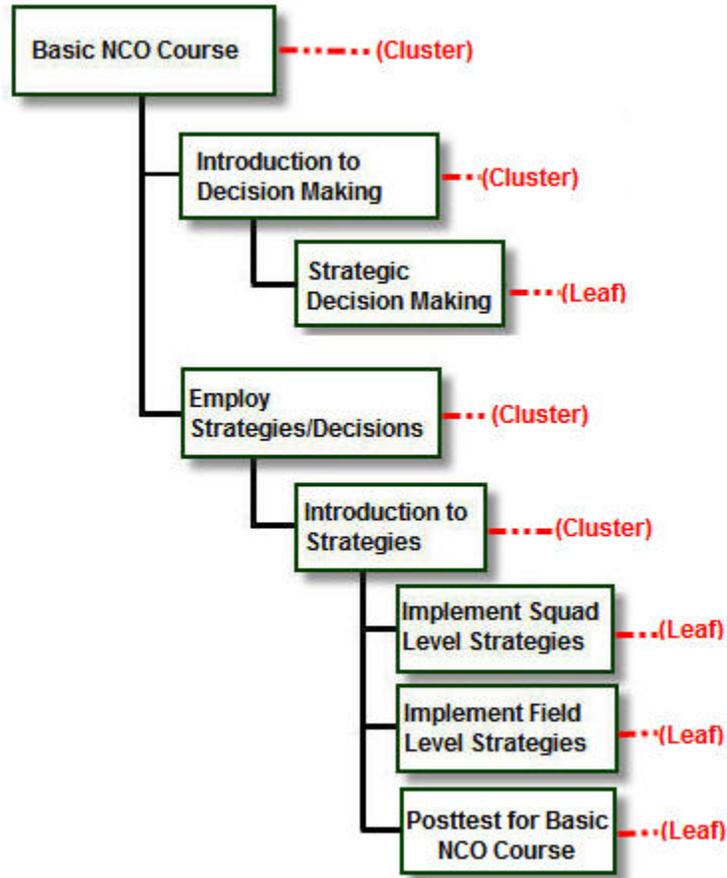


Figure 2.2.2a

Instruction	Army Classification	Structure	Activity
Basic Non-Commissioned Officer Course	Course Title	Phase	Cluster
Introduction to Decision Making	Cluster	Module	Cluster
Strategic Decision Making	Independent SCO	Learning Activity	Leaf
Employ Strategies/Decisions	Cluster	Module	Cluster
Introduction to Strategies	Dependent SCO	Lesson	Cluster
Implement Squad Level Strategies	Independent SCO	Learning Activity	Leaf
Implement Field Level Strategies	Independent SCO	Learning Activity	Leaf
Posttest for Basic Non-Commissioned Officer's Course	Test SCO	Learning Activity	Leaf

Figure 2.2.2b

Note: This manifest file, named 'imsmanifest.xml', is an essential part of all SCORM 2004 Content Packages. It describes the contents of the package (TOC for the LMS) and may include an optional description of the content structure (Metadata). A SCORM content package is a collection of related learning objects and can represent any learning level, i. e., a course, a module, a phase, a lesson, a TLO, an ELO or a learning step. The SCORM content package is the most significant object in SCORM since the SCORM content package IS the most portable SCORM object and is inherently more easily reused than a SCO. The SCORM content package in the Army dL environment is the only SCORM object that exists outside of the LMS and it is the only SCORM object that can be transported, without modification, from one LMS to another, or from one learning platform to another when playable outside of an LMS.

Army courses in the institutional domain are too large to be a single SCORM content package. Large collections should be broken down into smaller stand-alone objects, or chunks. Army chunking strategy is currently targeting 20 to 50 (twenty to fifty) minutes of instruction per chunk. These chunks will be delivered as independent “content object” packages. These content objects are portable SCORM packages that are loaded into the LMS as building blocks. The loaded SCORM content packages are combined using LMS construction methods the hierarchical level learning event that supports learner registration and tracking is built. Small 20 to fifty minute chunks will simplify the SCORM packages and much of the sequencing and credit awarding logic will be the responsibility of the Army LMSes; Saba, Blackboard, and in the future AtlasPro. This will reduce content complexity and therefore reduce development time. Reduced content complexity should reduce network and re-start errors that learners experience.

Refer to the TRADOC Reg 350-70, [Chapter VI-6, Training Course Design](#) or [Chapter VI-6-7 Structuring and Sequencing](#) for more information about constructing a course map.

### **2.2.3 Lesson Flow Diagram**

A lesson flow diagram defines lessons tasks with references, information frames or sequence, decision points, branching options, remediation, and other screen activity. Following is an example of a generic lesson flow diagram that is usable for implementing an Army course. The SCORM programmer will interpret the Lesson flow Diagram and produce the proper SCORM packages that will be used to recreate the diagram in the host LMS. Actual lesson flow diagrams should include the names of all the SCOs that are in a package supplied by the SCORM programming team.

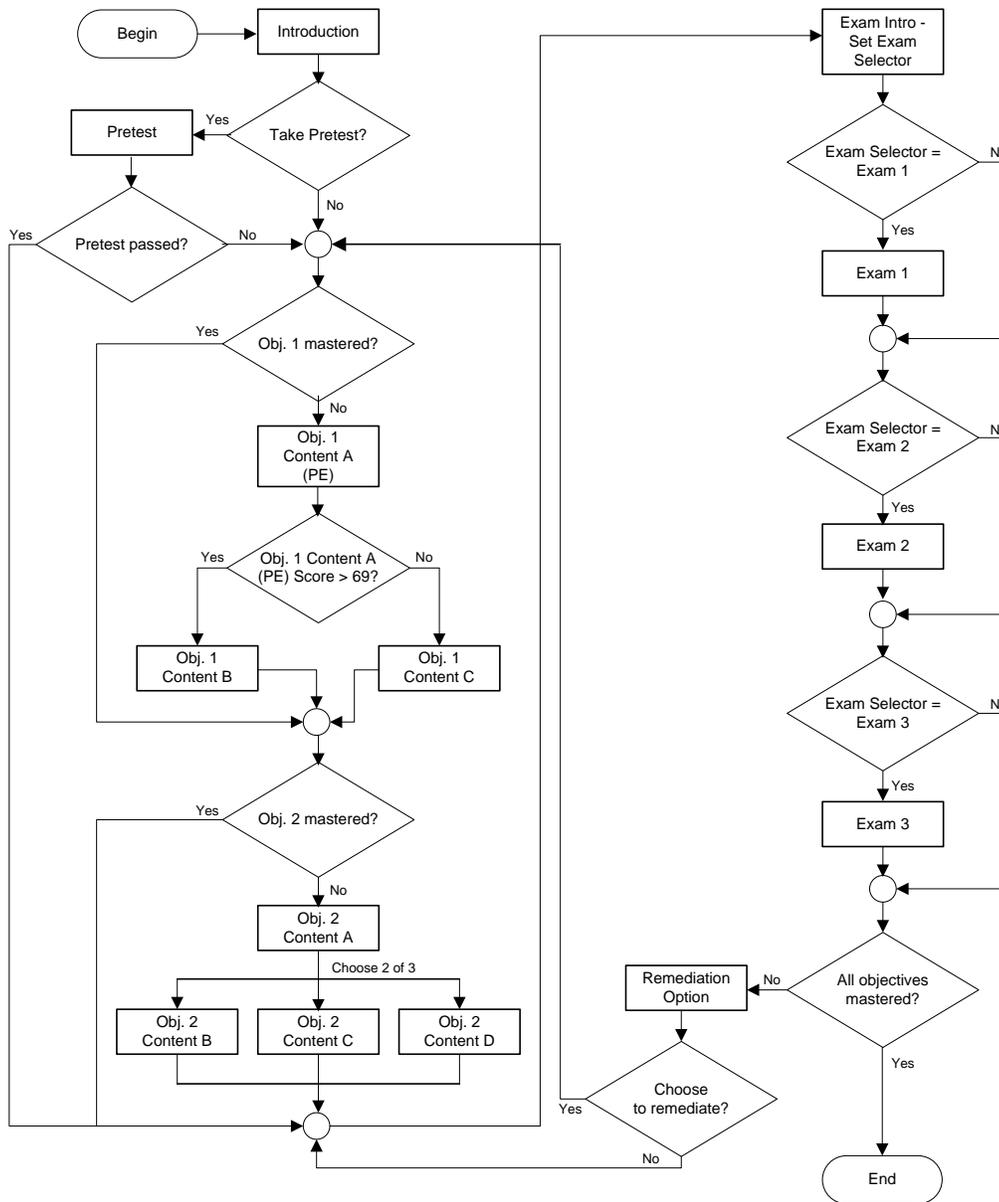


Figure 2.2.3a

### 3. Content Design Specifics

Design reusable learning content as small chunks of stand-alone, context-free learning objects for all learning objects that are determined to be candidates for reuse. Then the reusable objects and all the other learning objects can be gathered together (aggregated) into bigger chunks and the course constructed. The instructional designer and the programmer should work closely to meet the instructional goals. The development of specialized instructional components which enrich the learner experience and keep interest level elevated should be incorporated even if the tracking of the engaging content does not conform to the constant connection or tracking requirements of the SCORM. Instructional areas that must be tracked and required or content that evaluates learner performance will almost always adhere to the SCORM specifications, but if the instructional designer and the programmer work closely areas where engaging material can be placed if the engaging material demands limited LMS involvement or disconnected execution. The instructional designer will need to clearly define the required sequencing of the content and testing strategy to the programmer so that correct sequencing and application of credit will occur.

Thorough analysis and design of courseware brings about cost savings in development. Reusability brings about the benefit of not spending development dollars for content that already exists. The instructional designers should review Defense Automated Visual Information System/ Defense Instructional Technology Information System (DAVIS/DITIS) and Advanced Distributed Learning Registry (ADL-R) for courseware that may be used. The time spent in analysis and design phases will reap rewards during the implementation phase.

Programmer Info: Please note that the manifest file, which is part of the Content Package, is referred to many times in this section for understanding concepts. Refer to the Programming Examples, [SCO Example](#) and [Cluster Example](#) sections (9.1.2, 9.1.3) for examples of sequencing code for SCOs and clusters.

#### 3.1.1 SCO Acronym

The acronym SCO is a SCORM term that learners are neither familiar with nor expected to understand. Displaying this acronym will confuse the learner, possibly causing more delay in completing the learning material or eliciting more requests for help from support personnel.



**Best Practice:** Use the word 'Item', 'Activity', or the word 'Objective' (depending on the context) to replace the SCO acronym in learning content. For example: Select an activity from the table of contents.

### 3.1.2 SCO Titles



**3.1.4-1 Business Rule (Army SCORM):** The SCO title contained within the SCO displayed to the learner must be the same as the SCO title that is displayed on the LMS Table of Contents.

This section is also appropriate for non-SCORM instruction.

The following screen capture demonstrates how a SCO title contained within the SCO displayed to the learner is the same as the title in the LMS's Table of Contents. The SCO being viewed is titled 'Strategic Decision Making'. Likewise, the SCO title in the LMS's Table of Contents is also 'Strategic Decision Making'. Keeping the title the same in the SCO as in the table of contents makes it easier for the learner to orient him/herself within the course. It also reduces confusion and increases the ease of use.

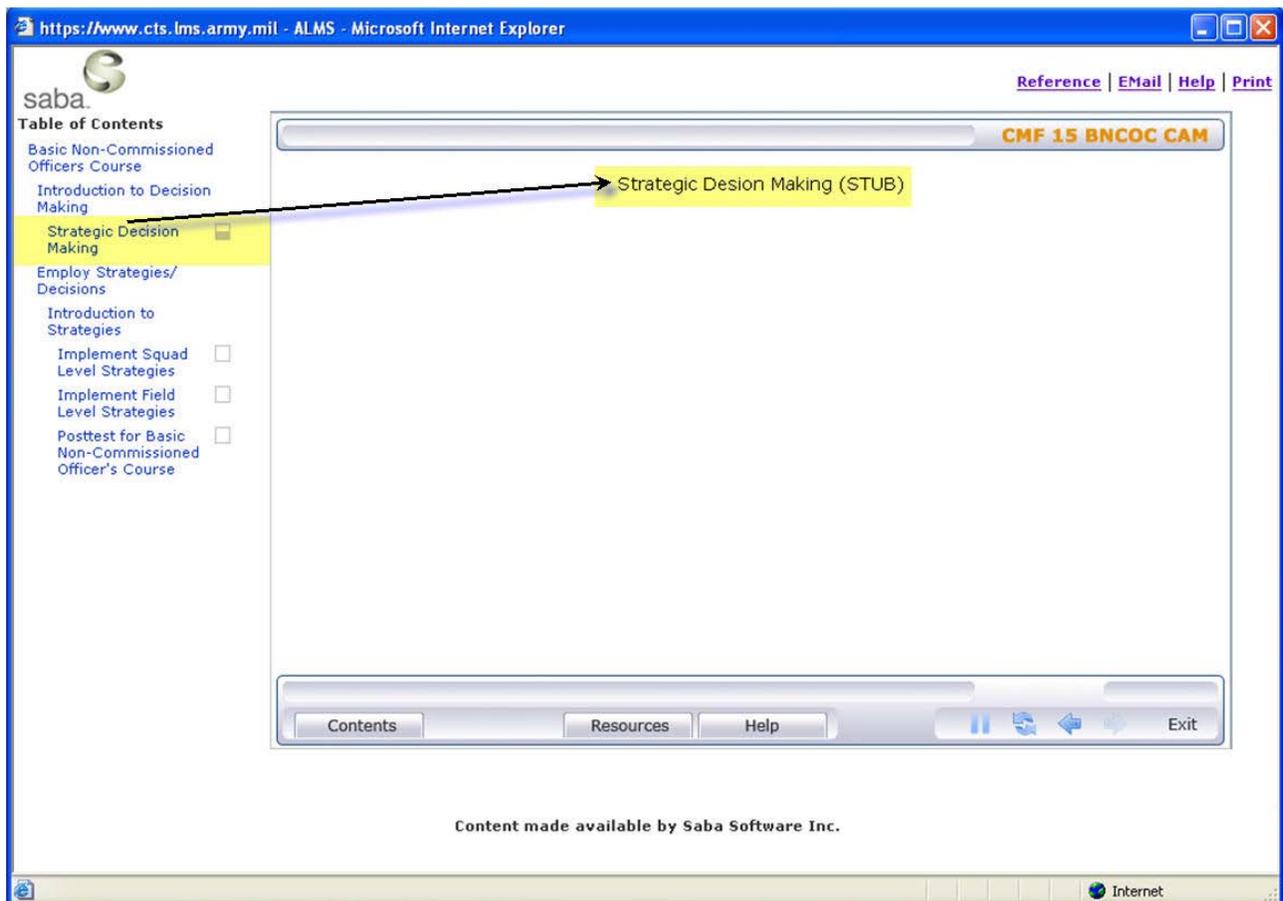


Figure 3.1.2a

A graphical table of contents is not needed, as titles for each SCO are XML tagged in the Organization section of the manifest file. The LMS reads the manifest file and generates the Table of Contents display. The Sequencing, Navigation, and Rollup rules also come from the Organization section. The “Sequencing Control” navigation tags can be set to allow user choice, where the table of contents is displayed within the LMS and the learner can choose any learning activity. If user choice is not desirable, the programmer can provide instructions to the LMS to automatically launch learning activities in the proper order.



**3.1.4-2 Business Rule (Army SCORM): Hierarchical higher level instruction titles or words (course, phase, module, lesson, etc.) must not be contained within independent SCO content or titles. An independent SCO containing an ELO must not reference its TLO within the content or title of the SCO. Independent SCO titles must not include the MOS, Skill Level, SQI, ASI, or SCO acronym within the actual SCO, navigation links, or the manifest; independent SCO content and titles must not include sequencing numbers.**

The following section has been edited to include non-SCORM content in its presentation.

One title exactly describing the learning content within a SCO or learning object is best and promotes reusability. A properly titled SCO or content object will always be able to be reused in a different course. When a SCO or content object designed as an ELO references its TLO within its content or in its title, that SCO or content object is not longer reusable, since it is no longer independent – it is now identified (or dependent) on its’ TLO. A properly titled SCO or learning object can be reused at a different level. SCOs and learning objects properly named and designed as ELOs may be reused as TLOs or as a learning step in another course. Reusability and independence is accomplished by SCO and content object titles that reflect the action statement of the learning objective.

SCO and content object titles should be displayed on every page of the content to keep the learner aware of where he is within the learning content.

A properly titled SCO or learning object does not have numerical sequencing. Reusing SCOs and learning objects that display numerical sequencing such as Chapter 1 or Topic 1 in the title will mislead and confuse the learner.



**Best Practice: Independent SCO and content object titles are significant to the learning resource and should display the Action statement of the learning objective contained within the independent SCO or content object. The SCO and content object title is usually displayed on every page.**

Independent titles with the action statements are more appropriate and more explanatory of the instruction that is being taught. The following are examples of well-formed independent SCO or content object titles:

Adjust Drive Belt Tension on M923 Prepare for Deployment Select Activities of Terrorist Operations Describe the Master Gunner's Role and Responsibilities Identify Safety Procedures for an Armor Unit
--

Figure 3.1.2b

4

### 3.1.3 Linking to Additional Learning Resources

Instructional designers should work with the programmer(s) to package additional learning resources. For example, the learner may need to link to a glossary, references, or help. These additional learning resources should be made to come up in a separate window from the courseware to prevent learner confusion or inability to navigate back to the course.

All files containing learning content must be physically located within the content package. Learning content is defined as content that is germane to the instruction and on which the learner will be tested.



#### **3.1.5-1 Business Rule (Army SCORM): Learning content files for all SCOs must be physically located within its SCORM content package (within the root folder or subfolder) and not referenced by a URL.**

Being physically located in the package would indicate that learning content files would not use a URL outside of the local domain to access the instructional files. While technically possible, the Army has decided not to link to external URLs. However, URLs are permitted when hyperlinking to references provided to the learner for further study. These hyperlinked references can be located outside of the package. The information in these 'external' references would not be evaluated as part of an assessment in order to complete the instruction. Refer to the [External References](#) section (3.12) for how to include URLs in courseware.

### 3.2 Learning Objectives and Global Objectives

The SCORM 2004 Sequencing and Navigation (S&N) functionality has introduced a new tool, 'Objectives', which programmers may use to affect navigation within a content package. This has led to some confusion throughout the community when discussing SCORM 2004 conformant instruction and objectives. These objectives may refer to actual learning objectives within the courseware or they may be simple SCORM variables used by the developer to affect different S&N strategies. In one respect objectives have pedagogical significance, while in another they are merely a tool to guide a learner through the instruction. Objectives will be discussed throughout this guide. The authors have attempted to be explicit in all discussions whether the objective is an instructional learning objective or a SCORM variable.

### 3.3 Course Title



**Best Practice: The course title should not be contained within or referenced from the title or content of independent SCOs or any content that may be reusable.**

A course title is only the name of a particular collection of SCOs or content objects. It is for reusability reasons that titles be excluded from independent SCOs and other content objects. The course title should not be contained anywhere in independent content since any reuse of that content would require the course title to be modified and the content could not be reused as-is.

Course title is the required name entry for the Organization section of a manifest file, as such it is normally displayed in the TOC area and is excluded from the restrictions - it is easily changed compared to items displayed from content resource files. An MOS, course definition or description, school name, etc. are allowed in the course title.

Refer to the Programming Examples, [Course Title Example](#) section (9.1.1) for a coding example of the course title.

### 3.4 File Naming Convention

It is imperative that courseware be portable to other SCORM compliant LMSs. All courseware file names, folder names, and URLs should operate in Microsoft® Windows and UNIX® environments.

Since UNIX® file names are case sensitive and Microsoft® Windows file names are not, consistent naming of files should consider capitalization. 'File.txt' and 'file.txt' are not the same file in UNIX®, but are the same file in Microsoft® Windows.

Programmer Info: Unique file naming and identification is essential for sharing courseware across operating systems, Web servers, and workstation browsers. While it is technically possible to assign unique file names to all files, unique names are not always practical or desirable. It is common practice to use 'index.html' as the launch file name for launching Sharable Content Objects (SCOs) or any other content objects in organized presentations. By placing launch files in different folders, the 'index.html' file is uniquely named because the folder name is included in identifying Web and workstation resources. The file identification 'http:// www.any.mil/ folder1/index.html' is not the same file identification as 'http:// www.any.mil/folder2/index.html'.



**Best Practice: The Army recommends the following naming convention for all URLs, file names, and folder names within the all resource content packages:**

- **A-Z, a-z, 0-9, hyphen (-), and underscore (\_).**
- **A single period shall be used to separate the file name from its extension. Periods shall not be used in a folder name.**
- **In each directory, all file names with their extensions shall be unique.**

Examples:

Unitsafety\_sco\_a328.html  
p1180-launch.html  
ca217\_metadata.xml  
sco-21.xml  
M60\_graphic.gif

Figure 3.4a



### **3.4-1 Business Rule (Army Content): Mandatory URL, file, and folder format and size:**

- **A three character file extension is recommended. A single period shall be used to separate the file name from its extension. Folder names shall not contain a period.**
- **The path separator shall be either </> or <\> depending upon the operating system (Microsoft® Windows, UNIX®) because the browser will resolve the correct path using the appropriate separator.**
- **Long file names are allowed but must conform to ISO 9660 with Microsoft® Joliet extension and the Army file naming scheme. Joliet allows filenames up to 64 characters in length. While Joliet allows spaces, the Army naming scheme does not.**
- **The Army's file naming scheme is extended to Government Furnished Information (GFI) for all files internal to all resource content packages.**

GFI Example for file naming within a resource content package:

Contractor receives, as GFI, a file named 'AR 5-23 Unit Safety.pdf'. This file mixes use of upper and lower case characters and uses the restricted 'space' character. Before using the file in the courseware, it is renamed as 'AR\_5-23\_Unit\_Safety.pdf', replacing the space with an underscore.



**Best Practice: Use of reserved and unsafe characters in dL courseware: There are a number of characters that should not be used in a Uniform Resource Locator (URL) because they cause a conflict in the interpretation of URLs or they are reserved by URL schemes. The reserved and unsafe characters listed below are not recommended for use in dL file/folder/URL names for files/folders/URLs contained within a SCORM 2004 or any content package containing web-based content. However, these characters may be used for courseware URLs external to the content package as long as they are 'escaped' using their equivalent hexadecimal notation (for example %3A, which is equal to: [colon]).**

Reserve and Unsafe Characters with their 'Escape' Code:

Space	%20
"	%22
#	%23
%	%25
&	%26
/	%2F
:	%3A
;	%3B
<	%3C
=	%3D
>	%3E
?	%3F
@	%40
[	%5B
\	%5C
]	%5D
^	%5E
'	%60
{	%7B
	%7C
}	%7D
~	%7E

Figure 3.4b



**3.4-2 Business Rule (Army Content): Additional characters that shall not be used in dL courseware for naming files/folders: The following characters cause problems in a UNIX® Operating System environment:**

| ; , ! @ # \$ ( ) < > / \ " ' ` ~ { } [ ] = + & ^ <space> <tab>

An industry practice and an Army recommendation is to limit the alphabetic characters to all lower case letters, the underscore (\_) and the hyphen (-) characters. Spaces should be excluded. This reduces many of the human errors associated with the consistent naming of URLs, files, and folders names that have upper and lower case letters.

Examples:

```
unitsafety-sco-a328.html
p1180_launch.html
ca217-metadata.xml
sco_21.xml
m60-graphic.gif
```

Figure 3.4c



**3.4-3 Business Rule (Army Content): All Uniform Resource Locators (URLs) within the courseware with unsafe or reserved characters shall be replaced with their hexadecimal equivalent.**

Files external to the resource content package:

Contractor receives the URL 'http://hostname/AR 5-23 Unit Safety.pdf' as GFI to insert into the courseware as a hyperlink. The hyperlink to the file is coded in the courseware as 'http://hostname/AR%205-23%20Unit%20Safety.pdf' where 'space' = %20 (hexadecimal).

## 3.5 Navigation

### 3.5.1 Navigation Help Feature



**3.5.1-1 Business Rule (Army SCORM): A Navigation Help feature must be available at all times. It must explain the function of all internal navigation buttons available to the learner because the behavior of each button and button label is not determined by SCORM 2004. If the LMS User Interface buttons are enabled at points throughout the learning experience, the navigation Help button shall refer the learner to the LMS for further explanation of the LMS navigation features.**

The Navigation Help feature can be implemented by including navigation instructions within the introductory SCO or topic, a separate dependent SCO or topic, or a launchable asset.

### 3.5.2 Intra-SCO or simple Navigation

Navigation within the SCO or topic itself is intra-SCO or simple navigation and is designed by the courseware developer.



**Best Practice: The same navigation button behavior and name label should be implemented across all content in a resource content package with a consistent look and feel.** Courseware buttons that have the same button label but different functionality creates confusion for the learner when clicking on a button and being taken to a different location than assumed. This is frustrating to the learner.

A standardized design for navigation supports the easy reuse of learning content. The instructional designer can harvest content from a repository and use them in other courses without creating a frustrating learning event for the learner. The TTADOC GUI, mandatory for interactivity level one and level two content is provided for this reason.

#### Navigating Out of a SCO

SCORM navigation has changed with the introduction of the LMS User Interface (UI) This is going to be difficult for the instructional design team unless they are familiar with the target LMS.

Consider the SCO and LMS navigation buttons when the learner:

- Reaches the last page of the SCO
- Wants to prematurely suspend the lesson
- Wants to launch another lesson
- Wants to return to where the lesson was suspended

Refer to the [LMS User Interface \(UI\) Devices for Navigation](#) section (4.3.3). Also refer to the [Bookmarking](#) section (5.3.1).

When navigating away from the SCO, a learner may click on the X at the top right of the browser window instead of using the courseware navigation buttons. In this case, content must provide code to trap the unusual exit to properly handle exit processing. Refer to the [Communication with the LMS](#) section (5) and the Programming Examples, [SCO Example](#) section (9.5.2) for examples of code regarding terminating communication appropriately. Army SCORM business rule 9.5.2-1 requires all content to provide the processing to gracefully exit a SCO even when the content provided exit controls are not used by the learner.

### 3.6 Table of Contents (TOC)

The SCORM manifest file contains the XML-tagged TOC. The LMS reads the manifest file, translates organization item tags into an activity tree for processing purposes and to the TOC for display and navigation purposed. During setup and during run-time, the LMS interprets sequencing and navigation rules. A well presented TOC can enhance the learning experience of the learner.

Consideration should also be given to the number of items presented in the LMS generated TOC. Even though the TOC will scroll when items extend further than one screen, scrolling to reach TOC items is not an acceptable presentation or processing option for the learner.



**Best Practice: The Table of Contents or any navigation tool provided for the learner should not be presented in a scrolling screen. It is recognized that some lists are best processed with scrolling and some presentations can have different number of items before scrolling, otherwise this Best practice would be a business rule.**

Menus inside of SCO and independent content objects are acceptable as needed for intra-SCO and intra-topic navigation and branching to topics.

Refer to the Programming Examples, [XML Examples in the Manifest](#) section (9.1) for an XML example of a TOC contained within the manifest.

### 3.7 Folder Structure Example

The directory structure should be logical and intuitive, and created early in the design process. It is the developer's responsibility to define the folder structure; it is not mandated by SCORM.

The following is a recommended file structure, consisting of suggested folders in the right frame:

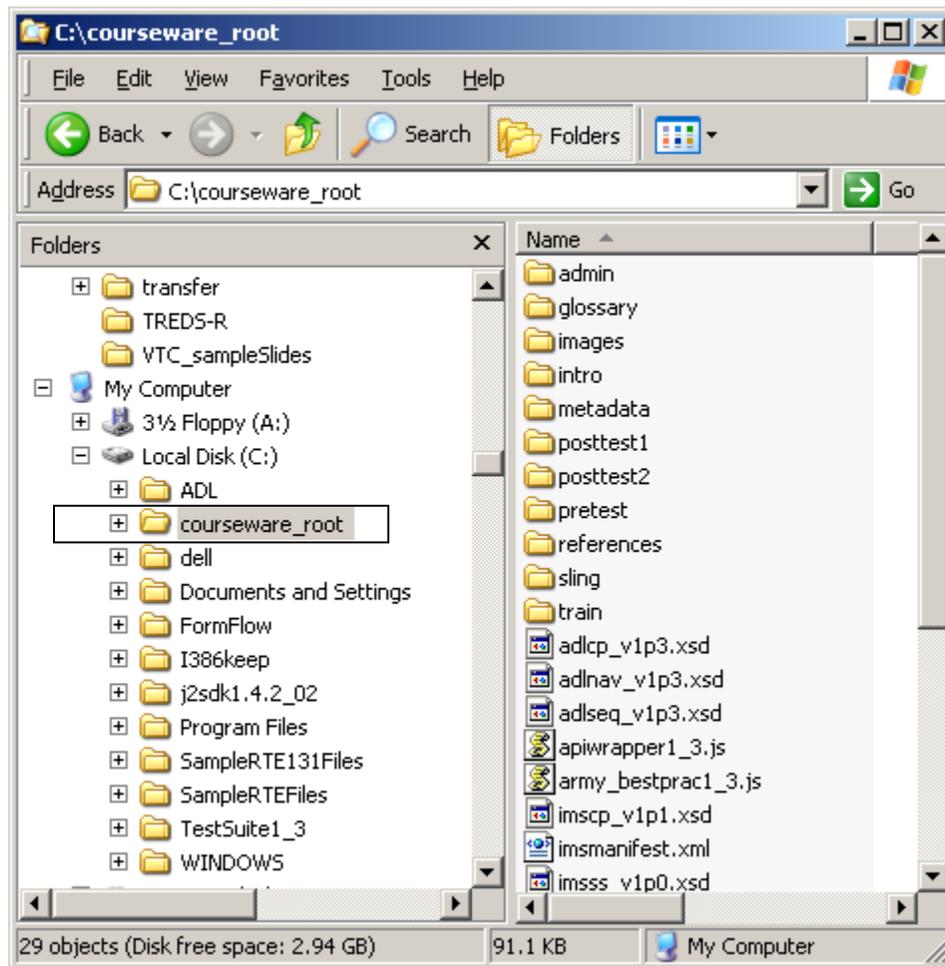


Figure 3.7a



**Best Practice: Recommend storing all the metadata files in one place by creating a metadata folder, as shown above.**

### 3.8 Checks-on-Learning

Checks-on-Learning are not normally included in test item analysis. Therefore, test item data for checks-on-learning are not collected and the Army does not require this test item data.

However, if determined by the proponent that test item data for checks-on-learning are desired to be collected, then refer to the [Test Item Data](#) section (5.3.11) for further information.

### 3.9 Course Map

A course map displayed to the learner within the courseware provides the learner with an overview of the course and the number of modules and/or lessons the learner will have to complete. It is usually developed graphically and included within the courseware to enable the learner to understand the course structure.

The course map is specifically correlated to the structure of the course. It usually cannot be reused in a different course. Placement of the course map must not be within independent, reusable SCOs. Options are placing the course map within a dependent SCO or as a launchable asset. Refer to Dependent SCOs in the [Army Classification of SCOs](#) (2.1.3.4) and the [Launchable Assets](#) sections (3.10).

There are several different design strategies for placement of the course map page(s) in SCORM compliant courseware, listed as follows:

- A course map may be included in the course as a dependent SCO. Consideration should be given to the fact that the learner is required to launch the SCO and receive a "completed" completion status. Not opening this SCO would interfere with the learner completing the course because the completion status would remain as "not attempted". However, if it is mandatory that the learner access the course map and the completion status is needed to be tracked, then a separate dependent SCO would be reasonable. If developed as a SCO, the course map cannot link to another SCO directly but can use navigation requests for the LMS to launch specific lessons. A training developer would not want to communicate navigation requests to the LMS to jump to an individual SCO, because the sequencing process would be suspended.
- The course map may be included in an existing dependent SCO, such as an Introduction to the Course SCO containing a hyperlink to the course map page. The actual course map could be placed within the dependent SCO. Completion status would be tracked in the context of the entire SCO and not specifically the course map.
-  **ADL Reference:** As a launchable asset, the course map could be excluded from having any effect on the learner's completion of the course. The learner may access the course map only if desired, and the course map would be available from the TOC. The course map can be an HTML page with a graphic and cannot use navigation requests. Unlike a SCO, a launchable asset does not communicate with the LMS.

Refer to the [Course Map, Lesson Flow Diagram, and Content Organization](#) section (2.5) regarding a course map diagram that is required for the sequencing phase.

Note: Sequencing depicted by a course map is normally only used for initial training. For sustainment training, the learner will normally be given free choice of what SCOs they will take.

### 3.10 Launchable Assets



**Best Practice: Develop an activity (introduction, overview, summary, course map) as a launchable asset when learner completion progress does not need to be tracked by the LMS.**

An activity can be developed as a launchable asset when learner tracking is unnecessary. Instead of developing an introduction or summary as a SCO, which requires communication and tracking within the LMS, a launchable asset may be recommended. A launchable asset is exactly the same as normal Web pages containing text and/or graphics and is launched from the TOC. There is one file that is designated as an entry point (launch file).

Refer to the Programming Examples, [Launchable Asset Example](#) section (9.1.4) for a coding example of a launchable asset.

### 3.11 Designing for Reusability



**ADL Reference: The SCORM approach [to sequencing] fosters reusability of learning resources by allowing content developers to define sequencing and navigation behavior and instructional strategies independent of the actual learning resources.**

Reusability is one of the foundations for the SCORM specification. Instructional designers and/or developers should keep this aspect in mind. Each independent SCO should be able to be extracted from an existing course or selected from a SCO repository to be reused in another course structure. There should be nothing within independent SCOs that is dependent on other SCOs or dependent on course context.

Eliminating context seems to cause confusion for the educational designers who are taught that each lesson must build on previous knowledge. Therefore, the Army has developed the concept of independent and dependent SCOs as explained in the Army Classification of SCOs section (3.1.2). Dependent SCOs provide the transition for the learner to connect the context-free independent SCOs and dependent SCOs are not reusable.

A sequenced instructional strategy that is hardwired within the courseware produces a course that is tied together. SCORM has adopted a set of sequencing rules that is not contained inside of the SCOs, but is defined external to the SCO, specifically, within the manifest file. For a properly prepared course, when a SCO is reused within another course, the sequencing rules can be redefined.

### 3.12 External References



**Best Practice: Recommend external references not link directly to the URL.**

SCORM allows access to learning content by URLs, although the Army has chosen not to recommend this method within Army courseware. Refer to the [Linking to Additional Learning Resources](#) section (3.1.5). The Army allows external references that refer to non-learning content

that support professional development and/or a more in depth understanding of the subject matter that is beyond what the learner needs to meet course requirements. URLs and their content, by the very nature of the Web, are subject to change and are not controllable by the course. External references could be used for Army Regulations (ARs), Field Manuals (FMs), TRADOC Regulations (TRs), Pamphlets (Pams), etc. These external references should come up in a separate window to prevent learner confusion or course navigation problems.



**Best Practice: External references should link to a local redirect page. This method would allow ease of maintenance and conversion. For courses to be easily converted to CD-ROM stand alone courseware, all external references within the courseware should link to a local redirect page. For the online Web version, this redirect page will have the URL outside of the local domain. For the courseware to be used from a CD-ROM, the external reference itself would be copied or downloaded and stored locally on the CD-ROM. Programming code on this page will contain a statement directing the learner to either the URL (for online) or to the local resources that are contained within the package.**

The visual representation is as follows:

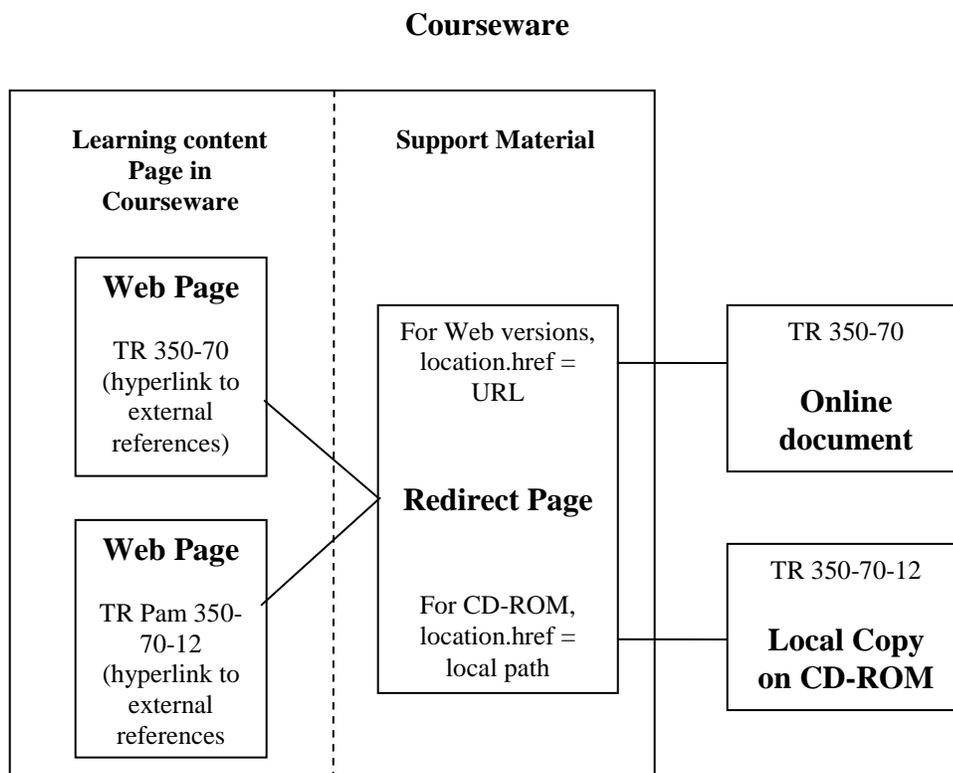


Figure 3.12a

Editing the redirect page is less time intensive than editing the SCO within the authoring tool, which requires the original files. Manual editing can easily occur on each redirect page by changing the URL to a local path. The filename for the redirect page should easily identify that it is a URL redirect page for a specific Web site, perhaps in a 'redirect' folder.

Refer to the Programming Examples, [External References Example](#) section (9.18) regarding an external reference coding example.

## 4. Sequencing and Navigation

This section provides a minimal presentation of SCORM's implementation of sequencing, navigation and rollup. Sequencing and navigation are two integral parts of the SCORM specification. One affects the other because in order to sequence activities, the learner must be able to initiate navigation. Though one might see these as three separate components, they are not.



**ADL Reference: SCORM defines a method for authoring the behavior of a learning experience that is repeatable within another LMS environment. In other words, a SCORM compliant LMS should sequence discrete learning activities in a consistent way. It defines the flow of learning activities in terms of an activity tree, based on a learner's interactions with launched content objects (files from the resource section of the imsmanifest.xml file) and the authored sequencing strategy (XML tags in the Organization section of the imsmanifest.xml file).**

Note: ADL acknowledges that the SCORM certification process for LMSs is inadequate to completely support the complete presentation of learning objects and all sequencing and navigation operability of different LMSs. There is no guarantee a course will behave identically in every different ADL-certified LMS. Therefore, the sequencing, navigation, and rollup of your course should be tested in your target LMS.



**ADL Reference: A SCORM conformant LMS translates the Content Organization within the manifest to an activity tree, which represents hierarchically defined learning activities, including the tracking status information for each activity on a learner by learner basis.**

### 4.1 Cluster/Leaf

Clusters define the piece by piece break down that allows rules and logic to create the instructional flow for an unbridled collection of individual learning objects. Clusters define how learning activities are gathered together into units of instruction. They are logical pairs (parent & child) used to painstakingly construction the hierarchical organization of logic statements and rules that provides an ordered delivery of activities to the learner.



**ADL Reference: A cluster is a specialized form of a learning activity that has sub-activities; the term is used in various sequencing behaviors. A cluster contains a single parent activity and its immediate children, but not the descendents of its children. The children of a cluster are either leaf activities or parent components of another cluster. A leaf activity is not a cluster it is always a child that contains content for delivery according to the sequencing strategy for the cluster that contains it.**

Clusters are important to sequencing because sequencing and rollup rules apply to either a parent or a child. Sometimes, clusters must be created to accommodate a particular instructional strategy.

Following is an example of a single cluster:

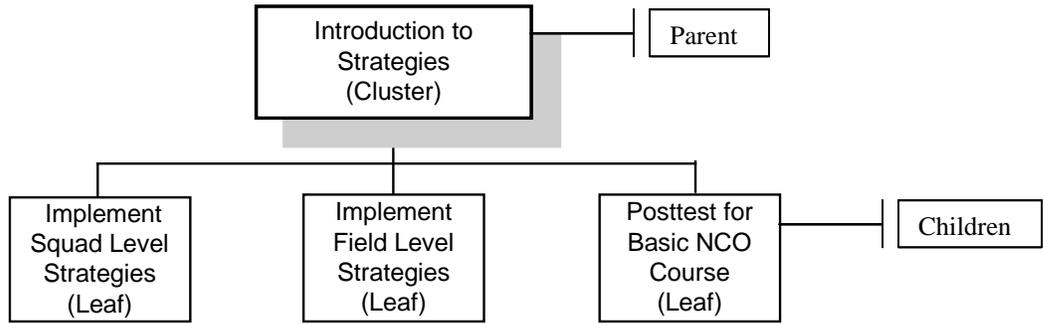


Figure 4.1a

Following is a diagram of four clusters (clusters are circled):

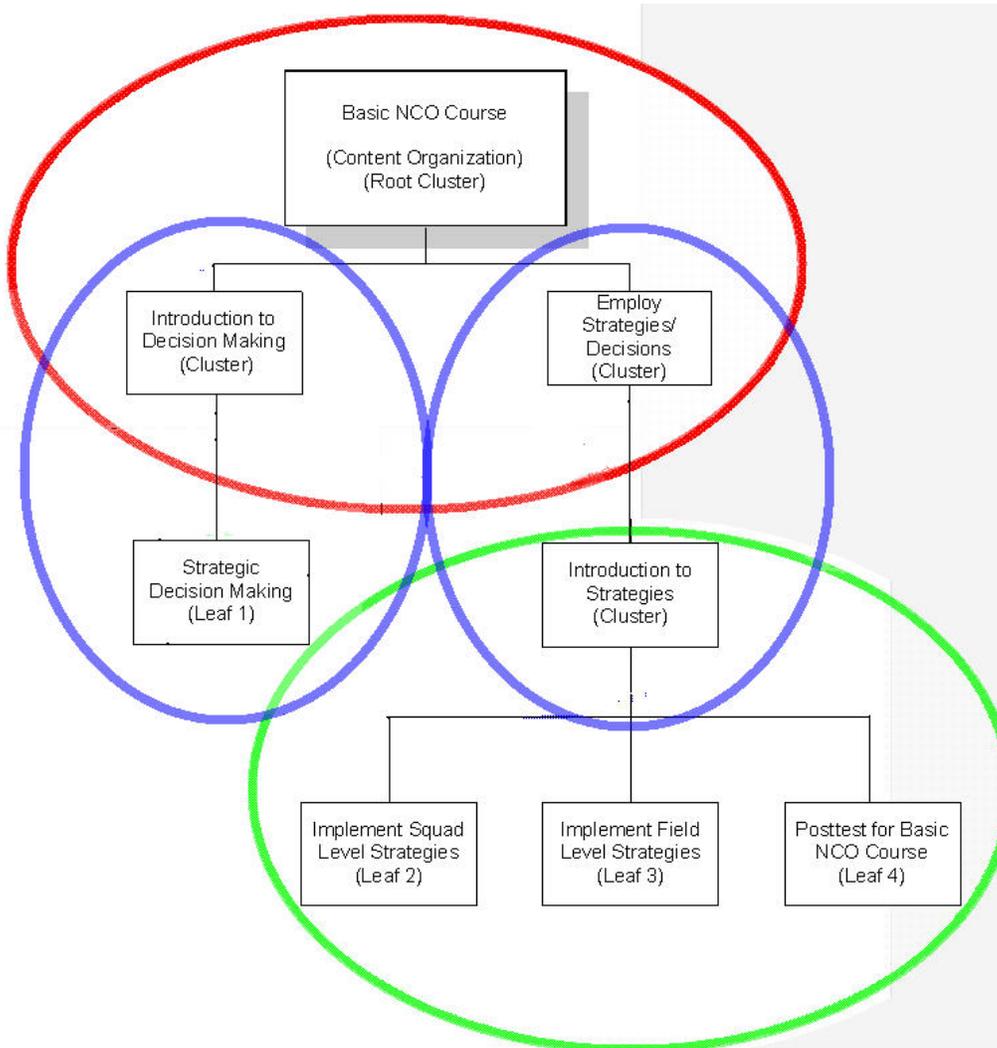


Figure 4.1b

## 4.2 Sequencing Strategy

The SCORM sequencing component allows for many and various sequencing strategies. Rule-based sequencing provides a method to specify a certain condition, and based on that condition, cause an action to occur. Rollup Rules indicate how sub-activities are evaluated to produce one result to carry forward. Shared data allows an activity to function differently based on the results of another activity. All these capabilities produce sequencing strategies for many instructional and testing strategies.

The worksheets provided in this section are helpful in understanding and applying rule-based sequencing and rollup rules by presenting the choices in columnar format. The choices for the condition are in one column and the choices for the action are in another column.

### 4.2.1 Rule-Based Sequencing

Navigation behaviors may be achieved by assigning rules to different nodes at different levels of nesting. When the learner is guided to the next activity, rules are evaluated to determine whether that activity should be presented or not. Rule-Based Sequencing can be compared to 'conditional rules' because the rules follow a pattern of "if this condition is true, then take this action."

List of possible conditions:

<b>CONDITIONS</b>	
If Attempted	If Not Attempted
If Objective Status is Known	If Objective Status is Not Known
If Satisfied	If Not Satisfied
If Activity Progress is Known	If Activity Progress is Not Known
If Completed	If Not Completed
If Objective Measure is Known	If Objective Measure is Not Known
If Objective Measure is Greater Than	If Objective Measure is Not Greater Than
If Objective Measure is Less Than	If Objective Measure is Not Less Than
If Always	If Not Always
If Attempt Limit is Exceeded	If Attempt Limit is Not Exceeded

Figure 4.2.1a

List of possible actions:

<b>ACTION</b>
Skip
Disabled
Hide from Choice
Stop Forward Traversal
Exit the SCO
Exit the Cluster
Exit the Course
Retry the SCO
Continue the SCO
Go to previous SCO

Figure 4.2.1b

SCORM allows the conditional sequencing rules to consist of one or more conditions and a corresponding action. When one or more conditions occur, then the action is implemented by the LMS.

Following is an example of a Worksheet that can be used by the instructional design/training development staff to communicate to the programming staff:

SCO # OR TITLE	CONDITION COMBINATION	CONDITION	ACTION
	<input type="checkbox"/> Any <input type="checkbox"/> All	(Choose one or more) <input type="checkbox"/> If Attempted ( <input type="checkbox"/> Not) <input type="checkbox"/> If Objective Status is Known <input type="checkbox"/> If Satisfied ( <input type="checkbox"/> Not) <input type="checkbox"/> If Activity Progress Known ( <input type="checkbox"/> Not) <input type="checkbox"/> If Completed ( <input type="checkbox"/> Not) <input type="checkbox"/> If Objective Measure is Known ( <input type="checkbox"/> Not) <input type="checkbox"/> If Objective Measure is Greater Than ( <input type="checkbox"/> Not) <input type="checkbox"/> If Objective Measure is Less Than ( <input type="checkbox"/> Not) <input type="checkbox"/> If Always ( <input type="checkbox"/> Not) <input type="checkbox"/> If Attempt Limit is Exceeded ( <input type="checkbox"/> Not)	(Choose one) <input type="checkbox"/> Then Skip <input type="checkbox"/> Then Disable <input type="checkbox"/> Then Hide from Choice <input type="checkbox"/> Then Stop Forward Traversal <input type="checkbox"/> Then Exit the SCO <input type="checkbox"/> Then Exit the Cluster <input type="checkbox"/> Then Exit the Course <input type="checkbox"/> Then Retry the SCO <input type="checkbox"/> Then Retry the Course <input type="checkbox"/> Then Continue to next SCO <input type="checkbox"/> Then go to Previous SCO

Figure 4.2.1c

Note: The worksheet may not be needed for every rule-based sequence necessary; however, it can be helpful to the instructional designer and the programmer when working through a hard spot.

## 4.2.2 Global Objectives



**4.1.2-1 Business Rule (Army SCORM): The scope of all manifest global objectives shall be the manifest itself. Manifest global objectives will be used for sharing the sets of Objective Progress Information.**

The SCORM specifications have the default manifest objectives mapped to the LMS system level. This makes a learner's result from one course possibly influence the settings for that learner in another course in the LMS. This is not desirable in the Army environment and the alternative setting is the focus of the Army business SCORM business rule 4.1.2-1. At this setting the success

of a learner in a SCORM package is only significant for that package. Until the Army has defined objectives globally all Army training and education this Business Rule will be appropriate. This business rule means that Army manifest objectives are for Objective Progress Information related to that single manifest.

The following is important for the SCOs can indirectly access data from other SCOs with the implementation of global objectives. For example, a pretest SCO can write a value such as "passed" to a global objective. Then, other SCOs, as they are launched, can read this value and apply sequencing rules based on this value. If the objective indicates "passed", then the LMS could automatically launch the next lesson pretest or target a specific SCO for launching.

To be used, objectives need sequencing rules such as a condition with a resulting action. The condition of the SCOs reading the "passed" value could have the sequencing condition of 'if satisfied' and the action would be "skip".

Following is an example of sharing data from a pretest:

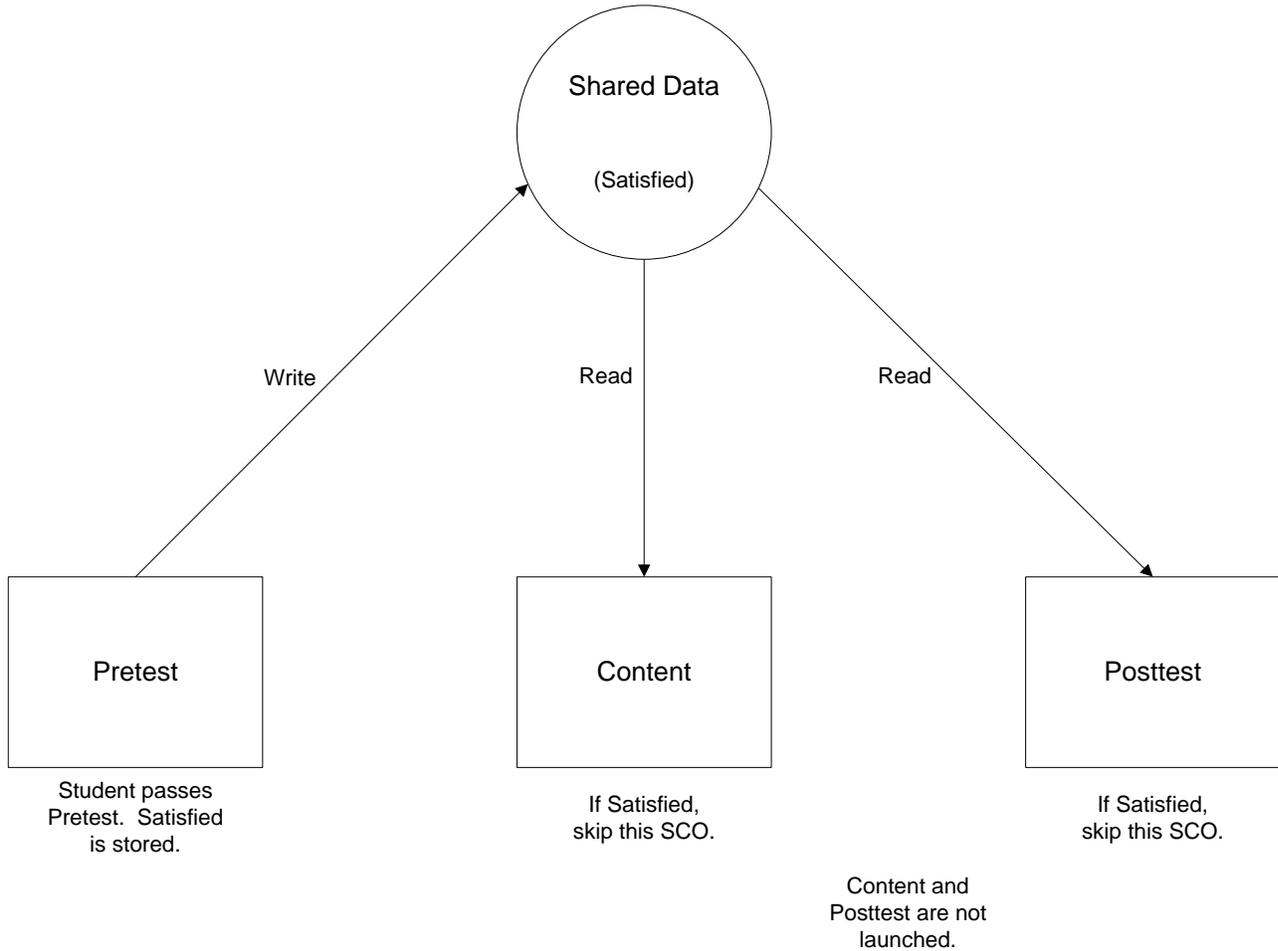


Figure 4.2.2a

### 4.2.3 Rollup Rules

Rollup rules function within parent-child relationships (Clusters, remember?). Child activities indicate a hierarchical relationship with a parent activity. Rollup rules provide a way to determine a single outcome on a set of child activities.

Data is collected by tracking the results of the sub-activities and the results are rollup up into summary results. This may be desirable for the course's instructional strategy. Perhaps, the learner is only required to complete two out of three child activities. Possibly, one of the child activities may not need to be included in the rollup and can be excluded using a rollup rule.

List of possible conditions:

<b>CONDITIONS</b>	<b>SUB-ACTIVITY EVALUATION</b>	<b>ACTION</b>
Choose one or more	Choose one	Choose one
<input type="checkbox"/> If Attempted	<input type="checkbox"/> All	<input type="checkbox"/> Evaluate as Satisfied
<input type="checkbox"/> If Not Attempted	<input type="checkbox"/> Any	<input type="checkbox"/> Evaluate as Not Satisfied
<input type="checkbox"/> If Satisfied	<input type="checkbox"/> None	<input type="checkbox"/> Evaluate as Completed
<input type="checkbox"/> If Not Satisfied	<input type="checkbox"/> At Least a Certain Count	<input type="checkbox"/> Evaluate as Incomplete
<input type="checkbox"/> If Never	<input type="checkbox"/> At Least a Certain Percent	
<input type="checkbox"/> If Objective Measure is Known		
<input type="checkbox"/> If Objective Measure is Not Known		
<input type="checkbox"/> If Activity Progress is Known		
<input type="checkbox"/> If Activity Progress is Not Known		
<input type="checkbox"/> If Completed		
<input type="checkbox"/> If Not Completed		
<input type="checkbox"/> If Attempt Limit is Exceeded		
<input type="checkbox"/> If Attempt Limit is Not Exceeded		

Figure 4.2.3a

#### 4.2.3.1 Rollup Considerations

By default, all sub-activities are included in their parent's rollup. However, SCOs can be further designated as to exclusion in the rollup process. For example, a sub-activity may not contribute to rollup if it was skipped by the learner through sequencing rules. This condition can be designated by 'rollup considerations' in the manifest.

Following is a list of the conditions:

<b>ACTION</b>	<b>DESCRIPTION</b>	<b>CONDITION</b>
Measure Satisfaction If Active	Indicates that the activity's rolled-up measure should be evaluated against the activity's 'mastery score' even if the activity is still active	If activity is still active
Required For Satisfied	Indicates when the activity's tracking information contributes to the rolled-up Satisfied status of its parent	Always If it is not suspended If it is attempted If it is not skipped
Required For Not Satisfied	Indicates when the activity's tracking information contributes to the rolled-up Not Satisfied status of its parent	Always If it is not suspended If it is attempted If it is not skipped
Required For Completed	Indicates when the activity's tracking information contributes to the rolled-up Completed status of its parent	Always If it is not suspended If it is attempted If it is not skipped
Required for Incomplete	Indicates when the activity's tracking information contributes to the rolled-up Incomplete status of its parent	Always If it is not suspended If it is attempted If it is not skipped

Figure 4.2.3.1a

#### **4.2.4 Attempt Limits**

Limiting the number of attempts on a graded assessment can be part of the testing strategy. SCORM implements an attempt limit, but advises to use with caution. Limitation of attempts should be based on proponents' training strategy and test security. The tighter test security influences the number of test attempts. If test security is not an issue, the use of unlimited attempts is recommended. Network problems and LMS problems could interfere with the learner's attempts that could create a lock out or an unwarranted failure. Time Limits

Time limits are usually needed in timed assessments or simulations. Time limits can be implemented in SCORM by designating a time limit. The limit applies to only the accumulation of time that the learner is actually interacting with the activity.

#### **4.2.5 Randomization**

Randomization is the re-ordering of sub-activities. This capability will allow one cluster to function as a multi-version posttest or a test item pool.

Randomization Controls can be customized by the following:

- When the randomization should occur (timing): never, once, on each new attempt
- Number of child activities that must be selected from the set
- Should child activities be randomized
- When selection should occur: never, once, on each new attempt

#### **4.2.6 Constrain Learner Choices**

When the learner is free to choose any activity from the table of contents, there may be instructional strategies that prohibit accessing some child activities before the main lessons are accessed. Two constraints are provided by SCORM, which are: (1) prevent activation, and (2) constrain choices to those logically next in the activity tree.

#### **4.3 Navigation Strategy**

Navigation becomes extremely important when sequencing is applied to the courseware. Normally, navigation is thought of as navigating the pages of a lesson with a 'Next' and 'Back' button contained on each page. However, in addition to navigating through the pages, there is also a need to navigate the lessons. When a learner has completed a lesson, the method by which a learner navigates to the next target lesson according to the sequencing rules is vital. Therefore, navigating through the pages and navigating between lessons makes up the navigation strategy of the instructional unit.

Typically, the SCORM conformant LMS will provide a set of user interface devices that the learner may use to indicate a desired navigation request. However, in some cases, training developers may not want the LMS to provide the user interface navigation buttons because the content will provide them. Refer to the [Navigation Requests](#) section (4.4).

There are different components to SCORM Navigation. Each component listed should be addressed in the navigation strategy:

- Will the learner's primary navigation be 'user choice' where the learner chooses the activity?
- If not, then the learner will be presented activities based on the pre-ordered path through the activity tree. This presentation of activities is based on a user action or performance results, which is initiated by either additional buttons inside the SCO or initiated by the LMS user interface (UI) buttons.
- Determine whether the need exists to use navigation requests where the learner is directed to an activity other than the pre-ordered activity tree. There is functionality to 'jump' outside of the pre-ordered path of the activity tree to present an unordered activity to the learner. The "jump" functionality does not override the prevailing learner access status for an activity, that is, if an activity is not "active" or is currently not available for the learner - the SCORM "jump" navigation command will be unsuccessful.

### **4.3.1 Control Mode**

Control Mode determines how the learner interacts with the learning activities, which determines how sequencing requests are applied to a cluster. Control modes can be described as follows:

- Choice indicates the learner's freedom to choose any activity by a visible table of contents.
- Flow indicates that the learner is guided forward through the activities according to the structure of the tree, being allowed to traverse to the previous activity only.
- Forward Only indicates that the learner is guided through the activities but unable to traverse to the previous activity.
- Choice Exit indicates that the learner is denied any navigational choice until a specific event occurs. This mode occurs as a sub-mode within Choice, Flow, or Forward Only.

#### **4.3.1.1 Choice Control Mode**

Choice mode allows the learner freedom to choose any activity in a cluster in any order without restriction. The table of contents is displayed but the LMS User Interface buttons of 'Previous' and 'Continue' are not displayed. Choice mode can be enhanced by combining two control modes, which are further defined in this section.

#### **4.3.1.2 Flow Control Mode**

Flow control mode displays the LMS User Interface of the 'Previous' and 'Continue' buttons. Flow mode automatically navigates to the next sequenced activity after clicking on the LMS 'Continue' button. Flow only allows the learner to access only the previous activity or the next activity. The learner is not allowed to choose but must follow the defined flow.

Following is an example with the 'Continue' and 'Previous' LMS buttons displayed:

**Note: If a SCO is composed of more than one HTML page, and requires internal navigation buttons, it is best to NOT use the LMS-provided navigation buttons. Examples following are for illustration purpose ONLY to explain how control modes and LMS buttons and navigation requests function.**

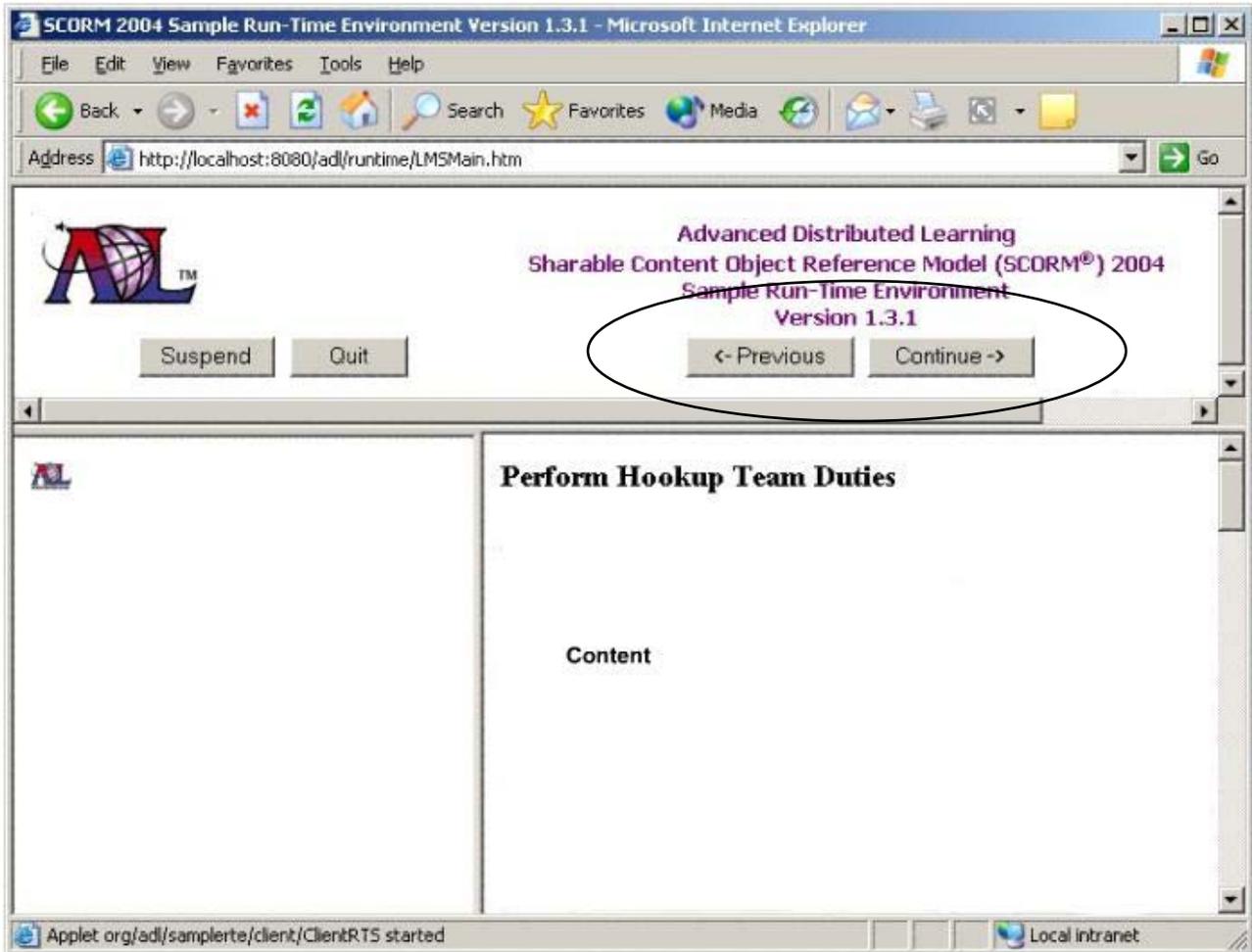


Figure 4.3.1.2a

Since the LMS User Interface provides the required 'Continue' button at the top of the LMS window, consideration should be given to the intra-SCO navigation and displaying similar buttons that navigate through the pages of the SCO. Refer to the [LMS User Interface \(UI\) Buttons for Navigation](#) section (4.3.3). The SCORM specification recommends that the LMS does not provide a redundant mechanism for the learner to indicate 'Continue' and 'Previous' navigation requests – doing so may result in two sets of navigation controls, which may confuse the learner.

### 4.3.1.3 Forward Only Control Mode

Forward Only control mode automatically navigates to the next sequenced activity after clicking on the LMS 'Continue' button. Navigating to the previous activity is not permitted. The learner will automatically access content in the preordered sequence.

Following is an example of Forward Only:

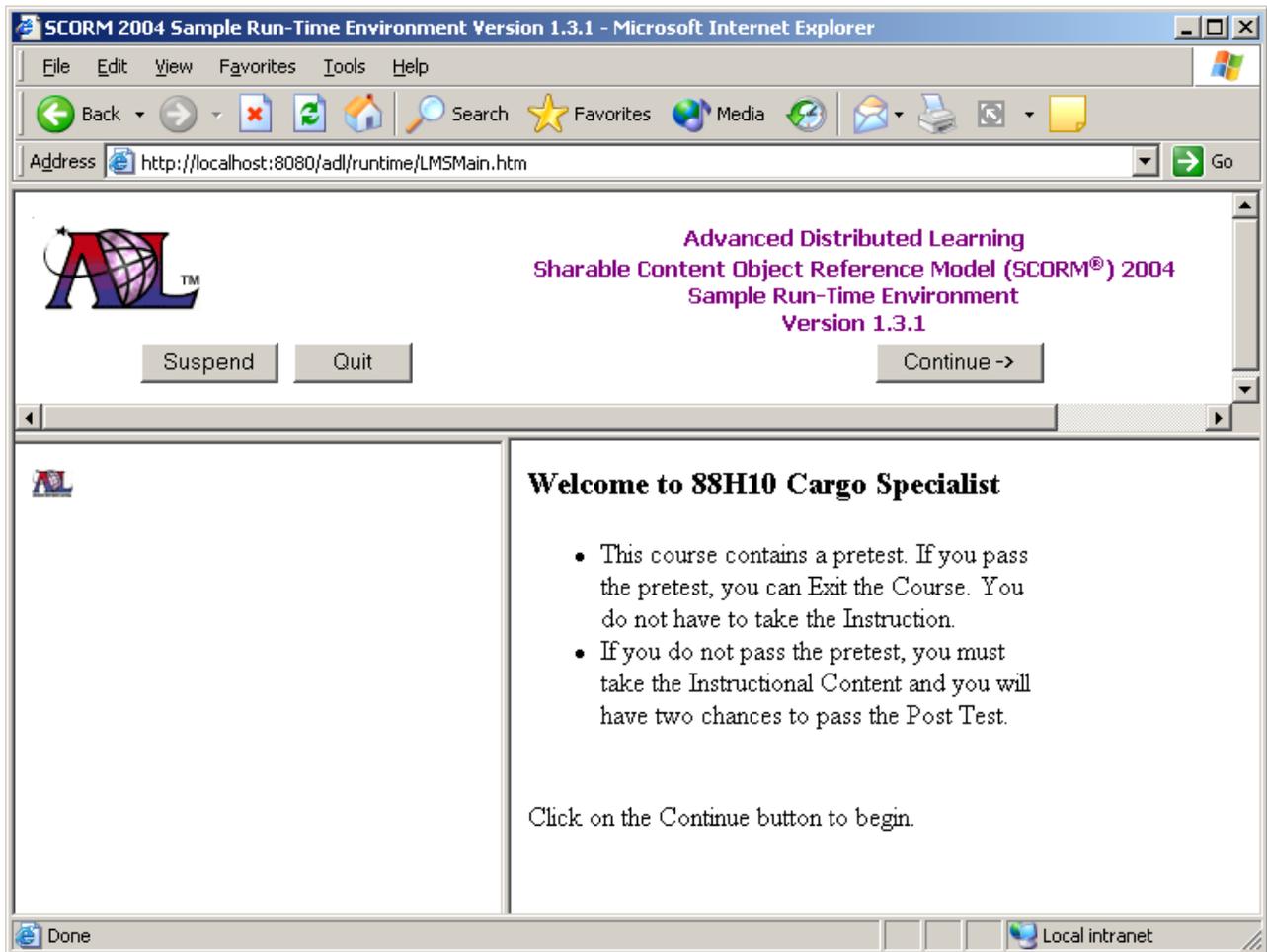


Figure 4.3.1.3a

### 4.3.2 Customization

Learner Navigation can be customized or automated for a specific instructional strategy by adding run-time navigation. Run-time navigation allows a SCO to direct the LMS to launch a specific target activity not in accordance with the activity tree. For example, this action may allow the learner to return to a previous topic without actually going to the TOC; however, it limits reuse. Refer to the [Navigation Requests](#) section (4.4).

### 4.3.3 LMS User Interface (UI) Buttons for Navigation



**ADL Reference:** The LMS provides the ability for a learner to trigger navigation events via UI devices. The allowable LMS UI buttons must, at a bare minimum, execute the navigation events 'Continue', 'Previous', 'Suspend', and 'Quit'. It is important to realize that two of these navigation events ('Continue' and 'Previous') enable the learner to navigate to the next SCO or the previous SCO. The other two events either suspend the course or totally exit the learner out of the course.



**ADL Reference:** Navigation events triggered by the LMS UI always take precedence over the navigation requests communicated by a SCO. If the learner initiates a navigation event using the LMS UI and the SCO has sent a different navigation request, the LMS ignores the navigation request and enforces the UI navigation event. Refer to the [Navigation Requests \(4.4\)](#) section for more information.

This is one reason that the Army does not recommend that the LMS UI be used. Learners may activate the wrong 'Previous'/'Continue' button and go to the next SCO when all they wanted to do was go to the next page.



For example, following is the display of a Completion page and the 'Previous' and 'Continue' buttons are hidden. The learner is forced to exit or suspend the instruction using the LMS buttons.

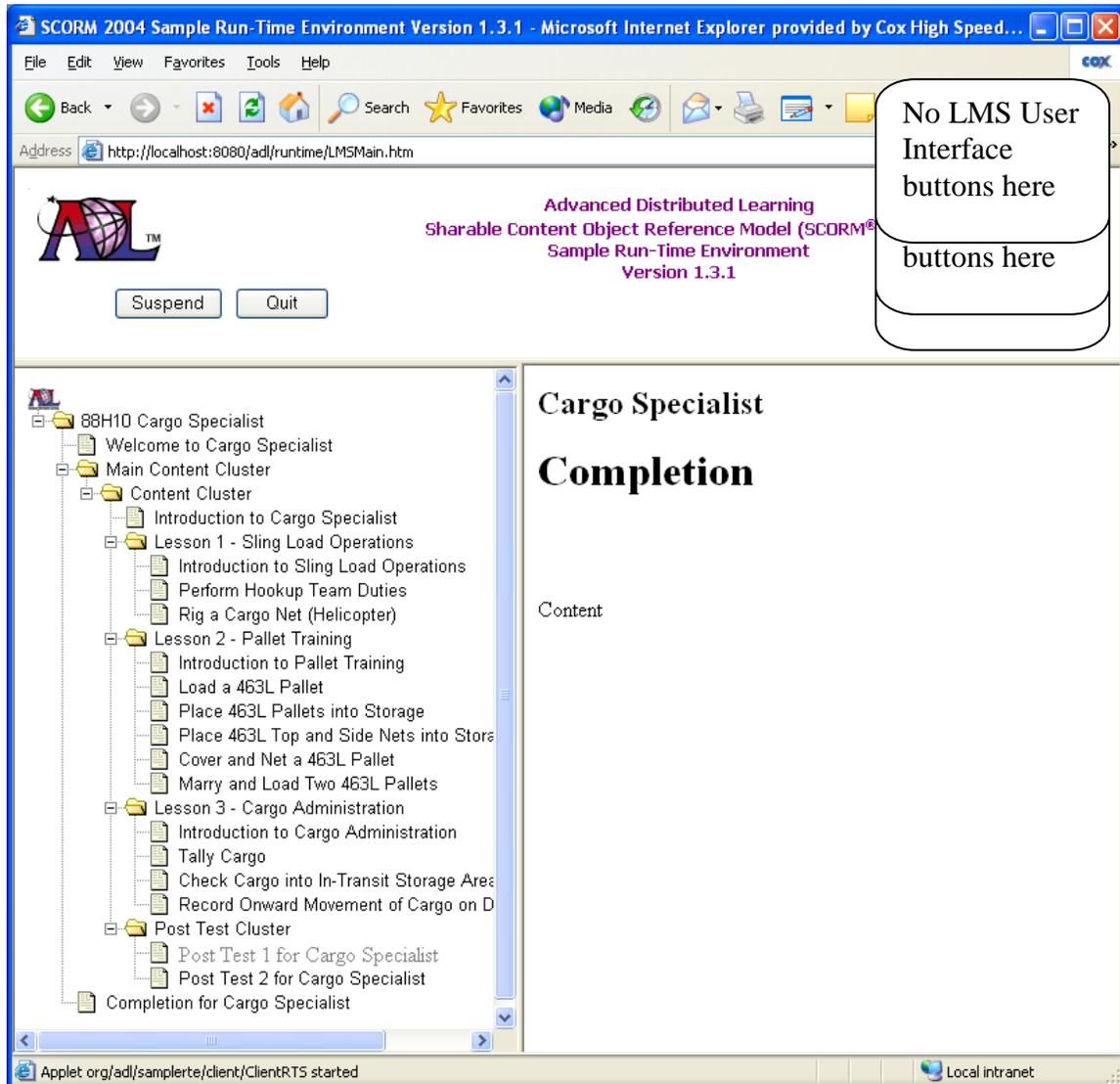


Figure 4.3.3.1a



**Best Practice:** The LMS UI buttons of 'Continue' and 'Previous' should be hidden during the learner's experience of a multi-page SCO. Additionally, the SCO has to provide the same 'Previous' and 'Continue' functionality within the SCO.

Refer to the Programming Examples, [Hiding the LMS User Interface \(UI\) Buttons Example](#) section (9.2.8) for XML examples.

## 4.4 Navigation Requests

Note: Only dependent SCOs can use navigation requests as presented in this section.

A SCO can communicate navigation intentions to the LMS. A SCO can indicate to the LMS to launch a specific activity. This activity could be the next logical activity according to the pre-ordered activity tree or could 'jump' to an activity that is outside of the activity tree order. Navigation requests are not the same as a SCO calling another SCO. The SCO allows the LMS to launch another SCO; the SCO does not launch the SCO.

If the LMS UI is hidden and the learner is not free to choose an activity from a visible table of contents, then the learner **must** rely on the SCO initiating one of the navigation requests to the LMS to launch the next target activity. **Otherwise, there would be no means of navigating to the next activity.**

Remediation can be implemented by using the navigation request. Upon determining a learner's weak area of the learning objective, a navigation request would jump to the activity containing the instruction to review.

If a course map is contained within the course as a dependent SCO, it could also use navigation requests to tell the LMS to launch that specific activity.

Refer to the Programming Examples, [Navigation Requests Examples](#) section (9.20) for coding samples.

## 5. Communication with the LMS (Run-Time Environment)

Instructional designers need to be cognizant of the SCO communication with the LMS and the data that can be stored and retrieved. It is very helpful to those who design courseware to understand what data can be stored and retrieved in the LMS.

A goal of the SCORM is that learning resources are reusable and interoperable across multiple Learning Management Systems. The SCORM Run-Time Environment (RTE) supports this by providing a common way to start learning resources (launch), a common mechanism for learning resources to communicate with an LMS (API), and a predefined language or vocabulary forming the basis of the communication (Data Model). JavaScript is the programming language necessary for implementing communication to the LMS. The following examines SCORM Run-Time Execution State Commands, State Management, and Data Transfer Commands.

Note: At the time of release of this document, the ADL Technical Working Group (TWG) acknowledges that the SCORM certification process for LMSs is inadequate to support interoperability of content between different LMSs. There is no guarantee that a course will behave in the same way across different ADL-certified LMSs. Therefore, your course should be tested in your target LMS.

The API enables the communication of data between content and the RTE provided by an LMS. The use of a common API fulfills many of SCORM's high-level requirements for interoperability and reuse. This provides a standard way for SCOs to communicate with the LMS. The purpose of the data model is to ensure that a defined set of information about SCOs can be tracked by different LMSs. Refer to the [SCORM Data Model](#) section (5.3) for further discussion on the data model.

### 5.1 Begin Communication

Begin communication is a specific command by the SCO to begin communication with the LMS. It is the responsibility of the SCO to initiate communication with the LMS, which signals the LMS to prepare for storing and retrieving SCORM data for a SCO session.

### 5.2 End Communication

End communication is a specific command given to end communication with the LMS. It is the responsibility of the SCO to terminate communication with the LMS, which signals the LMS to store all buffered data and cease communication with the SCO.

#### 5.2.1 Retrieving Data

Data is retrieved by the specific command of GetValue. It allows the SCO to obtain information from the LMS. The SCORM Data Model defines the variables and vocabulary that is used for this information exchange. The Army requires the SCO to retrieve certain information from the LMS.

## 5.2.2 Storing Data

Data is stored by the specific command of SetValue. It allows the SCO to write information to the LMS. The SCORM Data Model defines the variables and vocabulary that is used for this information exchange. The Army requires the SCO to store or write certain information to the LMS.

## 5.3 SCORM Data Model



**ADL reference:** The SCORM Data Model ensures a standardized and common way that SCO information can be stored and retrieved by different LMSs. For example, the completion status is always defined in the exact same way for all SCORM conformant courses. The data set may include, but is not limited to, information about the learner, interactions that the learner had with the SCO, objective information, success status, and completion status. This data can be used to track the learner's progress and status, aid in sequencing decisions, and report on the overall learner interaction with the SCO.

### 5.3.1 Bookmarking



**5.3.1-1 Business Rule (Army): SCOs shall provide a learner with the ability to bookmark their progress in a SCO when the learner leaves the SCO prematurely. The SCO must allow the learner the option of resuming their progress in that SCO from the bookmarked location. This bookmarking capability is not required for any proponent allowed exceptions; for example, assessments, summaries, introductions, etc.**

Bookmarking is a process where the learner or the system suspends learning for a certain period of time, and upon re-entry, the learner may return to the point within the learning content where the learner left off. The learner is given a choice to return to the last location or, depending on the amount of time has elapsed, go to the first page and review previous material. Bookmarking does not bookmark which SCO the learner last accessed, but only bookmarks the last page accessed by the learner inside the SCO.

### 5.3.2 Learner Entry

The value is an indication of whether the learner has been in the SCO before. There are three values: "ab-initio", "resume", or normal (blank value). The LMS is responsible for establishing the initial value of "ab-initio" upon initial launch of the SCO. The value of learner Entry is determined by how the learner exited the SCO previously. If the learner suspends or logs out of the SCO, the LMS changes the learner entry value to "resume". If the learner times out, exits normally, or exits without setting a condition, then the learner entry value is blank.

This value is important to the instructional strategy of the SCO. If the value retrieved is "ab-initio", then the learner has not accessed the SCO previously. SCO developers can use this information to implement bookmarking, delivery of testing, etc.

### 5.3.3 SCO Status

SCO status is established using two values: completion status and success status. Completion status indicates when the learner is finished with the SCO and did not suspend prematurely. Success status indicates successful mastery of the objective. These two track separate concepts, completion and mastery. Completion does not indicate mastery.

#### Completion Status

Completion status is vitally important for a content SCO, which is a SCO that does not contain a graded assessment. There are four status values: "completed", "incomplete", "not attempted", and "unknown". The first time a course is presented to the learner, the completion status for all SCOs is "unknown". If the learner is not considered to have used the SCO in any significant way, the completion status is "not attempted". For example, the learner may accidentally open the SCO, realize he has opened the wrong thing, and immediately close it without navigating through any of the content. If the learner prematurely exits the SCO without experiencing enough of it to be considered completed, the completion status is "incomplete". If the learner proceeds to the end of the SCO, the completion status is "completed".

How the SCO determines its completion status is outside the scope of SCORM. The four completion status values listed above are determined by the SCO developer. The completion status can be set by a number of ways (for example, number of pages visited, result of pressing a button on the page, completion of viewing a video, completion of reading a document, completion of various objectives in the SCO, etc.). An example of setting different completion status values follows. Before the learner has entered the SCO, the completion status is set to "unknown". The developer may decide that if the learner only looks at the first page of a 30 page SCO, and then exits the SCO, the completion status should be set to "not attempted", because he has not experienced the SCO in any significant way. The learner may have simply opened the wrong SCO. The developer may decide that if the learner only visits five pages of a 30 page SCO and then exits the SCO, the completion status should be set to "incomplete". Finally, the developer may decide that the completion status is "completed" only if all 30 pages are visited by the learner.

Sequencing rules are applied based on a condition of "completed" or "incomplete". Completion status enables the LMS to track the SCO status. When a navigation request is submitted for the next activity, the completion status is considered by the LMS. If the completion status of a SCO is required to be "completed" to obtain credit for the course, then a completion status of "incomplete" will not provide the learner with mastery of the objective. Vice versa, if the instructional strategy only requires the learner just to attempt the SCO, then the SCO is "completed" based on just an attempt of the SCO. The SCO can determine when to set the status.

#### Success Status

Graded assessments are not judged to be completed or not. They are judged by the success status, which indicates mastery of the learning objective or not. Success status is the result of the learner's score compared to the 'mastery score' and determining "passed" or "failed".

Both the completion status and the success status determine a single rollup result (GO/NO GO), interpreted by the LMS and uploaded to the Army Training Requirements and Resources System (ATRRS).

### **5.3.4 Learner Score**

The learner score indicates the test score received by the learner. The format of the score is scaled, indicating that it is between -1 and 1.0, inclusive. The success of the SCO is determined by comparing the learner score to the 'mastery score'. The learner score is stored in the LMS. If the learner score is equal to or greater than the 'mastery score', "passed" is stored in the success status, and the learner receives credit. If the learner score is less than the 'mastery score', "failed" is stored in the success status, and the learner does not receive credit.

### **5.3.5 Mastery Score**

The 'mastery score' is the minimum passing score that must be attained by the learner to pass the graded assessment. SCORM allows the mastery score to be designated outside of the SCO within the manifest file. The LMS reads the manifest file and stores this value in the LMS. This will ensure maximum reusability.

### **5.3.6 Exit Status**

This value indicates how or why the SCO was exited by the learner. The values include "suspend", "time-out", or "normal". The "logout" value for the exit status has been deprecated by ADL and should be avoided.

"Suspend" indicates the learner leaves the SCO with the intent of returning to it later at the point where the learner exited.

"Time-out" indicates the SCO ended because the SCO has determined an excessive amount of time has elapsed, or the maximum time allowed in the SCO had been exceeded. The maximum time that is allowed in the SCO can be found within the manifest. Refer to the [Time Limit and Resulting Action](#) section (5.3.9) for more information regarding a duration limit.

"Normal" indicates that the learner has exited normally.

### **5.3.7 Session Time**

Session time is the amount of time the learner accessed the SCO. It is calculated from times stored in the LMS from the launching of the SCO to the closing of the SCO. This duration may be important in certain instructional strategies.

### **5.3.8 Total Time**

Total time is tracked by the LMS and indicates the accumulated time of all the learner's sessions in the SCO as reported by the SCO (using session time) when exited by the learner. Total time can be averaged and used to determine the average time needed for a learner to complete the instruction.

This value would be very important in a timed learner performance test or in a timed test that could be resumed.

### **5.3.9 Time Limit and Resulting Action**

The time limit indicates an amount of time the learner is allowed to have in the current attempt of the SCO. This time limit is required for 'timed' graded assessments. When the time has been exceeded, the training developer can determine what action should occur.

The possible time limit actions are:

- Display a message to the learner; no further action
- Display a message to the learner and force SCO exit
- Do not display a message to the learner and force SCO exit
- Do not display a message to the learner; no further action

### **5.3.10 Data Storage Area**

SCORM has implemented one data storage field to allow data needed by the SCO to be stored on a per SCO basis by the LMS. The LMS only provides a storage area for data. Usage of that data is not determined by the LMS. Data is simply returned to the SCO when requested. Examples of ways the storage field could be used are to:

- Store the number of attempts in a multi-version posttest SCO
- Store the start time the SCO was launched
- Store a bookmark within a large Flash file
- Store learner's bookmarks within the SCO content
- Determine if a posttest has been taken
- Determine if objectives are passed or failed
- Determine tracks learners have taken
- Determine errors learners have made (for after action review [AAR])

This field could contain unique information generated by the SCO during previous uses that is needed for the current use. Normally, this is an element used by the SCO for restart information, but can be used for other purposes.

This field can store learner placeholders or bookmarks within the SCO or pages so the learner can return. In a large SCO, functionality could be given to the learner to bookmark certain pages that the learner would like to return within the SCO.

### 5.3.11 Test Item Data

The **Interactions Data Model Element** is provided to allow training developers the ability to track information for validation purposes. Interactions record the status of a learner's interaction with an individual test item/question in a SCO. The data stored in an interaction element can be used to evaluate the effectiveness of checks on learning, or exam questions, either by the SCO or by some course management program provided by the LMS vendor.

**Test item data is required for all graded assessment SCOs. Refer to Business Rule 9.16-1.**

The developer needs to provide certain information to the programmer. The programmer needs to know:

- Type of each question (true/false, multiple choice, fill in, long fill in, matching, performance, sequencing, likert, numeric, other)
- Correct responses for each question
- If the assessment is timed
- If the assessment is weighted

See Appendix A for a table that can be used by the developer to fill out required data for the programmer's use. Also included is a populated example of a test item data table.

Refer to the Programming Examples, [Interactions](#) section (9.16) for examples and specific interactions.

## 6. Metadata

SCORM defines metadata as a way "to provide a common nomenclature enabling learning resources to be described in a common way." SCORM adopted the [IMS Global Learning Consortium's Learning Resource Metadata Information Model](#).

Metadata files should be separate XML files referenced within the manifest file. The file name and location is identified in the manifest. Army required metadata should not be placed directly into the manifest file.

Metadata is used for reusability and discoverability, and provides information about the learning content. SCORM highly recommends the practice of including metadata, but it is not required. However, the Army has identified optional SCORM metadata requirements as 'Army Mandatory' for Content Organizations and SCOs.

This SCORM specification makes the process of finding and reusing a resource more efficient by providing a structure of defined elements that describe the learning resource for reusability. Most elements in the metadata hierarchy have a specific definition, data type, and either a vocabulary or defined content.

When creating your metadata, always remember the target audience, knowledge base, search habits, and search vocabulary of those seeking content, and develop metadata with these things in mind. This includes using the most specific and descriptive terms possible.

The description and keywords will be useful for other individuals searching online content repositories for relevant content to use in their own courses.

### 6.1 Army Metadata Requirements



**6.1-1 Business Rule (Army): Metadata is required for all Content Organizations and for all SCOs. All metadata is in accordance with the instructions in this section and Figure 9.25b.**

The Army requires two types of metadata: Content Organization Metadata and SCO Metadata.

### **6.1.1 Content Organization Metadata**

Content Organization metadata is located in the <organization> section within the manifest file. It describes and provides information about the course structure in the organization section as a whole. The Army requires this type of metadata.

### **6.1.2 SCO Metadata**

SCO metadata is the context-independent metadata associated with each SCO. The Army requires this context-independent metadata, and this file contains information about the content of the SCO.

A software tool is highly recommended for the automatic creation of metadata .xml files. The Government has developed a Metadata Editor tool that is now in the public domain. To download this tool, visit [Testing tool webpage](#) .

## **6.2 Developing the SCORM Metadata Files**

To develop your metadata, see Appendix B for a table listing the metadata required by the Army for each course and each SCO. This table can be used by the training developer to fill out required data for the programmer's use. Also included is a populated example table of required metadata.

### **6.2.1 Catalog Identifier and Entry Identifier**

The catalog and entry identifiers are values that remain the same for all courses. The catalog is "ATIA" and the entry is "TBD".

### **6.2.2 Title of Learning Resource**

The title of the learning resource is designated within the metadata. This title reflects the title at whatever level of instruction.

### **6.2.3 Language of Learning Resource**

The language of the learning resource content is specified with the two-character country code specified in universal standards.

### **6.2.4 Description of Learning Resource**

A full description of the learning resource is necessary to provide information to determine whether the instructional object would meet the qualifications. The description should exclude any reference to hierarchy (lesson, phase, module, etc.).

### **6.2.5 Keywords**

Select keywords or phrases that accurately and precisely define the object.



**Best Practice:** The keywords most relevant should be listed first in the 'general.keyword' metadata element.

### **6.2.6 Type of Metadata**

Different types of metadata files are normally not able to be differentiated from each other. This impacts testing because the proper requirements may not be applied. Therefore, the Army requires the 'type' of metadata contained in each metadata file to be designated within the file so that testing software can determine and apply the proper metadata test. The type of metadata should be designated as a "2" for a SCO or "3" for a Content Organization.

### **6.2.7 Version of Learning Resource**

The version of learning resource metadata element describes the edition of the learning resource. The initial version will be 1.0. Corrections will change minor version number; enhancements/updates will change major version number. This element upon initial final delivery to the Army must contain the value "1.0".

### **6.2.8 Status of Package Submittal**

The status of package submittal describes the status of the submission. For example, this element must contain the word "final" if the object was approved as final.

### **6.2.9 Proponent's Role**

The role of the proponent as contributor is designated as "publisher".

### **6.2.10 Name, Address, and E-mail of Proponent**

The proponent's name, address, school code, and e-mail address should be included in the metadata to identify the organization contributing to this course and to facilitate search criteria.

### **6.2.11 Date of Submittal**

The date that the course was submitted should be included in the metadata.

### 6.2.12 Metadata Catalog and Entry Identifier

This element represents a mechanism for assigning a globally unique label that identifies the metadata record describing the SCORM Content Model Component. Catalog represents the name or designator of the identification or cataloging scheme for the entry. Entry represents the value of the identifier within the identification or cataloging scheme that designates or identifies the metadata.

### 6.2.13 Metadata Schema

This element identifies the name and version of the authoritative specification used to create this metadata instance. The Army requires three elements: (1) an element with a value of "LOMv1.0", (2) an element with the value of "SCORM\_CAM\_v1.3", and (3) an element with the value of "ADLv1.0".



**Note:** Requiring these three values will ensure that Army metadata will be valid under both SCORM 2004 2<sup>nd</sup> Edition and SCORM 2004 3<sup>rd</sup> Edition.

### 6.2.14 Language of Metadata File

The language of the metadata file is designated within the metadata file to indicate the language of all content within the metadata file. If this value is provided, then it is not necessary to designate the language for each of the other information.

### 6.2.15 File Formats

File formats which are contained within the learning resource should be identified to obtain the proper plug-ins required for the course to play properly in the LMS. For example, a SCO may contain HTML files, graphic files (.gif, .jpg), Macromedia Flash files (.swf), etc.

### 6.2.16 Cost of Learning Resource

The cost of learning resource indicates whether use of the resource requires payment. This element must have the 'source' element equal to "LOMv1.0" and the 'value' element equal to "no".

### 6.2.17 Copyright and Other Restrictions

The copyright and other restrictions element indicates whether copyright or other restrictions apply to the use of this resource. This value must be the 'source' element equal to "LOMv1.0" and the 'value' element equal to "no".

## 6.2.18 Classification

Classification of the learning resource describes where this resource is placed within a particular classification system and is also specified for search criteria by designating the following to be contained in the metadata associated with the SCO or course. Classification is a repeating field that allows multiple entries. Classification includes items such as MOS and skill level, SQI, ASI, task numbers and task descriptions, learning objectives (action, condition, and standard), 508 compliance, security level (foreign disclosure), and collection (DoD).

Note: The ADL-R currently only supports the **unclassified** security marking.

### Policy on Tagging SCORM Objects with Foreign Disclosure (FD) Restrictions

The rules should address the Army's learning hierarchy and be consistent with the published guidance.

1. For automated presentation/restriction/sharing of material, FD restrictions are determined and applied at the lesson level. For completeness in content description (for example, informational purposes), XML tags will include 'course FD' restrictions; or the words 'Public Domain', which will allow the deliver and release of that object to anyone without restriction.
2. All objects will have a FD restriction statement or 'Public Domain' assigned to them in accordance with the rules below and the matrices above.
3. If the lesson (and all SCOs/assets in the lesson) has no Controlled Unclassified Information (CUI) or Classified Military Information (CMI), then the lesson is labeled 'Public Domain'; if the lesson contains CUI/CMI, then the FD description that fits the lesson is assigned (and all parts of the lesson are assigned that designator). If all lessons in a module are 'Public Domain' (for example, no CUI/CMI) then the module is 'Public Domain'. If all lessons in a phase are 'Public Domain', then the phase is 'Public Domain'. If all lessons in a course are 'Public Domain', then the course is 'Public Domain'.
4. For course/phase level (in accordance with Army guidance) aggregations, use FD restrictions 1-4 with default to FD1. For module-level, lesson-level aggregations, SCOs, and assets use FDs restrictions 5-7 (with default to FD5).

Note: While restrictions are initially determined at lesson level, in general, it is NOT correct to think that these lesson-level restrictions can be 'rolled-up' to higher levels. If we did so, we would unnecessarily restrict access to lessons that have no access restrictions by the FD rules.

5. Specific Rules: See Figure 6.2.18.1a.
6. ALL objects within a lesson (SCOS/Assets) will have the same FD level assigned as that of the lesson aggregation of which they are a part (for example, roll-down lesson FD to all objects within lesson). In particular, a lesson which is determined to be FD7, all objects within that lesson should be FD7. We cannot/will not try to differentiate smaller objects that may/or may not have CUI information. See Figure 6.2.18.1b.
7. If the proponent determines that lack of foreign learner access to a restricted lesson will make worthless the training provided in the entire module, the proponent will use the FD restrictions 6 or 7 at module level to indicate this (use of 6 or 7 is determined by using the FD restriction of the most restrictive lesson within that module). If the Foreign learner can profit from the contents of the module without the lesson(s) that he cannot take, the module will be tagged as FD5 (and FD access restrictions will be determined at lesson level).

8. Stand-alone training products (for example, a lesson, module, or group of modules) will be tagged with the appropriate restrictions using FD5-7 or 'Public Domain'.

**Learning Object FD Code Assignment Table 1\*, Ver 1.0, April. 03**

Purpose: Implementation of these codes will allow appropriate delivery, avoid FD delivery violations, and maximize reusability.

Rule of Thumb: A code assigned to an aggregation object means all objects within that aggregation is restricted in the same way (although the phase and course codes differ from the module, lesson, and lower codes).

<b>If</b>	<b>Then</b>	<b>Then</b>	<b>And</b>	<b>Notes/Caveats</b>
Any SCO or asset within a lesson contains CUI/CMI.	The lesson must have an FD designator.		All other parts of the lesson have the same FD designator as that assigned to the lesson.	Lesson FD roles down to all parts.
There is no CUI/CMI in an object.	That object is 'Public Domain'.			A lesson, module, phase, or course might be 'Public Domain' if ALL of their parts are 'Public Domain.'
There is CUI/CMI in at least one lesson.	The rules for FD assignment below apply to that lesson, its parts, and aggregates.			
All lessons in the course are FD5.	Module FD code is FD5.	All phases have code FD1.	Course Assignment code is FD1.	Best and most likely case.
Any lesson in any one module is FD7.	The module can be either FD 5, 6, or 7 as appropriate (no rollup).	The phase containing the module can be FD2, 3, or 4, as appropriate. <u>Cannot be FD1.</u>	Course assignment can be FD 2, 3, or 4. Cannot be FD1	All other modules/phases are determined by their lessons' FD #.
All lessons in a module are FD7.	The module is FD7.	The phase containing the module can be FD2, 3, or 4. <u>Cannot be FD1.</u>	Course assignment can be FD 2, 3, or 4. <u>Cannot be FD1.</u>	Will occur when the CUI is concentrated in a single module.
All lessons in all the modules in a single phase are FD7.	All modules within that phase are FD7.	The particular phase is FD4. Other phases are whatever is appropriate.	Course can be FD 2, 3, or 4. <u>Cannot be FD1.</u>	Will occur when all the CUI is concentrated in a single phase of a course.
All lessons in all the modules in all phases are FD7.	All modules within all phases are FD7.	All phases in the course are FD3.	The course is FD3.	Will occur when an entire course is not releasable to any foreign learners.
Any lesson is FD6.	Module may be FD6 or 7, whichever is appropriate. <u>Cannot be FD5.</u>	1. The phase containing any FD6 lesson may be FD2, FD3, or FD4 depending on the other lessons in the phase. 2. The phase cannot be FD1.	The course can be FD 2, 3, or 4 based upon the other lessons in the course.	Will occur when all the CUI is concentrated in a number of lessons of a course and those lessons are releasable to some foreign learners. Other lessons may be more, less, or of equal FD restriction.
All lessons in module are FD6.	The Module is FD6.	1. The phase containing FD6 lessons is not less than FD2. 2. The phase may be FD3 or FD4 depending on the other modules in that phase.	The course can be FD2, 3, or 4 based upon the other lessons in the course.	Will occur when all the CUI is concentrated in a single module of a course and releasable to some foreign learners.
All lessons in all modules in all phases are FD6.	Module is FD6.	All phases are FD2.	The course is FD2	Will occur when all lessons contain CUI and the course is releasable to some foreign learners.

\* Foreign Disclosure Restrictions are part of TRADOC Regulation 350-70 Appendix I-1-3, Page 106

Figure 6.2.18.1a

<b>Learning Object FD Code Assignment Table 2, (SCOs and Assets), Ver 1.0, Apr 03</b>	
<b>If Lesson FD Assignment is</b>	<b>Then all SCOs within that lesson <u>must be</u> and are assigned</b>
Public Domain	Public Domain
FD5	FD5
FD7	FD7
FD6	FD6

Figure 6.2.18.1b

## **7. Army SCORM and Playability Testing of Courseware**

The procedures and programs used in the Army SCORM testing process can be obtained from the Army in a document entitled 'U.S. Army Acceptance Criteria for Continuous Testing of SCORM 2004 Conformant Courseware'. For a complete discussion of SCORM 2004 courseware testing, visit [ATSC Acceptance Criteria](#)

### **7.1 SCORM References and Tutorials/Online Courses**

For links to SCORM tutorials/online courses, visit [SCORM Training Resources](#) .

### **7.2 SCORM Tools**

[Army Metadata Editor \(Public Domain\)](#)

[Army Manifest Auditor \(Public Domain\)](#)

[Army SCORM 2004 Sequencing Templates \(AKO Sign-on required\)](#)

[Army SCORM Wire-Frame Example \(AKO Sign-on required\)](#)

[Army Log Parser \(Public Domain\)](#)

[Army Resource Validator \(Public Domain\)](#)

## 8. Glossary and Acronyms

Acronyms/Terms	Definitions
AAR	After Action Review
ADL	Advanced Distributed Learning. The ADL Initiative, sponsored by the Office of the Secretary of Defense (OSD), is a collaborative effort between government, industry and academia to establish a new dL environment that permits the interoperability of learning tools and course content on a global scale. ADL has developed the SCORM.
ADL-R	ADL Registry. The ADL-R is not a repository of content; it is a searchable index of content metadata that can be resolved to content located in distributed repositories.
ALMS	The Army's Learning Management System provides a suite of capabilities that integrate training and education support capabilities and enhance training and education. It is a Web-based information system that delivers training to Soldiers, manages training information, and provides training collaboration, scheduling, and career planning capabilities in both resident and non-resident training environments. It consists of Saba® Enterprise 5.3 Learning Suite, Blackboard® Academic Suite, CENTRA, and Adobe® Acrobat Connect.
ALO	ALO is a concise way in ANSI C code to write data in XML format to a file. The ALO API handles XML structure, XML syntax, and data formatting so your code can focus on what data to output. ALO is pronounced like 'A low'.

Acronyms/Terms	Definitions
API	Application Programming Interface. The API enables the communication of data between content and the RTE provided by an LMS. The use of a common API fulfills many of SCORM's high-level requirements for interoperability and reuse.
AR(s)	Army Regulation(s)
ASI	Additional Skill Identifier
Asset	An asset is the smallest, atomic, meaningful learning content object used to develop Web-based IMI training to include the raw media (separate and distinct instructional text, audio, video, graphic, animation, etc.) used to create a Sharable Content Object (SCO). Also referred to as a 'resource'.
ATRRS	Army Training Requirements and Resources System. Legacy system that tracks quota and non-quota managed training
ATSC	Army Training Support Center
Block	A term used in a previous SCORM standard (SCORM v1.1), which maps to a Content Aggregation in SCORM v1.2 and a cluster in SCORM 2004.
CAM	Content Aggregation Model. One of the SCORM books provided on ADL's Web site.
CBT	Computer-based Training
Cluster	 <b>ADL reference:</b> An activity and its children, but not the descendants of its children; activity that contains other activities.
CMI	Classified Military Information
Content Aggregation	A collection of SCOs.
Content SCO	A SCO that does not contain a graded assessment

<b>Acronyms/Terms</b>	<b>Definitions</b>
Context-specific	Specific to the context of a particular structure or situation.
CORDRA	Content Object Repository Discovery and Registration Architecture. CORDRA is designed to be an enabling model to bridge the worlds of learning content management and delivery, and content repositories and digital libraries. CORDRA aims to identify and specify (not develop) appropriate technologies and existing interoperability standards that can be combined into a reference model used to enable a learning content infrastructure.
CUI	Controlled Unclassified Information
Data Model	A set of data elements that ensures a defined set of information about SCOs can be tracked by different LMSs.
Dependent	Relying on another for support, aid, etc; Determined or conditioned by another.
Dependent SCO	A SCO that is 'not independent' and is context-specific and does not have a high degree of reusability (for example, an Introduction SCO for a module or group of lessons). A SCO that pertains to the particular course structure.
dL	Distributed Learning
DOM	Document Object Model
ELO	Enabling Learning Objective
Embedded Learner Performance Test	A graded assessment contained in the same SCO as learning content.
External References	Content that the learner is not tested. Documents that would invite further study, research, or information on the subject matter.
FD	Foreign Disclosure
FM(s)	Field Manual(s)
GFI	Government Furnished Information
Graded Assessment	The act of evaluation resulting in a score or grade.

Acronyms/Terms	Definitions
Granularity	'Becoming granular' or 'showing a granulated structure'. SCORM recommends small particles by reducing the size of learning objects for reusability and enhancing the learning experience.
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICW	Interactive Courseware
IEEE	Institute of Electrical and Electronics Engineers (pronounced eye-triple-e) is responsible for the standardized advancement of technology related to electricity.
IMDP	Instructional Media Design Package
IMI	Interactive Multimedia Instruction
imsmanifest.xml	An xml file that is stored in the root of a content package that describes the entire package.
Incremental Fielding	Lessons are delivered as they are completed, allowing the learner to take advantage of available content without waiting for an entire course to be delivered.
Independent SCO	A SCO that teaches a complete learning thought. An independent SCO must not be dependent on any other learning content, must be 'context independent' and must stand alone.
ISO	International Organization for Standardization is an international standard-setting body composed of representatives from various national standards bodies.
Launchable Asset	An activity can be developed as a launchable asset when learner tracking is unnecessary.
Leaf Activity	In a SCORM activity tree, activity that does not consist of another activity.

Acronyms/Terms	Definitions
Learner Performance Test	Any graded assessment of learner performance such as a pretest or posttest.
Learning Content	Content that is germane to the instruction and on which the learner will be tested.
Learning Content Package	A SCORM package that contains the learning content SCOs and other SCOs where completion of the instruction is emphasized.
Learning Steps	A learner activity that leads toward achievement of a learning objective. Learning steps are determined when the objective is broken down into its component parts. Often an explicit hierarchical relationship consisting of a terminal learning objective, an enabling learning objective, and learning step.
LOM	Learning Object Metadata. The root element or the beginning of the SCORM metadata XML record, for example <lom>.
LMS	Learning Management System. Any SCORM conformant LMS or the LMS yet to be chosen by the Army to play SCORM 2004 courseware.
Manifest	A common name that refers to the imsmanifest.xml file, which is an essential part of all SCORM Content Packages. This xml file is stored in the root of a SCORM content package and describes the contents of the entire package.
Menu	A list of available topics or available branching within a SCO.
Metadata	In SCORM, data about the resources contained in Web-based courseware; used for searching.
MIME	Multipurpose Internet Mail Extensions
MOS	Military Occupational Specialty
Non-learning content	Content that does not teach, but may inform, direct, explain, or provide understanding.

Acronyms/Terms	Definitions
Objective Definitions Page	A page within the SCO that displays the Learning Objectives Actions, Conditions, and Standards.
Offline Player	An application that plays SCORM conformant courseware outside of a SCORM conformant LMS.
Pam(s)	Pamphlet(s)
PIF	Package Interchange File. This file is a representation of the content package components using the PKZIP Version 2.04g archive format (zip).
POI	Program of Instruction
Reg(s)	Regulation(s)
Repository	A digital data storage warehouse where learning objects may be accumulated and cataloged for broad distribution and use.
Resource(s)	Files required for the courseware to play. These files can consist of HTML, Flash, graphics, audio, or video files.
RFC(s)	The Request(s) for Comment(s) form an official series of notes, started in 1969, about the Internet. The notes discuss many aspects of computer communication, focusing on networking protocols, procedures, programs, and concepts but also including meeting notes, opinions, and sometimes humor.
Rollup	A single result from many. For example, a rollup of scores is an average. A rollup of completion status is a determination of all sub-activities of a cluster. If there is one SCO that is 'incomplete', then the rollup is 'incomplete'. If all SCOs are 'completed', then the rollup is 'complete'.
RTE	Run-Time Environment
Schema	An XML file that contains programming rules and allowable XML tags that define the structure and content of other XML files.

Acronyms/Terms	Definitions
SCO	Sharable Content Object. A SCO must reference a specific launch URL that utilizes the SCORM Run-Time Environment (RTE) to communicate with LMSs. A SCO is independent of context and able to stand alone when extracted from the package for reuse.
SCORM	Sharable Content Object Reference Model
SME	Subject Matter Expert
SOW	Statement of Work
SQI	Skill Qualification Identifier
SS&N	Simple Sequencing and Navigation
Sustainment	Sustainment is a term used to indicate that the learner re-accesses the learning content after the learner has completed a graded assessment.
TADLP	The Army Distributed Learning Program
TBD	To Be Developed
TLO	Terminal Learning Objective
TOC	Table of Contents. The main list of available learning content for a course, phase, module, or lesson that describes the structure and behavior of the content. In SCORM, the TOC is defined in the manifest file and can only show the structure of a package.
TR(s)	TRADOC Regulation(s)
TRADOC	Training and Doctrine Command
TSP	Training Support Package
TWG	Technical Working Group
UI	User Interface
URI	Uniform Resource Identifier. A URI is a compact sequence of characters that provides a simple and extensible means for identifying a resource.
URL	Uniform Resource Locator
URN	Uniform Resource Name

<b>Acronyms/Terms</b>	<b>Definitions</b>
vCard	The generic term for an electronic, virtual information card that can be transferred between computers, PDAs, or other electronic devices through telephone lines, or e-mail networks, or infrared links. vCard is a metadata tag required by the Army.
www	World Wide Web
XML	eXtensible Markup Language

## 9. Programming Examples

This section covers SCORM specifications for the programmer that are specific to Army SCORM implementation.



**ADL Reference:** The SCORM Run-Time Environment (RTE) Data Model contains a set of data model elements that can be tracked by the SCO with an LMS during the run-time of the SCO. These elements can be used to track items, i.e. status, scores, interactions, objectives, etc. Some of the RTE data model elements impact one another or are used with others. Some of the data model elements, if used, impact the control and sequence of other SCOs being used in the same context, (for example lesson or course).

Some elements that are identified as trackable by SCORM are presented as outside the scope of SCORM. The handling of these elements may be possible by some LMSs, but not by others. These elements that are outside the scope of SCORM are listed below, but are LMS specific, and as such, usage should be avoided without thorough analysis of ALMS support and implementation.

- `cmi.comments_from_lms.n.comment`
- `cmi.comments_from_lms.n.location`
- `cmi.comments_from_lms.n.timestamp`

Note: The LMS may provide a way to allow the author/instructor to provide comments (along with a location and date); these comments may then be used to initialize these values.

- `cmi.credit`

Note: SCORM does not require that an LMS support a mechanism for prescribing the credit state for a SCO.

- `cmi.learner_preferences.language`
- `cmi.learner_preferences.delivery_speed`

Note: The LMS may not provide a mechanism for initializing these values.

- `cmi.mode`

Note: There is no mechanism in place to determine the mode of a SCO. It is currently left to the implementation of the LMS.

## 9.1 XML Examples on the Manifest

### 9.1.1 Course Title Example

Following is an example of the course title (or the instructional level at which it is packaged) as listed in the <title> tag of the parent element <organizations> within the manifest:

```
<organizations default="Course01">
  <organization identifier="Course01"
    adlseq:objectivesGlobalToSystem="false">
    <title>Course Title goes here</title>
    <item>...</item>
    ...
  </organization>
</organizations>
```

Figure 9.1.1a

## 9.1.2 SCO Example

Following is an example of the XML tagging for a SCO within the manifest with a SCO defined by the 'scormType' attribute on the <resource> element. The 'scormType' must have the value of "sco". File references must be relative paths from the location of the manifest file.

```
<organizations default="TOC1">
  <organization identifier="TOC1"
    adlseq:objectivesGlobalToSystem="false">
    <item identifier="S100" identifierref="R_S100">
      <title>SCO Title</title>
      <imsss:sequencing>...</imsss:sequencing>
    </item>
    ...
  </organization>
</organizations>

<resources>
  <resource identifier="R_S100" type="webcontent"
    adlcp:scormType="sco" href="unit1/index.html">
    <metadata>
      <adlcp:location>metadata/unit1_sco.xml</adlcp:location>
    </metadata>
    <file href="unit1/index.html"/>
    <file href="unit1/scopage2.html"/>
    <file href="unit1/scopage3.html"/>
    <file href="images/graphic1.gif"/>
  </resource>
  ...
</resources>
```

Figure 9.1.2a

## 9.1.3 Cluster Example

XML coding is used to define a cluster within the manifest file. A cluster is defined within the manifest file by using the <item> tag without the 'identifierref' attribute, because a cluster does not contain resources. This <item> tag is just a parent container for other <item> elements.

Note: The term of 'cluster' has replaced the SCORM v1.2 term of 'aggregation' and the SCORM v1.1 term of 'block'.

Following is the XML coding in the manifest for one cluster with three SCOs:

```
<manifest>
...
  <organization identifier="TOC1"
    adlseq:objectivesGlobalToSystem="false">
    ...
      <item identifier="C100">          (opening tag for a cluster)
        <title>Cluster Title</title>
        <item identifier="SCO101" identifierref="R_SCO101"> (SCO)
          <title>Introduction to TLO</title>
        </item>
        <item identifier="SCO102" identifierref="R_SCO102"> (SCO)
          <title>Title of SCO 1</title>
        </item>
        <item identifier="SCO103" identifierref="R_SCO103"> (SCO)
          <title>Title of SCO 2</title>
        </item>
      </item>          (closing tag for a cluster)
    ...
  </organization>
...
</manifest>
```

Figure 9.1.3a

#### 9.1.4 Launchable Asset Example

Following is an example of XML code for a launchable asset contained within the manifest:

```
<organization identifier="TOC1"
adlseq:objectivesGlobalToSystem="false">
...
  <item identifier="Item_10" identifierref="R_Item10">
    <title>Course Map</title>
  </item>
...
</organization>

<resources>
...
  <resource identifier="R_Item10" type="webcontent"
adlcp:scormType="asset" href="coursemap/cm.html">
    <file href="coursemap/cm.html"/>
    <file href="coursemap/cm2.html"/>
    <file href="coursemap/cm.jpg"/>
  </resource>
...
</resources>
```

Figure 9.1.4a

The above launchable asset example shows the course map as an item in the <organization> (TOC) pointing to resources of type 'asset' with an entry point or launch file (href).

Note: LMS User Interface buttons cannot be hidden for launchable assets.

Note: Complete information concerning various methods of hiding launchable assets/resources from the TOC has not been tested in live SCORM 2004 LMSs.

### 9.1.5 ADL Extensions to the Manifest File

Specific SCO data can be designated on the manifest file external to the SCO so that the data can be easily updated when the SCO is reused. SCORM provides this capability known as the ADL Extensions. External data will facilitate reusability because there is no need to re-author content just to change certain values such as the mastery score or the maximum time allowed.

The following describes and provides an example of the ADL extensions to the IMS specifications contained within the manifest file:

Manifest Tag	Description
adlcp:timeLimitAction	What action to take when the maximum time allowed is exceeded.
adlcp:dataFromLMS	Unique information generated at SCO creation that is needed for every use.
adlcp:completionThreshold	Percent Complete. Used to determine whether the SCO should be considered complete.

Figure 9.1.5a

Example of the ADL extensions for one SCO within the manifest file:

```

<organization identifier="TOC1"
adlseq:objectivesGlobalToSystem="false">
...
  <item identifier="S1101" identifierref="R_S1101">
    <title>SCO Title</title>
    <adlcp:timeLimitAction>exit,message</adlcp:timeLimitAction>
    <adlcp:dataFromLMS>3943,3,a,1</adlcp:dataFromLMS>
    <adlcp:completionThreshold>.60</adlcp:completionThreshold>
  </item>
...
</organization>

```

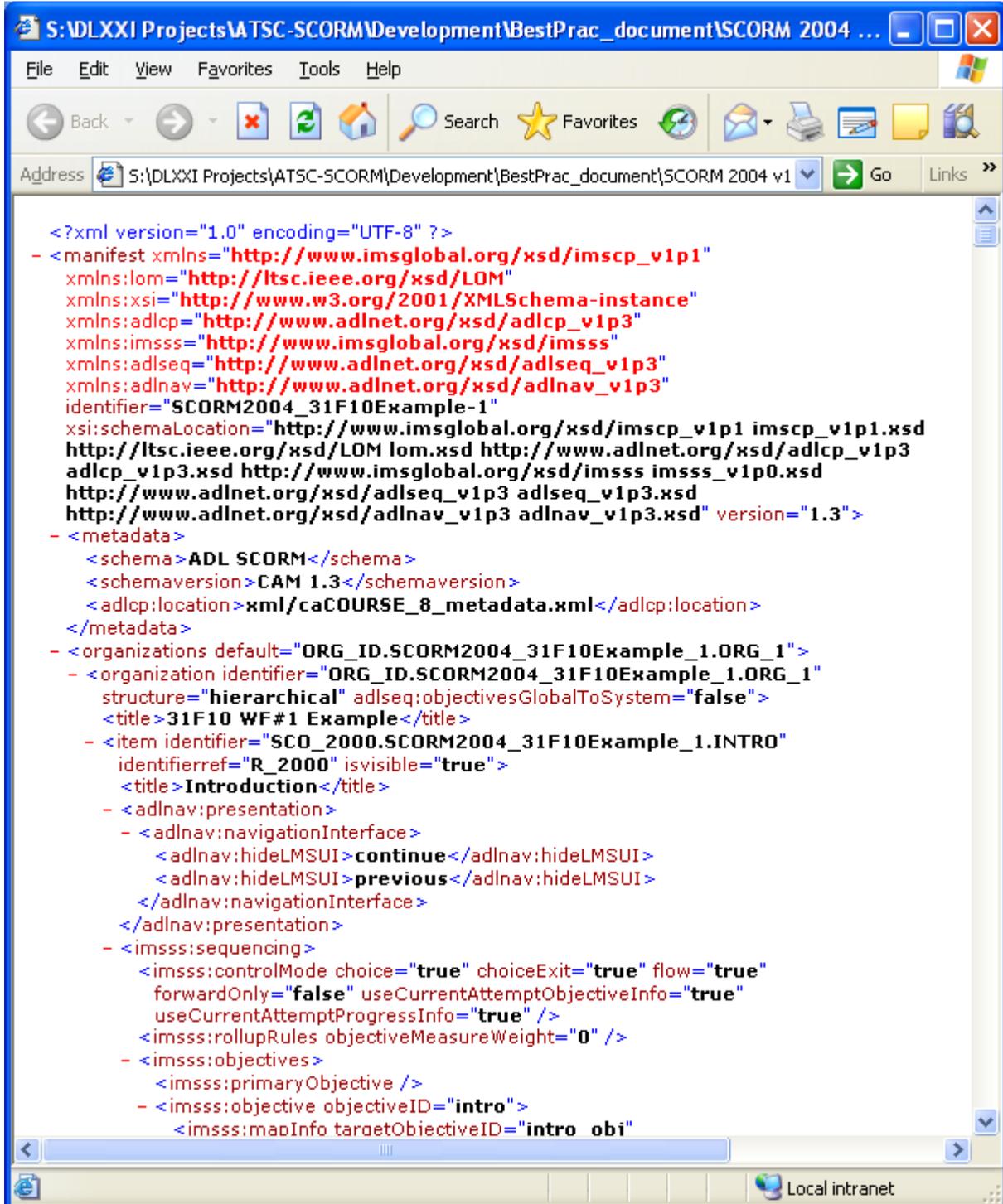
Figure 9.1.5b

Note: Within the <item> tag, ADL extensions (adlcp:xxx) are listed before the <imsss:sequencing> tag in order to validate against the schemas.

### **9.1.6 Screenshot of Manifest File**

The <organizations> section describes one or more content aggregations represented by the <organization> tag. Each <organization> tag specifies a distinct content structure such as a table of contents.

The XML tagging within the manifest file is partially shown in the following example



```
<?xml version="1.0" encoding="UTF-8" ?>
- <manifest xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
  xmlns:lom="http://ltsc.ieee.org/xsd/LOM"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3"
  xmlns:imsss="http://www.imsglobal.org/xsd/imsss"
  xmlns:adlseq="http://www.adlnet.org/xsd/adlseq_v1p3"
  xmlns:adlnav="http://www.adlnet.org/xsd/adlnav_v1p3"
  identifier="SCORM2004_31F10Example-1"
  xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1.xsd
  http://ltsc.ieee.org/xsd/LOM lom.xsd http://www.adlnet.org/xsd/adlcp_v1p3
  adlcp_v1p3.xsd http://www.imsglobal.org/xsd/imsss imsss_v1p0.xsd
  http://www.adlnet.org/xsd/adlseq_v1p3 adlseq_v1p3.xsd
  http://www.adlnet.org/xsd/adlnav_v1p3 adlnav_v1p3.xsd" version="1.3">
- <metadata>
  <schema>ADL SCORM</schema>
  <schemaversion>CAM 1.3</schemaversion>
  <adlcp:location>xml/caCOURSE_8_metadata.xml</adlcp:location>
</metadata>
- <organizations default="ORG_ID.SCORM2004_31F10Example_1.ORG_1">
- <organization identifier="ORG_ID.SCORM2004_31F10Example_1.ORG_1"
  structure="hierarchical" adlseq:objectivesGlobalToSystem="false">
  <title>31F10 WF#1 Example</title>
  - <item identifier="SCO_2000.SCORM2004_31F10Example_1.INTRO"
    identifierref="R_2000" isVisible="true">
    <title>Introduction</title>
    - <adlnav:presentation>
      - <adlnav:navigationInterface>
        <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
        <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
        </adlnav:navigationInterface>
      </adlnav:presentation>
    - <imsss:sequencing>
      <imsss:controlMode choice="true" choiceExit="true" flow="true"
        forwardOnly="false" useCurrentAttemptObjectiveInfo="true"
        useCurrentAttemptProgressInfo="true" />
      <imsss:rollupRules objectiveMeasureWeight="0" />
    - <imsss:objectives>
      <imsss:primaryObjective />
      - <imsss:objective objectiveID="intro">
        <imsss:mapInfo targetObjectiveID="intro obi">
```

Figure 9.1.6a

## 9.1.7 Manifest in Detail

The following describes the manifest in detail with examples of the XML code. Descriptions are in the left column and individual parts of the manifest are contained in the right column.

All identifiers must be unique within the manifest and provided by the author per SCORM specifications.

<p>This part always stays the same except for the identifier. These attributes declare schemas and namespaces. The physical Schema files are located in the root of the content package.</p>	<pre>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes" ?&gt; &lt;manifest version="1.3" identifier="manifest1" xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1.xsd http://www.adlnet.org/xsd/adlcp_v1p3 adlcp_v1p3.xsd http://www.imsglobal.org/xsd/imsss imsss_v1p0.xsd http://www.adlnet.org/xsd/adlseq_v1p3 adlseq_v1p3.xsd http://www.adlnet.org/xsd/adlnav_v1p3 adlnav_v1p3.xsd http://ltsc.ieee.org/xsd/LOM lom.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3" xmlns:imsss="http://www.imsglobal.org/xsd/imsss" xmlns="http://www.imsglobal.org/xsd/imscp_v1p1" xmlns:adlseq="http://www.adlnet.org/xsd/adlseq_v1p3" xmlns:adlnav="http://www.adlnet.org/xsd/adlnav_v1p3" xmlns:lom="http://ltsc.ieee.org/xsd/LOM"&gt;</pre>
<p>Required metadata designator tags</p>	<pre>&lt;metadata&gt; &lt;schema&gt;ADL SCORM&lt;/schema&gt; &lt;schemaversion&gt;2004 3rd Edition&lt;/schemaversion&gt; &lt;/metadata&gt;</pre>
<p>&lt;organizations&gt; tag describes one or more &lt;organization&gt; structures. The default value identifies which organization to launch for the default.</p>	<pre>&lt;organizations default="COURSE_1"&gt;</pre>
<p>Title of package goes here</p> <p>This is where the table of contents begins using the &lt;title&gt; tags to create the table of contents.</p>	<pre>&lt;organization identifier="COURSE_1" adlseq:objectivesGlobalToSystem="false"&gt; &lt;title&gt;Package Title&lt;/title&gt;</pre>

<p>Note: SCORM specifications have the default manifest global objectives mapped to system level. Since the Army only allows mapping to the manifest level, the system level objectives must be set to false. See Business Rule 4.1.2-1 for more information on global objectives.</p>	
<p>Dependent SCO</p>	<pre>&lt;item identifier="SCO_480" identifierref="R_421"&gt;   &lt;title&gt;Introduction to Instruction&lt;/title&gt; &lt;/item&gt;</pre>
<p>Clusters use &lt;item&gt; tag containing other &lt;item&gt; tags. No file resources for clusters</p>	<pre>&lt;item identifier="B_154"&gt;   &lt;title&gt;Action Statement of the TLO&lt;/title&gt;</pre>
<p>Independent SCO. (Uses the &lt;item&gt; tag Notice the identifierref. This &lt;item&gt; references resources (files) that are used in the SCO</p>	<pre>&lt;item identifier="SCO_521" identifierref="R_S521"&gt;   &lt;title&gt;Introduction to the TLO&lt;/title&gt; &lt;/item&gt;</pre>
<p>Another SCO reference (not expanded below in &lt;resources&gt;)</p>	<pre>&lt;item identifier="SCO_520" identifierref="R_520"&gt;   &lt;title&gt;ELO Title&lt;/title&gt; &lt;/item&gt;</pre>
<p>Another SCO (not expanded below in &lt;resources&gt;)</p>	<pre>&lt;item identifier="SCO_480" identifierref="R_480"&gt;   &lt;title&gt;ELO Title&lt;/title&gt; &lt;/item&gt;</pre>
<p>Sequencing code and closing item tag for Cluster</p>	<pre>&lt;imsss:sequencing&gt;   &lt;imsss:controlMode choice="true"/&gt; &lt;/imsss:sequencing&gt; &lt;/item&gt;</pre>

Content Organization Metadata reference for <organization> (metadata in separate file)	<metadata> <adlcp:location>metadata/ca_course.xml</adlcp:location> </metadata>
Sequencing code for organization	<imsss:sequencing> <imsss:controlMode choice="true" flow="true"/> </imsss:sequencing>
Closing tag for Organization	</organization>
Closing tag for Organizations	</organizations>
END TABLE OF CONTENTS and BEGINNING OF COURSEWARE PHYSICAL FILE REFERENCES	
Opening tag for Resources	<resources>
Opening tag for Resource (This is a SCO)	<resource identifier="R_S521" type="webcontent" adlcp:scormType="sco" href="unitsafety/index.html">
Assets for the SCO	<file href="unitsafety/index.html"/>
File tag for Asset (SCO Launch file must be included in a <file> tag.)	
Another file Asset	<file href="unitsafety/us0010.html"/>
Tag for Asset that this SCO is dependent on	<dependency identifierref="A_137"/>

Closing tag for R_S521 Resource	</resource>
Opening tag for Resource (This is an asset referenced as a dependency.)	<resource identifier="A_137 type="webcontent" adlcp:scormType="asset">
File tag for Asset	<file href="images/graphic1.gif"/>
Another Asset	<file href="images/graphic2.gif"/>
Another Asset	<file href="images/movie1.swf"/>>
Closing tag for A_137 Resource	</resource>
Closing tag for Resources	</resources>
Closing tag for Manifest	</manifest>

Figure 1.1.7a

## 9.2 Sequencing and Navigation Examples

Sequencing tags are contained only within the <organization> section of the manifest. The following examples are only snippets.

### 9.2.1 Sequencing Overview

Below is a complete list of parent sequencing tags within a manifest file.

```
<imsss:sequencing>
  <imsss:controlMode/>
  <imsss:sequencingRules>...</imsss:sequencingRules>
  <imsss:limitConditions/>*
  <imsss:auxiliaryResources/>*
  <imsss:rollupRules>...</imsss:rollupRules>
  <imsss:objectives>...</imsss:objectives>
  <imsss:randomizationControls>...</imsss:randomizationControls>
  <imsss:deliveryControls/>
  <adlseq:constrainedChoiceConsiderations/>
  <adlseq:rollupConsiderations/>
</imsss:sequencing>
```

\* ADL recommends using with caution.

Figure 9.2.1a

Below is the default functionality that every course will have *unless turned off*:

1. Any mapped global shared objectives defined in sequencing are global for the lifetime of the learner within the LMS across all Content Organizations. Note: LMS-level global objectives must not be used at the time of this writing. See Business Rule 4.1.2-1.

2. Objective Status information for an activity's objectives is local to that activity.
3. Control Modes of Choice, Choice Exit, Use Current Attempt Objective Information, Use Current Attempt Progress Information.
4. Rollup Conditions of all child activity sets are considered in rollup evaluation of cluster.
5. Rollup Controls of Rollup Objective Satisfied (activity contributes to the evaluation of its parent's "satisfied" and "not satisfied" rollup rules) and Rollup Progress Completion (activity contributes to the evaluation of its parent's "completed" and "not incomplete" rollup rules).
6. Delivery Controls – Tracked (progress information will be recorded and contribute to rollup).
7. Rollup Rules – If ANY rule condition evaluates to true, the cluster will evaluate to true (Condition Combination attribute).
8. Rollup Child Activity Set – ALL tracking status information of a cluster is considered in the rollup.
9. Rollup Actions – The default desired action that is applied to the cluster activity that defines the Rollup Rule is "satisfied".
10. Rollup Controls – The activity's tracking status information will be applied to its parent's Rollup Rules, Rollup Objective Satisfied, and Rollup Progress Completion.

### 9.2.2 Control Mode Example

Control Mode determines how the learner interacts with the learning activities, which determines how sequencing requests are applied to clusters. The following are sequencing control modes that may be applied:

Name	Description	Default Value
Sequencing Control Choice	Indicates that a Choice navigation request is permitted to target the children of the activity.	True
Sequencing Control Choice Exit	Indicates that the activity is permitted to terminate if a Choice sequencing request is processed.	True
Sequencing Control Flow	Indicates the Flow Sub process may be applied to the children of the activity.	False
Sequencing Control Forward Only	Indicates that backward targets (in terms of activity tree traversal) are not permitted for the children of the activity.	False
Use Current Attempt Objective Information	Indicates that the Objective Progress Information for the children of the activity will only be used in rule evaluations and rollup if that information was recorded during the current attempt on the activity.	True
Use Current Attempt Progress Information	Indicates that the Attempt Progress Information for the children of the activity will only be used in rule evaluations and rollup if that information was recorded during	True

	the current attempt on the activity.	
--	--------------------------------------	--

Figure 9.2.2a

Note: In the Control Mode section, any default values used in code appear in bold print.

### 9.2.2.1 Choice

Choice indicates that the learner can choose an activity from a visible table of contents. The activity tree should be visible as a table of contents and two LMS UI buttons of 'Suspend' and 'Quit' are shown. Choice only applies to clusters and will have no effect on leaves.

```
<imsss:sequencing>
  <imsss:controlMode choice="true" choiceExit="true" flow="true"
forwardOnly="false"/>>
</imsss:sequencing>
```

Figure 9.2.2.1a

Choice Control Mode example on a cluster:

```
<organization identifier="Org100"
adlseq:objectivesGlobalToSystem="false">
...
<item identifier="C100">                                (Cluster)
  <title>Start Course</title>
  <item identifier="S1001" identifierref="R_S1001">      (SCO)
    <title>Welcome to 88H10 Cargo Specialist</title>
  </item>
  <item identifier="S1002" identifierref="R_S1002">      (SCO)
    <title>Pretest for Module 1 - Cargo
Specialist</title>
  </item>
  ...
  <imsss:sequencing>
    <imsss:controlMode choice="true"/>>                (Code indicates
  </imsss:sequencing>                                  learner can choose
</item>                                                  any activity )
...
</organization>
```

Figure 9.2.2.1b

### 9.2.2.2 Flow

Flow indicates that the learner can navigate to the previous activity or the next activity. The LMS must provide four LMS UI buttons to execute the navigation events 'Continue', 'Previous', 'Suspend', and 'Quit'. Flow only applies to clusters and will have no effect on leafs.

```
<imsss:sequencing>
  <imsss:controlMode choice="false" choiceExit="true" flow="true"
forwardOnly="false"/>>
</imsss:sequencing>
```

Figure 9.2.2.2a

Following is a Flow XML example for one cluster within the manifest:

```
<organization identifier="Org100"
adlseq:objectivesGlobalToSystem="false">
...
<item identifier="C100">                                (Cluster)
  <title>Start Course</title>
  <item identifier="S1001" identifierref="R_S1001">      (SCO)
    <title>Welcome to 88H10 Cargo Specialist</title>
  </item>
  <item identifier="S1002" identifierref="R_S1002">      (SCO)
    <title>Pretest for 88H10 Cargo Specialist</title>
  </item>
  <imsss:sequencing>
    <imsss:controlMode choice="false" flow="true"/> → (Code indicates
  </imsss:sequencing>                                learner can go back
</item>                                                one activity and
...                                                    forward one
</organization>                                       activity)
```

Figure 9.2.2.2b

### 9.2.2.3 Forward Only

Forward Only indicates that the learner can only navigate to the next activity, and may not go to the previous activity. The LMS will provide the three LMS UI buttons to execute the navigation events of 'Continue', 'Suspend', and 'Quit'. ForwardOnly applies to clusters only and will have no effect on leafs.

```
<imsss:sequencing>
  <imsss:controlMode choice="false" choiceExit="true" flow="true"
forwardOnly="true"/>>
</imsss:sequencing>
```

Figure 9.2.2.3a

Following is a Forward Only example for one cluster within the manifest:

```

<item identifier="C100">                                     (Cluster)
  <title>Instructional Content Cluster</title>
  <item identifier="S1001" identifierref="R_S1001">
    <title>SCO 1</title>
  </item>
  <item identifier="S1002" identifierref="R_S1002">
    <title>SCO 2</title>
  </item>
  <imsss:sequencing>                                       (Sequencing code on Cluster)
    <imsss:controlMode choice="false" flow="true" forwardOnly="true"/>
  </imsss:sequencing>
</item>

```

Figure 9.2.2.3b

The following will enable a Forward-Only control mode that, upon launch, automatically flows into first target activity:

```

<imsss:controlMode choice="false" choiceExit="true" flow="true"
forwardOnly="true"/>

```

Figure 9.2.2.3c

#### 9.2.2.4 Choice Exit

Choice Exit indicates that the learner cannot navigate away from the current activity to non-descendent activities.

```

<imsss:sequencing>
  <imsss:controlMode choice="true" choiceExit="false" flow="false"
forwardOnly="false"/>>
</imsss:sequencing>

```

Figure 9.2.2.4a

#### 9.2.2.5 Constrained Choice Considerations

Constrained Choice Considerations only applies to a cluster and has no effect on a leaf. Following is a description of Constrain Choice Controls:

Name	Description	Default Value
Constrain Choice	Indicates that a Choice sequencing request should only allow activities that are logically next in a 'flow' from the activity to be identified for delivery.	False
Prevent Activation	Indicates that a Choice sequencing request should only allow descendents of the activity to be identified for delivery, it the	False

	activity is already active.	
--	-----------------------------	--

Figure 9.2.2.5a

Following is an example of how choice navigation requests should be constrained during the sequencing process:

```
<imsss:sequencing>
  <imsss:controlMode choice="true" flow="true"/>
  ...
  <adlseq:constrainedChoiceConsiderations constrainChoice="true"/>
  ...
</imsss:sequencing>
```

Figure 9.2.2.5b

Following is an example of how to prevent activation. This indicates that attempts on child activities should not begin unless the current activity is the parent:

```
<imsss:sequencing>
  <imsss:controlMode choice="true" flow="true"/>
  ...
  <adlseq:constrainedChoiceConsiderations preventActivation="true"/>
  ...
</imsss:sequencing>
```

Figure 9.2.2.5c

## 9.2.3 Sequencing Impact on Graded Assessments

### 9.2.3.1 Scaled Passing Score Example

The scaled passing score is indicated in the following example within the <imsss:minNormalizedMeasure> tag as 80%. The value is normalized between -1 and 1, inclusive.

```
<imsss:objectives>
  <imsss:primaryObjective objectiveID="SCO_MS" satisfiedByMeasure = "true">
    <imsss:minNormalizedMeasure>0.80</imsss:minNormalizedMeasure>
  </imsss:primaryObjective>
</imsss:objectives>
```

Figure 9.2.3.1a

For additional information, refer to the [Learner Scoring and Success Status](#) section (9.10).

### 9.2.3.2 Disable Graded Assessment When Completed Example

```
<imsss:sequencingRules>
  <imsss:preConditionRule>
    <imsss:ruleConditions>
      <imsss:ruleCondition condition="completed"/>
    </imsss:ruleConditions>
    <imsss:ruleAction action="disabled"/>
  </imsss:preConditionRule>
</imsss:sequencingRules>
```

Figure 9.2.3.2a

Note: The test will disable whether passed or failed, indicating a one attempt limit.

### 9.2.4 Rule-Based Sequencing

Rule-based sequencing can be likened to 'conditional rules' because the rules follow a pattern of "if this condition is true, then take this action".

Basic syntax example for 'If [Condition] is [true], then [Action]':

```
<imsss:ruleConditions>
  <imsss:ruleCondition operator="[not]" condition="[CONDITION]"/>
</imsss:ruleConditions>
<imsss:ruleAction action="[ACTION]"/>
```

Figure 9.2.4a

Code example for 'If [Condition] is [not true], then [Action]':

```
<imsss:ruleConditions>
  <imsss:ruleCondition operator="not" condition="satisfied"/>
</imsss:ruleConditions>
<imsss:ruleAction action="exitParent"/>
```

Figure 9.2.4b

Code example for 'If any [Conditions] are [true], then [Action]':

```
<imsss:ruleConditions conditionCombination="any">
  <imsss:ruleCondition condition="satisfied"/>
  <imsss:ruleCondition condition="activityProgressKnown"/>
</imsss:ruleConditions>
<imsss:ruleAction action="skip"/>
```

Figure 9.2.4c

Code example for 'If [Satisfied] is [true], then [Hidden from Choice] ':

```
<imsss:sequencingRules>
  <imsss:preConditionRule>
    <imsss:ruleConditions>
      <imsss:ruleCondition condition="satisfied"/>
    </imsss:ruleConditions>
    <imsss:ruleAction action="hiddenFromChoice"/>
  </imsss:preConditionRule>
</imsss:sequencingRules>
```

Figure 9.2.4d

Skip to next target activity if a Mastery Pretest is passed:

```
<imsss:sequencingRules>
  <imsss:postConditionRule>
    <imsss:ruleConditions>
      <imsss:ruleCondition condition="satisfied" />
    </imsss:ruleConditions>
    <imsss:ruleAction action="exitParent" />
  </imsss:postConditionRule>
</imsss:sequencingRules>
```

Figure 9.2.4e

## 9.2.5 Rollup Rules Examples

Rollup rules define how learner progress for cluster activities is to be evaluated. Rollup rules apply to a set of child activities and have no effect when defined on a leaf activity.

Following is a rollup example of "if all SCOs are attempted, then rollup a completion status":

```
<imsss:rollupRules>
  <imsss:rollupRule childActivitySet="all">
    <imsss:rollupConditions>
      <imsss:rollupCondition condition="attempted"/>
    </imsss:rollupConditions>
    <imsss:rollupAction action="completed"/>
  </imsss:rollupRule>
</imsss:rollupRules>
```

Figure 9.2.5a

If a cluster contains many activities and only a portion of the activities are required, then a minimum count can be designated. Following is a rollup example of a minimum count condition applied to the child activities:

```
<imsss:rollupRules>
  <imsss:rollupRule childActivitySet="atLeastCount" minimumCount="3">
    <imsss:rollupConditions>
      <imsss:rollupCondition condition="attempted"/>
    </imsss:rollupConditions>
    <imsss:rollupAction action="completed"/>
  </imsss:rollupRule>
</imsss:rollupRules>
```

Figure 9.2.5b

The portion of required activities can also be set as a percentage. Following is a rollup example with a minimum percent condition applied to the child activities:

```
<imsss:rollupRules>
  <imsss:rollupRule childActivitySet="atLeastPercent" minimumPercent=".50">
    <imsss:rollupConditions>
      <imsss:rollupCondition condition="attempted"/>
    </imsss:rollupConditions>
    <imsss:rollupAction action="completed"/>
  </imsss:rollupRule>
</imsss:rollupRules>
```

Figure 9.2.5c

Following is a rollup example where multiple rollup conditions are applied to all child activities:

```
<imsss:rollupRules>
  <imsss:rollupRule childActivitySet="all">
    <imsss:rollupConditions conditionCombination="all">
      <imsss:rollupCondition condition="attempted"/>
      <imsss:rollupCondition condition="objectiveMeasureKnown"/>
    </imsss:rollupConditions>
    <imsss:rollupAction action="completed"/>
  </imsss:rollupRule>
</imsss:rollupRules>
```

Figure 9.2.5d

Note: Default value for condition combination is 'any'.

### 9.2.5.1 Rollup Considerations Example

Following is an example of a rollup consideration:

```
<imsss:sequencing>
  ...
  <imsss:sequencingRules>
    ...
    <adlseq:rollupConsiderations requiredForSatisfied="ifNotSkipped"
requiredForCompleted="ifNotSkipped"/>
  </imsss:sequencingRules>
  ...
</imsss:sequencing>
```

Figure 9.2.5.1a

### 9.2.6 Randomization Example

Randomization Controls is the container for how child activities should be ordered or selected during the sequencing process.

Following is an example of randomization controls that select a portion of the activities:

```
<item identifier="posttest">
  <title>Posttest</title>
  <item identifier="Q1" identifierref="R_Q1">
    <title> Posttest Question </title>
  </item>
  <item identifier="Q2" identifierref="R_Q2">
    <title> Posttest Question </title>
  </item>
  <item identifier="Q3" identifierref="R_Q3">
    <title> Posttest Question </title>
  </item>
  <item identifier="Q4" identifierref="R_Q4">
    <title> Posttest Question </title>
  </item>
  <item identifier="Q5" identifierref="R_Q5">
    <title> Posttest Question </title>
  </item>
  <imsss:sequencing>
    <imsss:randomizationControls selectCount="3" reorderChildren="true"
selectionTiming="onEachNewAttempt" />
  </imsss:sequencing>
</item>
```

Figure 9.2.6a

## 9.2.7 Delivery Controls Examples

Delivery controls describe what data the LMS can expect to receive from the activity. It aids in the management of the activity's tracking status information. There are three designators: tracked, completion set by content, and objective set by content.

### 9.2.7.1 Tracked

The LMS will track all activities by default unless tracking is specifically turned off. An activity designated as 'not tracked' is not included in any rollup evaluations of the parent and behaves as if it does not initialize and manage or access any tracking status information. It restricts the set of sequencing strategies that may be applied to the activity.

Example indicating that the tracking of activities has been turned off:

```
<item>
  <title>Activity Title</title>
  <imsss:sequencing>
    <imsss:deliveryControls tracked="false"/>
  </imsss:sequencing>
</item>
```

Figure 9.2.7.1a

### 9.2.7.2 Completion Set by Content

The LMS will set the activity's Attempt Completion Status by default unless specifically turned off. The Completion Set by Content must be defined for a leaf activity and has no effect when defined on a cluster.

Example of indicating that the attempt completion status will be set by content:

```
<item>
  <title>Activity Title</title>
  <imsss:sequencing>
    <deliveryControls completionSetByContent="true"/>
  </imsss:sequencing>
</item>
```

Figure 9.2.7.2a

### 9.2.7.3 Objective Set By Content

The LMS will set the activity's Objective Satisfied Status by default unless specifically turned off. The Objective Set By Content must be defined for a leaf activity and has no effect when defined on a cluster.

Example of indicating that the objective satisfied status will be set by content:

```
<item>
  <title>Activity Title</title>
  <imsss:sequencing>
    <imsss:deliveryControls objectiveSetByContent="true"/>
  </imsss:sequencing>
</item>
```

Figure 9.2.7.3a

### 9.2.8 Hiding the LMS User Interface (UI) Buttons Example

The <hideLMSUI> element contains a restricted vocabulary of four values: 'previous', 'continue', 'exit', and 'abandon'. If the values are not listed, the LMS provides a UI device for those navigation requests by default. Following is an example of hiding two UI buttons for one SCO within the manifest:

```
<organization identifier="Org100"
adlseq:objectivesGlobalToSystem="false">                                     (SCO)
...
<item identifier="S100" identifierref="R_S100">
  <title>SCO 1</title>
  <adlnav:presentation>
    <adlnav:navigationInterface>
      <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
      <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
    </adlnav:navigationInterface>
  </adlnav:presentation>
  (Code to hide LMS
  Navigation buttons
  because navigation
  buttons are already
  inside the SCO*)
</item>
...
</organization>
```

Figure 9.2.8a



**Note:** SCORM 2004 3<sup>rd</sup> Edition has updated the restricted vocabulary to include the values of 'suspendAll', 'exitAll', and 'abandonAll'.

### 9.3 imsss:MapInfo – Shared Data Example

SCOs can indirectly access data from other SCOs through Objective Maps, applied externally in the manifest, using the Sequencing Model. Objective Maps represent the link between the SCO and the shared data needed by the sequencing process. Local objectives of a SCO can be mapped or linked to a defined shared global objective using Objective Maps.



**ADL Reference:** A SCO cannot directly access the values of other SCOs but can access the values of shared global objectives. By default, an activity will only be able to access Objective Progress Information for the set of learning objectives defined for the activity – these are called

'local' objectives. Other activities cannot directly reference the Objective Progress Information associated with another activity's objectives; however, an Objective Map may be defined to relate a local objective to a globally shared objective.

Following is an example of a pretest SCO with an Objective Map, which is designated by `<imsss:mapInfo>`. The MapInfo tag writes values to the shared data area (shared global objective):

```
<item identifier="Pretest" identifierref="R_Pretest">
  <title>Pretest 1</title>
  <imsss:sequencing>
    <imsss:objectives>
      <imsss:primaryObjective objectiveID="activity_obj"
satisfiedByMeasure = "true">
        <imsss:minNormalizedMeasure>0.70</imsss:minNormalizedMeasure>
        <imsss:mapInfo targetObjectiveID="ARMY_88H10_071-312-3028"
readSatisfiedStatus="false" readNormalizedMeasure = "false"
writeSatisfiedStatus="true" readNormalizedMeasure = "true"/>
      </imsss:primaryObjective>
    </imsss:objectives>
  </imsss:sequencing>
</item>
```

Figure 9.3a

## 9.4 Sequencing Collection Example

The sequencing collection allows the programmer to reuse sequencing code. The `<imsss:sequencingCollection>` tag acts as a container for the set of sequencing information.

Following is an example of the sequencing collection tag:

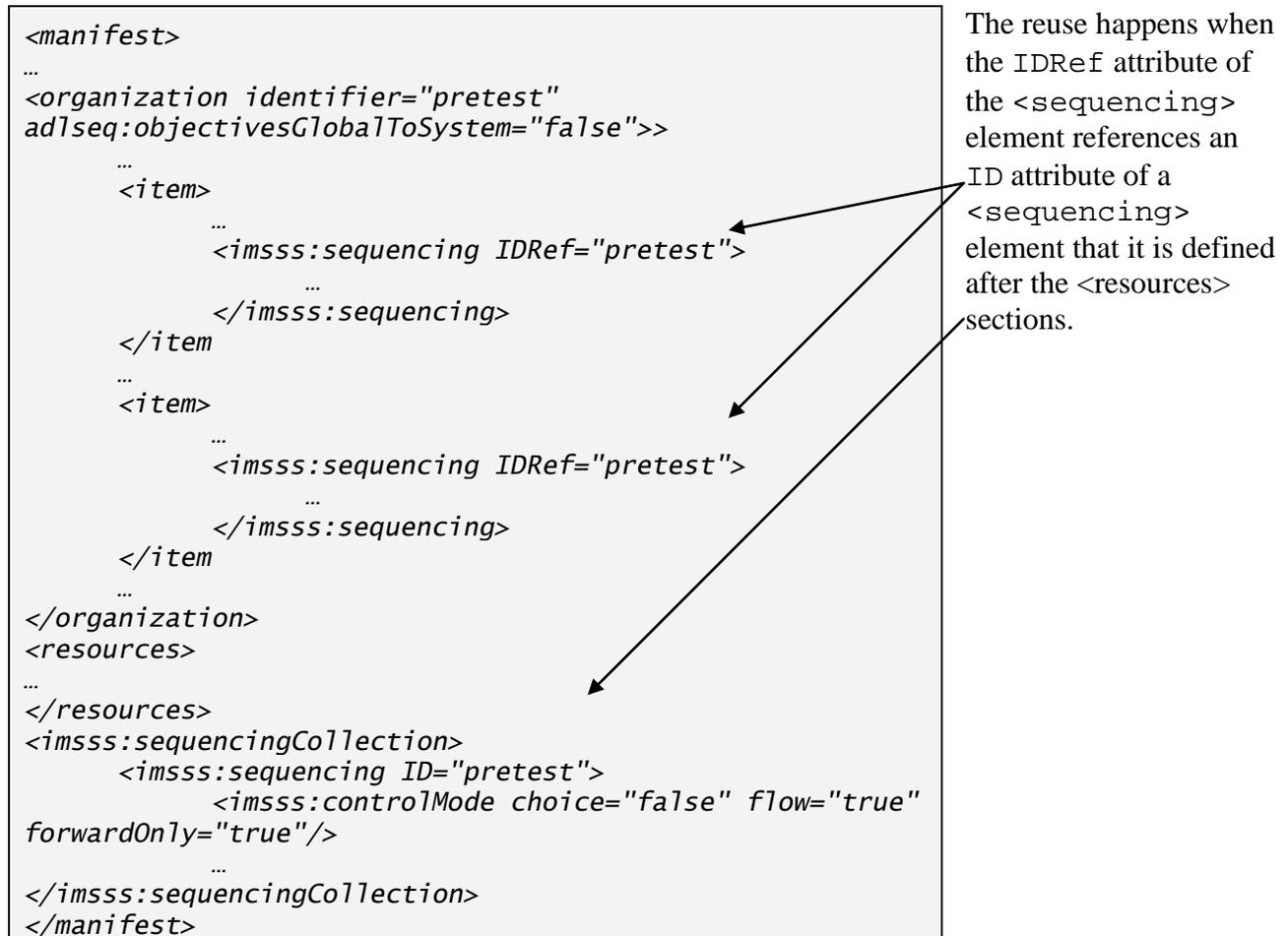


Figure 9.4a

## 9.5 Communication with the LMS (Run-Time Environment) Examples



**ADL Reference:** Every SCO is required, per SCORM specifications, to adhere to the SCORM Run-Time Environment. 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("")'. The SCO locates the API Instance according to the valid locations that an LMS is permitted to place its API Instance. All communication between the LMS and the SCO is initiated by the SCO.

Note: All of the examples in this section use the JavaScript scripting language and the functionality is obtained by using JavaScript through the Run-Time Environment or the browser.

Differences between SCORM v1.2 and SCORM 2004 API function calls:

SCORM v1.2	SCORM 2004
<i>LMSInitialize("");</i>	<i>Initialize("")</i>
<i>LMSGetValue(data model, value);</i>	<i>GetValue(data model, value);</i>
<i>LMSSetValue(data model, value);</i>	<i>SetValue(data model, value);</i>
<i>LMSCommit("");</i>	<i>Commit("");</i>
<i>LMSFinish("");</i>	<i>Terminate("");</i>

Figure 9.5a

The following API General Application Rules shall be followed in order to achieve interoperability:

- The function names are all case-sensitive, and must always be expressed exactly as shown and described in the SCORM specification.
- The function parameters or arguments are case-sensitive. All SCORM supported data model (SCORM Run-Time Environment Data Model and SCORM Navigation Data Model) parameters shall be represented in lower case.
- All parameters passed between a SCO and the API Instance are treated as ECMAScript strings and shall be compatible with the data types and formats described by the data models that use the API for communication.



**Best Practice:** A JavaScript API Wrapper file is provided by ADL as an example of the SCORM communication with the LMS. This file incorporates state management diagnostic functions to handle errors. Users may obtain a new API wrapper file (as of October 2011) ADL's Web site.

All of the programming code examples assume use of the API Wrapper file developed by ADL. Programming subroutines may begin with a similar name but have different parameters. It is the developer's responsibility to make sure to use the API Wrapper function calls if using the API Wrapper file or, if not, use the exact SCORM API function calls as designated in the SCORM specification. If not using the API Wrapper file, the developer would replace each function name listed in the following figure with the actual SCORM specification API function call:

Using actual SCORM API function calls without the API Wrapper file	With the API Wrapper file used in Army examples
<i>Initialize("")</i>	<i>doInitialize()</i>
<i>GetValue("")</i>	<i>doGetValue()</i>
<i>SetValue("")</i>	<i>doSetValue()</i>
<i>Commit("")</i>	<i>doCommit()</i>
<i>Terminate("")</i>	<i>doTerminate()</i>

Figure 9.5b

For example, the API Wrapper function call is 'doInitialize()' within this document, and the SCORM specification API function call is 'Initialize(')." There could be different versions of a Wrapper file with different function names.

### 9.5.1 Begin Communication



**ADL Reference:** 'Initialize' is the mandatory SCORM API function call for all SCOs, which indicates that the SCO is ready to begin communication with the LMS.

```
doInitialize();
```

Figure 9.5.1a

Note: All examples use the API Wrapper file function names.



**Best Practice:** Initialize should not be called from the onLoad event on the first page of the SCO. Errors will result when clicking the 'Back' button on the second page.

To avoid this error, a 'SCO Start' (redirect) page has been suggested that separately calls 'Initialize'. This page is placed as the launch page of the SCO. It will call 'Initialize' and then immediately will redirect the learner to the first 'real' page of the SCO. All that is required is changing the href of the anchor tag on the 'SCO Start' page to the redirected real first page of SCO. If this method is used, 'Initialize' should only be called on the 'SCO Start' page.

Sample code for the 'SCO Start' page ('scostart.html') is as follows:

```
<html>
<head>
<script type="text/javascript" src="APIWrapper.js"></script>
</head>
<body>
Initializing Lesson...<br>
Please wait
<script type="text/javascript">
    doInitialize();
    <!-- (This href would be real first page) -->
    document.location.href="unit/index.html";
</script>
</body>
</head>
```

Figure 9.5.1b

## 9.5.2 End Communication



**9.5.2-1 Business Rule (Army):** Whenever the learner leaves a SCO, (for example a SCO navigation request, pushing an 'Exit' button on the SCO page, etc.) the SCO must terminate communication with the LMS.

Note: Going to an external reference is not the same as leaving a SCO. For more information, see [External References](#) section (3.12) or [External References Example](#) section (9.18).



**ADL Reference:** 'Terminate' is mandatory for all SCOs. 'Terminate' indicates that the SCO no longer needs to communicate with the LMS. Upon receiving a 'Terminate' request from a SCO, the LMS shall set the cmi.exit to "" (empty characterstring) or resume and exit the user from the package.

```
doTerminate();
```

Figure 9.5.2a

In SCORM 2004, a SCO may be removed from a learner by either a SCO's internal navigation buttons, a window event (i.e. closing the browser window), or through a loss of connection with the LMS. In order to ensure that 'Terminate' is called in a majority of these cases, SCO developers should design their SCOs to 'Terminate' on a page event (that is, onBeforeUnload). In order to avoid the error that will occur if the SCO calls 'Terminate' more than once, SCOs should be designed to track their current status. This can be achieved using a frameset approach to building SCOs.

### Frameset Solution:

Following is the solution for a frameset, which places the exit code on the parent frame instead of the frame containing the learning content. This means that only when the page is unloaded will the events be executed. When navigating through the SCO's pages, the SCO's 'isTerminated' variable keeps track of the SCO's current status. Any time the SCO is removed, either by the learner, the LMS, or a browser window closing event, the SCO will attempt to terminate the session.

Sometimes it is necessary to call 'Terminate' from within the SCO in order to initiate a navigation request. This is often the case when SCOs implement their own navigation interface. In these cases, the SCO must call 'Terminate' first, and then the LMS will unload the SCO and execute the desired navigation event. If there is no way to detect the current status of a SCO, 'Terminate' will be called twice. This could have unanticipated results. It is therefore very important for a SCO to be able to track its termination status. If the SCO has already been terminated, then 'Terminate' should not be called again.

Using the above frameset scenario, an example of a frameset page ('frame.html') follows:

```

<html lang="en-US">
<head>
  <title>GUI Template</title>
  <script type="text/javascript" src="SCO_bestprac.js"> </script>
  <noscript>Sorry, your browser does not support JavaScript. For
    this training to run correctly JavaScript must be
    enabled.
  </noscript>
  <script type="text/javascript" src="APIWrapper.js"> </script>
  <noscript>Sorry, your browser does not support JavaScript. For
    this training to run correctly JavaScript must be
    enabled.
  </noscript>
  <script type="text/javascript">

    var isTerminated = false;

  </script>
</head>
<frameset rows="*,0px" border="0" onbeforeunload="endSCO('suspend')" >
  <frame name="content" src="scostart.html" scrolling="no"
    frameborder="0" title="Content frame, used to display
    training content." />
  <frame name="storage" src="storage.html" noresize="noresize"
    frameborder="0" title="Storage frame, used to store training
    related information." />
<noframes>
  Sorry, your browser does not support frames. For this
  training to run correctly Frames must be enabled.
</noframes>
</frameset>
</html>

```

Figure 9.5.2b

Notice that a variable 'isTerminated' is initialized on the parent frame. This will be used to track the termination status for the SCO.

The courseware content page ('scostart.html') is shown in the following example:

```
<html lang="en-US">
<head>
  <title>SCO Start</title>
  <script type="text/javascript" src="APIWrapper.js"> </script>
  <noscript>Sorry, your browser does not support JavaScript. For
    this training to run correctly JavaScript must be
    enabled.
  </noscript>
  <script type="text/javascript" src="SCO_bestprac.js"></script>
  <noscript>Sorry, your browser does not support JavaScript. For
    this training to run correctly JavaScript must be
    enabled.
  </noscript>
</head>

<body leftmargin="0" topmargin="0" rightmargin="0" bottommargin="0"
  marginwidth="0" onLoad=" beginSCO( 'index.html' ); ">

  <p align="center">
    
  </p>
  <h1 align="center">Initializing Training</h1>
  <p align="center">Loading...</p>

</body>
</html>
```

Figure 9.5.2c

Notice that this content page will call the JavaScript function 'beginSCO()'. Excerpts from this function are seen in the example that follows. It is important to also note that 'beginSCO()' takes as a parameter a relative file location. This parameter will be loaded into the parent frameset as the content page. This is insignificant for this example, but does provide another example of developing SCOs in a modular fashion.

Excerpts from the JavaScript file ('SCO\_bestprac.js') that defines how this SCO will interface with the APIWrapper is shown in the following example:

```
// =====  
// beginSCO() will start up a new sco. Generally called in the onLoad of  
// body tag of the first page of a SCO  
function beginSCO( defaultStartPage ) {  
  
    doInitialize();  
    isTerminated = false;  
  
    // More business logic that gets executed on initialization of SCO  
    // ...  
  
    // Move the page!!  
    if ( defaultStartPage ) {  
        document.location = defaultStartPage;  
    }  
}  
  
// =====  
// End the SCO. Generally gets called when the user leaves  
// the sco in some fashion.  
function endSCO( exitStatus ) {  
  
    // Check to see if the SCO is already Terminated  
    if( isTerminated ){ return;}  
  
    // More business logic that gets executed when a SCO ends,  
    // such as setting session time, exit status, etc...  
  
    parent.isTerminated = true; //SET THIS TO ENSURE ONLY 1 TERMINATE  
    //isTerminated = parent.isTerminated;  
  
    doTerminate();  
}
```

Figure 9.5.2d

Notice that the 'beginSCO()' function will initialize the 'isTerminated' value to "false". The 'endSCO()' function will first check this variable to ensure that the SCO hasn't already been terminated. It will then continue to wrap up all necessary communications with the LMS. Once the 'endSCO()' function has completed all necessary communications with the LMS, it sets the 'isTerminated' variable to "true" in the parent frame before it calls the 'doTerminate()' function from the 'APIWrapper.js' file. Now, if the 'endSCO()' function is called again, that is, the page unloads, the 'isTerminated' variable will equate to 'true', and the 'endSCO()' function will close without setting 'Terminate' twice.



**Best Practice: Use the Document Object Model (DOM) reference of 'parent' within your content (instead of 'top') to avoid interfering with LMS frames.**

## 9.6 Data Transfer

Data Transfer functions are used to transfer data to and from a LMS using the SCORM data model. The data model is a standard set of data elements used to define the information being communicated, such as the status of the learning resource. The data model is elements explicitly described in the SCORM Run-Time Environment (RTE) specifications document.

### 9.6.1 Retrieving Data

The SCORM Run-Time Environment specification (RTE) designates which data model supports the 'retrieve' capability.

```
variable = doGetValue("[data model element]");
```

Figure 9.6.1a

Example:

```
var status = doGetValue("cmi.success_status");
```

Figure 9.6.1b

### 9.6.2 Storing Data

The SCORM RTE designates which data model supports the 'write' capability.

The 'SetValue' API function call allows the SCO to store or 'set' information in the LMS.

```
doSetValue("[data model element]", "[value]");
```

Figure 9.6.2a

The value of "incomplete" is stored in the SCO's completion\_status data field in the following example:

```
doSetValue("cmi.completion_status", "incomplete");
```

Figure 9.6.2b

### 9.6.3 Commit

The 'Commit' command releases stored values to be written to the LMS database.

The 'Commit' API function call requires that any cached values, previously set via SCO, calls to 'SetValue' that have not been persisted by the LMS, be persisted.

```
doCommit( );
```

Figure 9.6.3a



**Best Practice: Use the 'Commit' command for data that needs to be persisted in the LMS to protect against loss of connectivity.**

'Commit' should be called, at a minimum, whenever there is a 'SetValue' of critical data elements such as "cmi.success\_status", "cmi.completion\_status", and "cmi.score.scaled".

**PROGRAMMER INFO:** When using an error handler to resolve an issue with executing 'commit' commands, the error handler must correct the error. If not properly handled the commit could reexecute the error handler again which will result in an error handler loop. An error handler must correct the error condition or provide an exit route clearing the error condition.

## 9.7 State Management

State Management defines a standardized way to determine an error condition with the API communication mechanism. The API has three API function calls that are used to handle errors.

### 9.7.1 GetLastError

'GetLastError' allows the SCO to determine an error condition on the communication to and from the LMS. GetLastError is automatically included in programming code when using an API Wrapper file provided by ADL.

This API function call provides a way of assessing whether or not any given API function call was successful. It returns an error status code resulting from the previous API function call.

```
var retVal = doGetLastError();
```

Figure 9.7.1a

Error codes are listed in the 'Content Aggregation Model (CAM)' specifications document, which is a part of the SCORM Documentation Suite.

## 9.8 Macromedia Flash Specific Run-Time Example

Communication with the LMS can be achieved with Macromedia Flash by using JavaScript.

Following is a JavaScript 'setScore' function. The Flash code (shown in the following figure) would be similar to the following JavaScript function in a <script> tag on the HTML page:

```
function setScore(score) {  
    doSetValue("cmi.score.raw", score);  
    doCommit();  
}
```

Figure 9.8a

Within Flash 5 or Flash MX, the Actionscript would be similar to the following:

```
var score=0;  
score = 75;    (whatever the learner scored on the performance test)  
getURL("javascript:setScore(' "+ score + " ' )" );
```

Figure 9.8b

The learner's score from within Flash is sent via the Flash 'getURL' function to the JavaScript 'setScore' function.

## 9.9 Bookmarking Example

The data model element most commonly used for bookmarking is "cmi.location". This value can be stored when the learner suspends the SCO and then retrieved when the learner re-accesses the SCO.

Following are examples of bookmarking code:

Retrieving:

```
// retrieving the bookmark from the LMS  
var retVal = doGetValue("cmi.location");
```

Figure 9.9a

Storing (pseudo code):

```
// storing the bookmark in the LMS  
doSetValue("cmi.location", "[page the learner is viewing]");
```

Figure 9.9b

Storing:

```
doSetValue("cmi.location", window.location);
```

Figure 9.9c



**Best Practice:** A bookmark, when required, should be sent on every page to help maintain communication with the LMS. For simulations, bookmarks should be sent after each simulation event. A loss of connectivity may occur in courseware when the LMS is not

receiving data or is timing out. The increased frequency of bookmarking has been recommended as a solution to this problem.



**Best Practice: SCOs should prompt the user to determine whether or not the user would like to return to a bookmarked location within a SCO, unless the bookmark is set to the first page.**

At the entry of a SCO, provide the learner a choice by prompting the learner to either return to the last location or go to the beginning. After 'Initialize', the SCO should determine whether or not the SCO has previously been attempted. It is recommended that the SCO retrieve the value of "cmi.entry" to determine whether or not a SCO has already been attempted. If the value of "cmi.entry" is "ab-initio", then the user has not entered the SCO, and no bookmark will exist. If the value of "cmi.entry" is "resume" or "", then the learner has previously entered the SCO, and there may be a bookmark. Next, read "cmi.location" and if a value exists, the SCO should prompt the learner to either return to where the learner left off or go to the beginning.

Following is an example of code to prompt the learner after retrieving the bookmark:

```
var entry = doGetValue("cmi.entry");

If(entry != "ab-initio"){
  var lastLocation = doGetValue("cmi.location");
  if (lastLocation != "") {
    result = confirm('Would you like to return to your last
location?');

    if ( result ) {
      document.location.href=lastLocation;
    }
  }
}
```

Figure 9.9d



**Best Practice: An empty bookmark should be set on the last page of content SCOs.**

An empty bookmark is set on the last page of content SCOs, which would overwrite the previous value stored in the LMS. For example, a learner, after earlier completing all of the training, might want to review the material for a learner performance test and reassess the training. Launching the SCO would invoke the bookmark, which would take the learner to the last page and would be unnecessary and frustrating.

Following is an example of setting an empty bookmark on the last page of a Content SCO:

```
doSetValue("cmi.location", "");
```

Figure 9.9e

## 9.10 Learner Scoring and Success Status

A "mastery score" in SCORM 2004 is the value that determines the minimum score possible for a learner to successfully pass a graded assessment. The `<minNormalizedMeasure>` manifest element along with the run-time API data fields "cmi.score.scaled", "cmi.scaled\_passing\_score", and "cmi.success\_status" combine within the LMS to evaluate learner performance.

### 9.10.1 Mastery Score Example



**9.10.1-1 Business Rule (Army): Graded assessment SCOs shall ensure that an appropriate value for the success status, either "passed" or "failed" is set in the LMS. For graded assessment SCOs that pass a score to the LMS, a corresponding value for the `<minNormalizedMeasure>` element must be delineated on the manifest. These SCOs shall make evaluations of the success status based upon the `<minNormalizedMeasure>` element provided in the manifest file.**

SCORM allows the ‘**scaled passing score**’ to be designated outside of the SCO within the sequencing code associated with a SCO on the manifest file. This value can be compared with the ‘`scaled_passing_score`’ that learner achieved on the assessment to determine the learner's success or failure of the assessment. To ensure that a graded assessment SCO stores the appropriate status of a learner's success or failure, the developer should:

- Set the ‘`scaled_passing_score`’ using the `<minNormalizedMeasure>` element within the manifest
- Initialize the learner's success status
- Set the learner's score
- Retrieve the learner's success status

#### Setting SCO ‘**Scaled Passing Score**’ in the Manifest

To support greater reuse and customization of graded learner assessments, the Army requires that the ‘**Scaled Passing Score**’ be set within the XML manifest file. This allows developers to change the ‘**Scaled Passing Score**’ just by altering one value in the 'imsmanifest.xml' file, rather than reworking a SCO within the original authoring tool to edit the embedded value. The ‘**Scaled Passing Score**’ is defined in the manifest as the value of the `<imsss:minNormalizedMeasure>` element tag associated with the SCO. This value must be of the XML Data Type `xs:decimal`, and be between -1.0000 and 1.0000, inclusive, with a precision of up to four decimal places.

Following is a code example of a ‘**Scaled Passing Score**’ being set to 80%:

```

<imsss:objectives>
  <imsss:primaryObjective objectiveID="SCO_MS" satisfiedByMeasure = "true">
    <imsss:minNormalizedMeasure>0.80</imsss:minNormalizedMeasure>
  </imsss:primaryObjective>
</imsss:objectives>

```

Figure 9.10.1.1a

The value set for the **<imsss:minNormalizedMeasure>** within the manifest file will be used by the LMS to initialize the run-time data field, "cmi.scaled\_passing\_score". This data field may be retrieved by the SCO at any time during a learner's session.

Following is an example of retrieving scaled passing score from the LMS by the SCO:

```
variable = doGetValue("cmi.scaled_passing_score ");
```

Figure 9.10.1.1b

### 9.10.1.1 Initializing Learner's Success Status

The first time, in which a learner enters a graded assessment SCO, the value of the run-time data field, "cmi.success\_status", should be "unknown". The SCO developer should test this value to ensure that a correct value is set in case the learner is prematurely disconnected from the SCO. This step is redundant in that the LMS is responsible for setting the "cmi.success\_status" to "unknown" in cases where the "cmi.score.scaled" value is unknown; however, testing for this value will ensure that this SCO is as portable as possible.

```

var = doGetValue("cmi.success_status");
if(var == "passed" || var == "failed")
{
  [Insert business logic here for re-entry of assessment SCO here]
}
else if(var == "unknown")
{
  //Do nothing. Just checking to make sure we had a good value.
}
else
{
  doSetValue("cmi.success_status", "unknown");
}

```

Figure 9.10.1.2a

Depending on the learning strategy implemented by the courseware, a learner may or may not be allowed to exit and re-enter a graded assessment SCO. If a learner is allowed to re-enter a SCO, and the value for "cmi.success\_status" is already set to "passed", then that value shall not be changed. A value of "failed" may or may not be changed, depending on the learning strategy implemented by the courseware.

Setting Learner's Score

The learner's performance is evaluated within the SCO using whatever business logic the developer desires. The "cmi.score.scaled" data model element allows the developer the ability to read or write the raw score ("cmi.score.raw"), the minimum score ("cmi.score.min") or the maximum score ("cmi.score.max") for an assessment SCO. However, the only value that will affect sequencing or the learner's Success Status for a graded assessment SCO is the scaled score ("cmi.score.scaled"). This value must be of the XML Data Type xs:decimal, and be between -1.0000 and 1.0000, inclusive, with a precision of up to four decimal places.

Following is an example of how to set the value for "cmi.score.scaled":

```
// learner's scaled score
var LearnerScore = ".58";

// sending the value to the LMS
doSetValue("cmi.score.scaled", LearnerScore);
```

Figure 9.10.1.3a

If this value is set, then the Objective Measure Status for the primary objective will be set to "true" and the Objective Normalized Measure for the primary objective will be equal to the set value.

#### Retrieving Learner's Success Status

The "cmi.success\_status" data field is meant to indicate a learner's current status within a graded assessment SCO. Valid values are: "passed", "failed", and "unknown". This field is readable and writable from within a SCO; however, the final value of this data field is ultimately determined by the LMS. Whenever a SCO requests the value of the "cmi.success\_status" field, the LMS must determine the value for that field, and then return to the SCO the up-to-date value.

In order to determine the final success status, the LMS will consider the values set for "cmi.scaled\_passing\_score", "cmi.score.scaled", and "cmi.success\_status". If there is a value set for "cmi.scaled\_passing\_score", then the LMS will determine the value of "cmi.success\_status" by comparing the value of "cmi.scaled\_passing\_score" with "cmi.score.scaled". If the scaled score is greater than or equal to the scaled passing score, the learner's success status computes to "passed", otherwise it computes to "failed".

If "cmi.scaled\_passing\_score" has been set to a valid value, but there is no valid value for "cmi.score.scaled", then "cmi.success\_status" will be "unknown", regardless of whether or not the SCO explicitly sets a value for "cmi.success\_status" as described in the [Initializing Learner's Success Status](#) section (9.10.1.2).

If the learner is allowed to re-enter a test and is not denied re-entry when the learner has a success status of "passed", then the "passed" status must not be changed.

## 9.10.2 Confusing Success Status with Completion Status

Success status has become confused with the completion status. "Completed" does not mean that the learner passed the test. "Completed" means that the learner has gone through the entire SCO but does not indicate that the learner was successful or unsuccessful. A completion status of

"completed" as used in a graded assessment indicates that the learner did not exit out prematurely from the test but saw every test item and every page. The learner can complete the test but still be unsuccessful. A learner can obtain a completion status of "completed" and a success status of "failed".

## 9.11 Completion Status Example

Completion status for all SCOs should be stored in the LMS. Valid values are: "completed", "incomplete", "not\_attempted", and "unknown". Following are syntax examples of how 'getting' (reading) and 'setting' (writing) the completion status values are coded:

```
variable = doGetValue("cmi.completion_status");  
doSetValue("cmi.completion_status", "[vocabulary]");
```

Figure 9.11a

Following are examples of completion status read from the LMS and written to the LMS:

```
//getting or reading the value from the LMS  
var status = doGetValue("cmi.completion_status");  
  
//setting or writing the value to the LMS  
doSetValue("cmi.completion_status", "incomplete");
```

Figure 9.11b

There is confusion regarding the completion status and success status, because some LMSs use completion status as an indicator of success.



**9.11-1 Business Rule (Army): Once a learner receives credit for a SCO or course, the courseware shall not change that status for any reason.**



**Best Practice: Completion status should not be arbitrarily changed without checking existing value. Conditional programming statements should be used. The concern is that a completion status would change when the learner returns to the lesson for remediation or later for sustainment.**

Following is an example of the programming logic for a Content SCO (no learner Performance test):

Condition	Completion Status Value sent to LMS
SCO launched	Get Value of "unknown" Set Value to "incomplete"
SCO exited prematurely	Do nothing (already set to "incomplete")
SCO exited on last page	Set Value to "completed"

Figure 9.11c

The JavaScript code for getting and setting completion status in a conditional statement is as follows:

```
var status = doGetValue("cmi.completion_status");

if (status != "completed") {
    doSetValue( "cmi.completion_status", "incomplete" );
}
```

Figure 9.11d

Notice that only under the condition of completion status not equal to "completed" will the status be changed to "incomplete". This allows learners, after completing the SCO, to re-enter the SCO to review the content, perhaps, before taking a learner performance test. The learner's "completed" status would not be changed.

## 9.12 Exit Status Example

The Army recommends an empty string ("") to indicate a normal exit state. Using "log-out" is not recommended. The Army only allows a SCO to close itself.

```
doSetValue( "cmi.exit", "" );
```

Figure 9.12a

## 9.13 Session Time Example

This value is the amount of time in hours, minutes, and seconds that the learner has spent in the SCO at the time they leave it. That is, this represents the time from beginning of the session to the end of a single use of the SCO.

The JavaScript date object is the vehicle that allows the manipulation of the date object based on a millisecond value, which is then converted back to the form desired. The format is PTnHnMnS where n is the numeric value. If a value is zero, then the notation for that unit of time is deleted from the format. For example, if there are no minutes, then the "nM" is deleted from the formatted value. The time format is completely presented in the SCORM Content Aggregation Model (CAM) specifications document.

Example logic:

```
//format example for sessionTime - "PT23M56S"  
// example shows 23 minutes, 56 seconds  
  
1. Get time the SCO was launched (start time)  
2. Get the time the learner is leaving the SCO (end time)  
3. Subtract the start time from the end time which indicates the  
   session time.  
4. Convert to correct timeinterval data type format  
5. Send session time to the LMS
```

Figure 9.13a

A timer function is needed to track the time the learner spends in the current session of the SCO and send to the LMS upon leaving the SCO. The timer would begin immediately after calling Initialize and stop right before setting the value in the LMS and then Terminate. Because of the stateless nature of HTML, using frames would be one solution to store the time value that the SCO was launched.

Following are JavaScript functions that will start the timer for the beginning value, obtain the ending value and find the difference, and then convert the seconds to hours, minutes, and seconds, and write the formatted value to "cmi.session\_time":

```
/* set a global variable on the parent frame */  
var startTime;  
  
/* This function stores the time the SCO was launched in a global  
variable 'startTime'. Call this function after doInitialize */  
function startTimer()  
{  
    startTime = new Date().getTime();  
}  
  
// This function calculates the time learner was in the activity. Call  
this function right before Terminate  
function setSessionTime() {  
  
    var currentTime = new Date();  
    var endTime = currentTime.getTime()  
  
    var calculatedTime = endTime-startTime;  
    var totalHours = Math.floor(calculatedTime/1000/60/60);  
  
    calculatedTime = calculatedTime - totalHours*1000*60*60  
    if ( totalHours < 1000 && totalHours > 99 ) {  
        totalHours = "0"+totalHours.toString();  
    } else if ( totalHours < 100 && totalHours > 9 ) {  
        totalHours = "00"+totalHours.toString();  
    } else if ( totalHours < 10 ) {  
        totalHours = "000"+totalHours.toString();  
    }  
}
```

```

var totalMinutes = Math.floor(calculatedTime/1000/60);
calculatedTime = calculatedTime - totalMinutes*1000*60;
if ( totalMinutes < 10 ) {
    totalMinutes = "0"+totalMinutes.toString();
}

var totalSeconds = Math.floor(calculatedTime/1000);
if ( totalSeconds < 10 ) {
    totalSeconds = "0"+totalSeconds.toString();
}
var sessionTime = "PT";
if (parseInt(totalHours)!=0)
    sessionTime += totalHours + "H";
if (parseInt(totalMinutes) !=0)
    sessionTime += totalMinutes + "M";
if (parseInt(totalSeconds)!=0)
    sessionTime += totalSeconds + "S";

doSetValue("cmi.session_time", sessionTime);
}

```

Figure 9.13b

Another method would be to use "cmi.suspend\_data" or "cmi.comments" as a place to store the start time just after Initialize is called. Just before Terminate is called, this value could be retrieved and calculate the elapsed time.

Code executed right after Initialize:

```

var startTime = new Date().getTime(); // time in seconds
doSetValue( "cmi.suspend_data", startTime );

```

Figure 9.13c

Code executed right before Terminate:

```

var startTime=doGetValue("cmi.suspend_data");
// then use the setSessionTime function shown above

```

Figure 9.13d

## 9.14 Total Time Example

This is a 'read-only' value. The LMS will initialize total\_time to PT0H0M0S the first time the SCO is launched and then use SCO reported values of session\_time to keep an accumulated total.

```

variable = doGetValue("cmi.total_time");

```

Figure 9.14a

## 9.15 Data Storage Area Example

This data is created by the SCO and stored in the LMS. This data field can be used as needed by the SCO. The LMS is required to store at least 4000 characters. If the character string is greater, there is no guarantee that the data will be stored.

```
var retVal = doGetValue("cmi.suspend_data");  
doSetValue("cmi.suspend_data", value );
```

Figure 9.15a



**Best Practice: When designing the suspend data storage format, avoid leading zeros.**

For a learner performance test SCO with two versions of the test, the number of attempts could be stored in "cmi.suspend\_data" to facilitate serving up the correct learner performance test. Avoid leading zeros in this storage area. In some instances, the leading zeros of a number are suppressed.

This field can also store the learner response from a learner performance test when the testing strategy allows suspending and resuming the test. The learner will not be able to view which questions were answered correctly since the results would be stored in "cmi.suspend\_data" and is not viewable by the learner. When the learner resumes the performance test and the SCO retrieves the previous results, then all of the results should be sent after all questions have been answered and scored.

"Cmi.suspend\_data" could be used to store a bookmark within a large Flash file in addition to storing the HTML page of the Flash object in the "cmi.lesson\_location" field. Suspend data can store multiple bookmarks from a Macromedia Flash file when the Flash file has several movies playing with the main file.

## 9.16 Interactions



**9.16-1 Business Rule (Army): Developers of SCO assessments must make use of the SCORM Interactions Data Model Element to record information about the learner's response for validation purposes. Usage of the Interactions Data Model Element must conform to Figure 9.16.1a.**

### 9.16.1 Overview

The **Interactions Data Model Element** is provided to allow SCO developers the ability to record the status of a learner's interaction with an individual test item/question. Interactions are learner responses to individual questions in a SCO. The data stored in an interaction element can be used to evaluate the effectiveness of checks on learning or exam questions, either by the SCO or by some

course management program provided by the LMS vendor. Each test item/question may have its own Interactions Element Data Model, which is a collection of more than ten different pieces of information about the learner's response to a test item/question. The SCORM specification requires LMS vendors to allow for at least 250 of these collections to be stored per SCO. This means that an LMS must allow every SCO in a content package the ability to track learner responses for at least 250 test items/questions. Beyond this minimum, "[t]here is no implied behavior an LMS shall have when interactions are requested to be set, other than storage of the data." The data within an Interactions Data Model Element is not processed by the LMS in any way. The LMS makes no decisions and takes no actions based on the data stored within an interactions element. In previous versions of the SCORM specification, the information written to an interaction could not be read by the SCO, and previously set values could not be overwritten. For SCORM 2004, the data in an Interactions Data Model Element is available to the SCO for use. A SCO may read previously set data from an interactions element created by that SCO, update data stored in an interactions element, or write new data to the interactions element. A SCO may NOT read or write to an interactions element created by another SCO. If an LMS provides the capability, an LMS may report on the data stored within the interactions element of the different SCOs in a content package, but this is not required by SCORM.

<b>Interaction Element Data Fields</b>		<b>Programming code</b>
1) Identifier	Required SCORM	cmi.interactions.n.id
2) Type	Required SCORM	cmi.interactions.n.type
3) Objective IDs	Not Required	cmi.interactions.n.objective.n.id
4) Timestamp	Required Army SCORM (only for timed exams)	cmi.interactions.n.timestamp
5) Correct Responses	Required Army SCORM	cmi.interactions.n.correct_responses.n.pattern
6) Weighting	Required Army SCORM (when question is weighted differently than others in SCO)	cmi.interactions.n.weighting
7) Learner Response	Required Army SCORM	cmi.interactions.n.learner_response
8) Result	Required Army SCORM	cmi.interactions.n.result
9) Latency	Required Army SCORM (only for timed exams)	cmi.interactions.n.latency
10) Description	Not Required	cmi.interactions.n.description

Figure 9.16.1a

Figure 9.16.1a shows the Interactions Element Data Model, and ten different data fields that are stored within each collection for each test item/question. There are more data fields within the Interactions Element Data Model that are not depicted by Figure 9.16.1a, but those shown are the ones that the SCO developer may write or update from within the SCO, and are the only ones that are addressed in this section.

## 9.16.2 SCORM Required Interaction Data Fields

Of the ten data fields in these collections, the 'identifier' and 'type' data fields are required by SCORM to be present whenever the SCO uses the interaction element.

### Identifier

The 'identifier' must be unique from any other identifier used by other interaction elements in that SCO. That means each test item/question in a SCO must have a unique 'identifier'.

The "cmi.interactions.n.id" is specifically formatted as a long identifier type. Refer to the [Data Types for Formatting Values](#) section (9.23).

```
doSetValue("cmi.interactions.[number].id", "[long_identifier_type]");
```

Figure 9.16.2.1a

```
doSetValue("cmi.interactions.0.id", "ARMY-GLOBAL-071-33-3401-1");
```

Figure 9.16.2.1b

### Type

The 'type' data field must be included if there is any use of either the 'correct responses' or 'learner responses' field. This value designates what type of question (multiple choice, true-false, etc.) was delivered to the learner. There is a limited vocabulary for the 'type' field (true\_false, multiple\_choice, fill\_in, long\_fill\_in, matching, performance, sequencing, numeric, other).

```
doSetValue("cmi.interactions.[number].type", "[vocabulary]");
```

Figure 9.16.2.2a

The word 'number', designates a counting mechanism that always begins with "0" and then increases by 1.

Example:

```
doSetValue("cmi.interactions.1.type", "multiple_choice");
```

Figure 9.16.2.2b

Note: The 'number' designated above as "1" can only be used subsequently to using "0".

### 9.16.3 Army SCORM Required Interaction Data Fields

Timestamp (required only for timed SCOs)

The timestamp information field in the interactions element is meant to indicate the point in time at which the test item/question was first given to the learner. This value is set by the SCO. No value will be supplied for the timestamp field by the LMS. It is the SCO's responsibility to set this value. Prior to setting a value for this field, the SCO must first designate the ID field for the interaction element.

This field allows the SCO to indicate the time at which the learner begins working on a test item/question. Once the learner concludes the interaction (in other words, answers the item/question), the SCO may then retrieve the timestamp, compare it to the current time, and compute the total amount of time the learner spent working on the question. This field is helpful in determining the latency information field, especially in situations where a learner is allowed to leave the SCO and reenter while working on the question, but is not the only conceivable way of accomplishing this goal. This information field is also helpful when reviewing test item data for a set of students. Knowing when a learner began a question may be helpful for reviewers examining test item data that relates to time-sensitive material in rapidly changing areas of study. The Army requires that this information be set in timed SCOs.

Correct Responses

This information field defines the correct responses available for a given test item/question. Prior to setting a value for correct responses, the SCO must first define the type information field. The number of correct responses available for an individual question depends on the type of question. A 'true/false' question, for example, will only have one possible correct response, either "true" or "false". While it would be possible to evaluate the correctness of learner's response to a question without using this field of the Interactions Data Model Element, there would be no way to validate the question from the outside without tediously, manually stepping through the SCO. Therefore, this field is critical for test validation.

The "cmi.interactions.n.correct\_responses.n.pattern" value describes the pattern of the correct learner response(s) to the interaction.

Type	Correct Responses Pattern
true_false	true or false
multiple_choice	short_identifier_type[, ]short_identifier_type
fill_in	{case_matters="true"} {order_matters="true"} localized_string_type[, ]localized_string_type
matching	Pairs of identifiers separated by [, ] delimiter. For the pairs, the target and source delimited by [.] . Ex. target[.]source[, ]target[.]source

performance	{order_matters}step_name[.]step_answer[.] step_name[.]step_answer
sequencing	short_identifier_type[.]short_identifier_type
likert	short_identifier_type
long-fill-in	{case_matters="true"}localized_string_type
numeric	min[:]max if lower and upper bound. If min and max are the same, then pattern is that number. If no lower bound, [:]max. If no upper bound, min[:]. Min and Max designated as valid real(10,7).

Figure 9.16.3.2a

Following is a code example for storing the pattern of the correct answer of the test item:

```
doSetValue("cmi.interactions.[number].correct_responses.[sub-  
number].pattern", "[pattern for the correct answer]");
```

Figure 9.16.3.2b

**Note:** The [number] must start at zero and increment in steps of one, otherwise the LMS will not properly record and report results.

Following are code examples of different types of questions and setting the pattern of the correct response:

```
// for multiple choice if the correct answer is C  
doSetValue("cmi.interactions.0.correct_responses.0.pattern", "c");  
  
// for true_false if the correct answer is true  
doSetValue("cmi.interactions.0.correct_responses.0.pattern", "true");  
  
// for fill_in if correct answer is rifle or gun  
doSetValue("cmi.interactions.0.correct_responses.0.pattern", "rifle");  
doSetValue("cmi.interactions.0.correct_responses.1.pattern", "gun");  
  
// for matching if correct answer is 1c, 2a, 3b, 4d  
doSetValue("cmi.interactions.0.correct_responses.0.pattern",  
"1[.]c[, ]2[.]a[, ]3[.]b[, ]4[.]d");
```

Figure 9.16.3.2c

Weighting (required only for weighted questions)

The weighting field is intended to describe the weight given to the test item/question. The specifics of how this field is implemented and used by the SCO are NOT defined by SCORM. The only restriction on this field is that the data be a real number, limited to no more than seven significant decimal figures. If a SCO developer has assigned different weights to test items/questions, it is necessary to use this field.

Learner Response

The learner response field is intended to represent the answer generated by the user in response to the test item/question. Prior to setting a value for correct responses, the SCO must first define the type. The limits on the value of the learner response depend on the type of question, as defined by the type information field of the Interactions Data Model Element. This field is crucial for test validation.

The "cmi.interactions.n.Learner\_response" value is programmed similar to the following:

```
doSetValue("cmi.interactions.[number].learner_response", "[Learner answer]");
```

Figure 9.16.3.4a

**Note:** The [number] must start at zero and increment in steps of one, otherwise the LMS will not properly record and report results.

```
doSetValue("cmi.interactions.10.learner_response", "c");
```

Figure 9.16.3.4b

## Result

The result data field of the interaction element is meant to represent the correctness of the learner response. This field can only be set to one of five values:

Appendix A	"correct"
Appendix B	"incorrect"
Appendix C	"unanticipated"
Appendix D	"neutral"
Appendix E	" [a real number with a precision of seven significant digits] "

Regardless of the value set by the SCO for the result field, the SCO is responsible for passing the correct score for the entire SCO. The LMS will not make any determinations of a learner's success, or score for an assessment SCO, based on the result data field in an interactions element. This field is essential for validation purposes and later tracking of student responses.

The "cmi.interactions.n.result" value indicates how the system judges the described response. The syntax is shown in the example that follows:

```
doSetValue("cmi.interactions.[number].result", "[vocabulary]");
```

Figure 9.16.3.5a

The following is a valid working example of the API function call:

```
doSetValue("cmi.interactions.0.result", "correct");
```

Figure 9.16.3.5b



**Best Practice: Results of the test item (correct/incorrect) could be stored in an array using JavaScript and sent to the LMS after scoring the learner performance test using a 'for loop' as shown in the following example:**

```
var arResult = new Array();

// storing correct/incorrect in array for each test item
function storeResult(result) {
    var i = arResult.length;
    arResult[i] = result;
}

// send results to LMS after scoring
function sendResults(arResult) {
    for (i=0;i<arResult.length;i++) {
        doSetValue("cmi.interactions." + i + ".result", arResult[i]);
    }
}
```

Figure 9.16.3.5c

When using Flash to develop a learner performance test, it may be easier to use an array inside of Flash to store the test item results. Sending the Flash array to JavaScript may require converting the array into a string and sending the string to the sendResults function. Use the split method to store the string back into an array as shown in the following example:

```
/* Question results (correct/incorrect) must be sent to the LMS
   after the learner performance test has been scored */

/* Flash sends a string to a Javascript function with the
   Comma as the string delimiter */

function sendResults(stringList) {
    var resultsArray = stringList.split(",");
    for(i=0;i<resultsArray.length;i++) {
        if (resultsArray[i] ) {
            doSetValue("cmi.interactions." + i + ".result",
resultsArray[i]);
        }
    }
}
```

Figure 9.16.3.5d

Latency (required only for timed SCOs)

The latency field is intended to represent the time elapsed between the time the test item/question was given to the learner, and the time the learner answered the question. The value of this field is limited to the **timeinterval** data type as defined by SCORM. This value must be set by the SCO. No value will be supplied for the latency field by the LMS. It is the SCO's responsibility to compute and set this value. This data could be very informative when analyzing test item data. The amount

of time learners take to complete an interaction may be an indication of the difficulty of the question.

The latency field of the interaction element has no relation to the "cmi.total\_time" data element, or any other element outside the Interactions Data Model Element, in the SCORM Run-Time Environment. The LMS will make no determination about the timing of a SCO based upon this information field. SCO developers must use other elements within the SCORM Run-Time Environment to handle 'timed SCOs'.

```
doSetValue("cmi.interactions.[number].latency", "[time]");
```

Figure 9.16.3.6a

```
doSetValue("cmi.interactions.0.latency", "PT12S");
```

Figure 9.16.3.6b

#### 9.16.4 Not Required Interaction Data Fields

##### Objective ID

Each test item/question may specify up to ten different objective IDs within its interaction element. The Objective ID field is a label for objectives associated with the interaction. These labels may or may not relate to an actual objective id as found in the 'Objectives Data Model Element' within the SCORM Simple Sequencing and Navigation model. In other words, this field does not need to relate to any objective that is being tracked by the LMS. The objectives designated by these labels may relate to learning objectives, and may be present for reporting purposes only. Regardless of the objective ID's association; it is the responsibility of the SCO to act on the data, if any action is required. The LMS will not make any decisions, or process any information, based on an objective ID.

##### Description

The description field is intended to be a brief, informative description of the test item/question. This field is merely information supplied by the developer about the question, and is unnecessary for any purpose other than explanation.

#### 9.16.5 Test Item Data Collection Example

Following are shown code examples of different types of learner scenarios in a learner performance test and the coordinating cmi.interactions data model:

<i>Multiple Choice, Correct Learner Response</i>	<code>doSetValue("cmi.interactions.0.id", "ARMY-071-33-3401-1"); doSetValue("cmi.interactions.0.type", "choice"); doSetValue("cmi.interactions.0.correct_responses.0.pattern", "b"); doSetValue("cmi.interactions.0.learner_response", "b"); doSetValue("cmi.interactions.0.result", "correct");</code>
<i>Multiple Choice, Incorrect Learner Response</i>	<code>doSetValue("cmi.interactions.0.id", "ARMY-071-33-3401-1"); doSetValue("cmi.interactions.0.type", "choice"); doSetValue("cmi.interactions.0.correct_responses.0.pattern", "b"); doSetValue("cmi.interactions.0.learner_response", "a"); doSetValue("cmi.interactions.0.result", "incorrect");</code>
<i>True-False, Correct Learner Response</i>	<code>doSetValue("cmi.interactions.0.id", "ARMY-071-33-3401-1"); doSetValue("cmi.interactions.1.type", "true-false"); doSetValue("cmi.interactions.1.correct_responses.0.pattern", "t"); doSetValue("cmi.interactions.1.learner_response", "t"); doSetValue("cmi.interactions.1.result", "correct");</code>
<i>True-False, Incorrect Learner Response</i>	<code>doSetValue("cmi.interactions.0.id", "ARMY-071-33-3401-1"); doSetValue("cmi.interactions.1.type", "true-false"); doSetValue("cmi.interactions.1.correct_responses.0.pattern", "t"); doSetValue("cmi.interactions.1.learner_response", "f"); doSetValue("cmi.interactions.1.result", "incorrect");</code>

Figure 9.16.5a

One type of data that is collected is whether the learner answered the test item correctly or not. This data model is the "cmi.interactions.[item number].result". Concerns have arisen about the possibility of a learner viewing the test item results (correct/incorrect) before completion of the learner performance test. Refer to the [Result Example](#) section (9.16.5) for more detail.

When the learner skips questions and then has to go back to the questions, the cmi.interactions should be overwritten using the same "n" number.

### 9.17 Comments from the Learner Example

The "cmi.comments\_from\_learner.n.comment" value is an optional data field to store learner feedback. Following is an example:

```
doSetValue("cmi.comments_from_learner.[n].comment", value);  
variable = GetValue("cmi.comments_from_learner.[n].comment");
```

Figure 9.17a

## 9.18 External References Example



**Best Practice: External references should not link directly to the URL.**

SCORM allows access to learning content by URLs, although the Army has chosen not to recommend this method within Army courseware. Refer to the [Location of Learning Content](#) section (3.1.5). The Army allows external references that refer to non-learning content. This content is beyond what the course has control over. External references could be used for ARs, Field Manuals, TRs, Pams, Regs, etc., as long as access to this material is not necessary for successful completion of the course content or examinations.



**Best Practice: External references should link to a local redirect page. This method would allow ease of maintenance and conversion.**

For courses to be easily converted to CD-ROM stand alone courseware, all external references within the courseware should link to a local redirect page. For the online Web version, this redirect page will have the URL outside of the local domain. For the courseware to be used from a CD-ROM, the external reference itself would be copied or downloaded and stored locally on the CD-ROM. Programming code on this page will contain a statement directing the learner to either the URL (for online) or to the local resources that are contained within the package.

The visual representation is as follows:

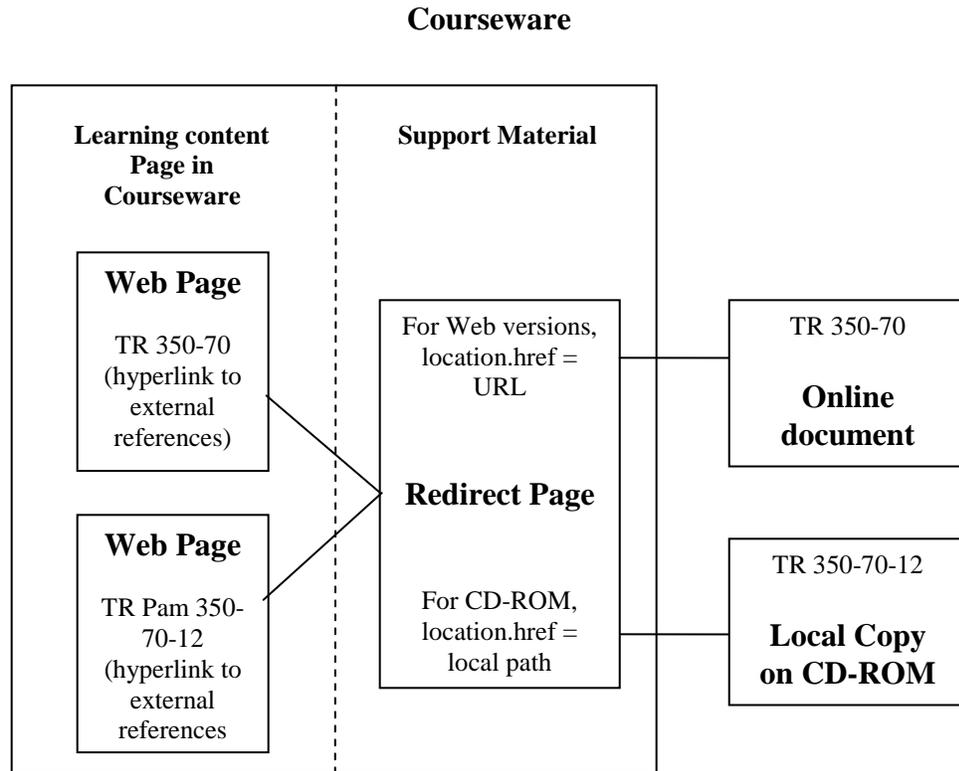


Figure 9.18a

Editing the redirect page is less time intensive than editing the SCO within the authoring tool, which requires the original files. Manual editing can easily occur on each redirect page by changing the URL to a local path. The filename for the redirect page should easily identify that it is a URL redirect page for a specific Web site, perhaps in a 'redirect' folder.

A conditional programming statement can also be used to code an external reference. Set an LMS variable to "true" in the top frame of the SCO for the LMS version. Change the variable to "false" for the CD-ROM version. An example of a conditional statement is as follows:

```
// set the variable to true if using an LMS
// set to false if using CD-ROM
var LMS=true;

//
// redirect to URL if using an LMS or if not, redirect to local path
if (LMS)
    document.location.href= [URL_of_Online_Resource];
else
    document.location.href=[relative path to downloaded resource];
```

Figure 9.18b

Building on this method could include one redirect page for all external references using a JavaScript function for multiple redirects.

Following is an example of multiple redirects using a single JavaScript function:

```
function redirectURL(page) {  
    if (page=="atsc") {  
        document.location.href = "http://www.atsc.army.mil";  
    }  
    if (page=="tradoc") {  
        document.location.href = "http://www.tradoc.army.mil";  
    }  
    if (page=="cnn") {  
        document.location.href = "http://www.cnn.com";  
    }  
}
```

Figure 9.18c

## 9.19 Learner Directions Page

At certain points in the course, a page for learner directions may be needed. The page can display in the right frame at the termination of an activity during the course.

After sending a "failed" success status to the LMS for the pretest with a control mode of 'choice', then a selection direction page may be initiated:

```
<script type="text/javascript">  
...  
// already sent fail to LMS for success status of pretest  
  
    document.location.href = "../index_select.html";  
  
    doTerminate();  
}
```

Figure 9.19a

After sending a failed success status to the LMS for Posttest #1 with a control mode of 'choice', then a remediation direction page can be initiated:

```
<script type="text/javascript">
...
// already sent fail to LMS for success status of posttest 1
    document.location.href = "../index_remed.html";
    doTerminate();
}
```

Figure 9.19b

## 9.20 Navigation Requests Examples

Navigation requests are contained in the Run-Time Environment because the navigation requests communicate to the LMS to launch the next target activity or a specific activity.

The SCO has the ability to designate the next target activity by providing navigation code within the SCO and is invisible to the learner. The SCO communicates the target request to the LMS using the Run-Time Environment. It can be a request for a specific activity or a request for the next logical activity according to the activity tree. Upon the SCO's termination, the LMS launches the next target activity according to the SCO's navigation request.

If an activity requires a specific follow-on activity in such cases like remediation, help, or glossary, then a navigation request could be implemented in the Run-Time Environment. Navigation requests tell the LMS to 'jump' to a specific SCO as a follow-on activity.



### 9.20-1 Business Rule (Army): A SCO cannot hyperlink to another SCO.

Note: The term hyperlink, as stated in Business Rule 9.20-1, indicates a transfer of control to another SCO. The SCO that has been initialized is the only SCO that communicates with the LMS. Navigation requests are not the same as the SCO hyperlinking to another SCO.

### 9.20.1 Navigation Request According to the Activity Tree

Following is an example of a navigation request by a SCO. The SCO provides a navigation request to the LMS and, upon termination, the LMS will launch the target activity according to the pre-ordered path of the activity tree:

```
doSetValue("ad1.nav.request", "continue");
doTerminate();
```

Figure 9.20.1a

## 9.21 Graded Assessments Examples

The examples in this section assume use of the API Wrapper file because state management API function calls are already included in the API Wrapper file and, therefore, are not needed in the actual programming code. The examples are also set up similarly.

### 9.21.1 Non-Resumable Learner Performance Test

Testing Strategy Example for Non-Resumable Learner Performance Test:

- a). Learner able to skip questions and go back to questions.
- b). Learner can exit out of the learner performance test anytime before submitting answers and receive an "incomplete".
- c). Learner cannot resume from where the learner left off (no bookmarks).
- d). Score will be displayed to learner after scoring.
- e). After scoring, provide test review. Do not hyperlink to other SCOs.
- f). Learner is unable to go back to questions after scoring.
- g). Learner score is sent by the SCO and stored in the LMS.
- h). Success Status (Passed/Failed) is stored in the LMS.
- i). Learner has unlimited attempts with alternate learner performance test versions.
- j). Scaled passing score is contained within the manifest and stored in the LMS.
- k). There are two SCOs for the learner performance test: Version 1 and Version 2.

Following is the pseudo code applying the above testing strategy:

<b>Description</b>	<b>Event</b>	<b>Pseudo-code for SCORM Run-Time Data Model</b>
First Page of Test	Launch	Send Initialize Begin timer function Retrieve Scaled passing score from the LMS Retrieve Completion Status and Success Status If Completion Status is not attempted, deliver Version A of the learner performance test If Success Status is failed, deliver Version B of the learner performance test Send Completion status of incomplete Increment number of attempts
Learner progresses through Learner performance test	Next and Back Button	Send cmi.interactions.n.type Send cmi.interactions.n.correct_responses.n. pattern Send cmi.interactions.n.learner_response Store cmi.interactions.n.result in an array Send Commit
	Close button	Send cmi.session_time Send cmi.exit Completion status is already set to incomplete Send Terminate
Last Page Submit Answers	Calc score and status.	Calculate learner score Send score with cmi.score.raw If failed, send success status to failed If passed, send cmi.success_status of passed Send Commit
	Display score	Display score to learner Send test item results from array using cmi.interactions.n.result
	Test Review	// Display Test Review of questions missed for learner to review
	Close SCO	Send cmi.session_time Send cmi.exit Send completion status of completed Store number of attempts to cmi.suspend_data Send Commit Send Terminate

Figure 9.21.1a

### 9.21.2 Timed Non-Resumable Learner Performance Test

A timed learner performance test has a maximum time limit. If the maximum time limit has been reached, a particular action is taken. The timing should be used only when necessary to verify mastery in accordance to the TLO standard.

For example, a learner has launched a lesson, but the maximum time allowed in the SCO is 30 minutes. The time elapsed in the session of the SCO would be continuously compared with the maximum time allowed. Either the session time would be written to the LMS upon learner exit or the time limit action would occur.

Testing Strategy Example for a Timed Non-Resumable Learner Performance Test:

- a). Learner is able to skip questions and go back to questions.
- b). Learner must complete the entire learner performance test to receive credit and if exit prematurely out of the learner performance test, will receive "incomplete".
- c). Score will be displayed to learner after scoring.
- d). Learner unable to go back to questions after scoring.
- e). Score is calculated by the SCO and sent to the LMS to be stored.
- f). Scaled passing score is contained within the manifest and stored in the LMS.
- g). Success Status (Passed/Failed) is determined by the SCO.
- h). When time exceeded, learner receives a message and must exit the learner performance test after maximum time allowed.
- i). There is one version of the learner performance test.

Following is the pseudo code applying the above testing strategy:

Description	Event	Pseudo code for SCORM Run-Time Data Model
First Page of SCO- Introductory Page	Launch	Send Initialize Begin timer function Retrieve Scaled passing score from LMS  Retrieve Success Status  If success status is passed, deny access If completion status is equal to not attempted, then retrieve max_time_allowed and time_limit_action  Set completion status to incomplete in conditional statement
First Test Item; Learner progresses through learner	Next Button	Use setTimeout function to determine when session_time exceeds max_time_allowed  Send cmi.interactions except result

Description	Event	Pseudo code for SCORM Run-Time Data Model
performance test		
Submit Answers	Calc score	Calculate learner's score and display to learner
	Send Status	If failed, set success status to failed If passed, set success status to "passed" Send results Send Commit
	Close button	Send Terminate

Figure 9.21.2a

Two specific API function calls and data model needed for a timed learner performance test are:

```
var mta = doGetValue("cmi.max_time_allowed");
var tla = doGetValue("cmi.time_limit_action");
```

Figure 9.21.2b

### 9.21.2.1 Maximum Time Allowed (Time Limit) Example

This value indicates a time limit or the amount of time the learner is allowed to have in the current attempt on the SCO. This time limit is required for 'timed' learner performance tests.

Maximum time allowed must be specified within the manifest using the sequencing tag of <imsss:attemptAbsoluteDurationLimit> tag for an <item> that references a SCO resource. This value, when input into the LMS through the manifest, is stored in "cmi.max\_time\_allowed". Format for this value is PTnHnMnS where n is the numeric value.

Note: SCORM recommends the use of the time limit with caution.

The "cmi.max\_time\_allowed" value is required to be retrieved in timed learner performance tests to determine whether the maximum time allowed in the SCO has been exceeded.

Following is an example of retrieving the maximum time allowed value:

```
variable=doGetValue("cmi.max_time_allowed");
```

Figure 9.21.2.1a

The following programming logic reflects a timed learner performance test SCO:

```
Get Value of max time allowed stored in the LMS
Convert PTnHnMnS to milliseconds

Use a setTimeout function to execute when time has elapsed
```

Figure 9.21.2.1b

### 9.21.2.2 Time Limit Action Example

The time limit action value is used to indicate to the SCO what the action should occur when the maximum time allowed in the SCO has been exceeded.

There are four vocabulary options; to display a message to the learner and two do not. It is recommended that two of the vocabularies, which display a message to the learner, be used.

Following is an example of retrieving the time limit action from the LMS:

```
variable=doGetValue("cmi.time_limit_action");
```

Figure 9.21.2.2a

This value is used to indicate to the SCO what the action should be when the maximum time allowed in the SCO has been exceeded.

Two recommended vocabulary values are "exit,message" and "continue,message".

Time limit action value is external to the SCO and is specified within the manifest in the <adlcp:timeLimitAction> tag within the parent <item> tag.

```
// declaring variables
var mta_ms=0;
var timeAction = "";

function calculate() {
    // retrieving values from LMS
    var maxTimeAllowed=doGetValue("cmi.max_time_allowed");
    var timeAction=doGetValue("cmi.time_limit_action");

    // *convert maxTimeAllowed to milliseconds
    mta_ms = convertToMilliseconds(maxTimeAllowed);

    /* Use a setTimeout JavaScript function to execute after the time has
    elapsed */
    setTimeout("noMoreTime(timeAction)", mta_ms );
}

function convertToMilliseconds(mta) {
    var seconds = convertToSeconds(mta);
    var mta_ms = parseInt(seconds) * 1000;
    return mta_ms;
}
```

```

// maxtimeallowed is in format PTnHnMnS.  if n=0, letter is omitted
function convertToSeconds(mta) {
    var totalSeconds=0;

    arTime = mta.split("T");
    var entireTimeFormat = arTime[1];

    if (entireTimeFormat.indexOf("H") != -1) {
        arTime1 = entireTimeFormat.split("H");
        var hours = arTime1[0];
        totalSeconds = parseInt(hours) * 60 * 60;
        var minSecFormat = arTime1[1];
    } else {
        minSecFormat = entireTimeFormat;
    }

    if (entireTimeFormat.indexOf("M") != -1) {
        arTime2 = minSecFormat.split("M");
        var minutes = arTime2[0];
        var show1 = parseInt(minutes) * 60;
        totalSeconds += parseInt(minutes) * 60;
        var secFormat = arTime2[1];
    } else {
        secFormat = entireTimeFormat;
    }

    if (entireTimeFormat.indexOf("S") != -1) {
        arTime3 = secFormat.split("S");
        var seconds = arTime3[0];
        totalSeconds += parseInt(seconds);
    } else {
        //alert("There are no seconds");
    }

    return totalSeconds;
}

function noMoreTime(timeAction) {
    switch (timeAction) {
        case "exit,message":
            alert("Your time has expired. The session will terminate.");
            //call terminate and exit
            break;
        case "continue,message":
            alert("Your time has expired and the session will be marked
accordingly.\nYou may continue the test.");
            // call terminate and exit
            break;
    }
}

```

Figure 9.21.2.2b

### 9.21.3 Resumed Learner Performance Test

A resumed learner performance test is when a learner may exit the learner performance test before submitting the learner's answers to be scored, bookmarking the point at which the learner exited the learner performance test, and upon re-entry, place the learner where the learner left off.

Testing Strategy Example for Resumed Learner Performance Test:

- a). Learner able to skip questions and go back to questions.
- b). Learner will receive an "incomplete" when suspending a learner performance test.
- c). When learner exits, test item data and bookmark are stored.
- d). Bookmark is retrieved upon re-entry to the learner performance test and learner is directed to the page where the learner left off.
- e). Learner may begin again at the bookmark but cannot go back to questions answered during previous session.
- f). Scaled passing score is contained within the manifest and stored in the LMS.
- g). Score will be displayed to learner after scoring.
- h). Learner is unable to go back to questions after scoring.
- i). Learner Score is calculated by the SCO and sent to the LMS to be stored.
- j). No limit to the number of attempts to take the learner performance test.
- k). Success status of "failed" or "passed" is to be sent by the SCO.
- l). There is one version of the learner performance test.
- m). Once passed, the learner is denied re-entry to the learner performance test.

Following is the pseudo code applying the above testing strategy:

<b>Description</b>	<b>Event</b>	<b>Pseudo code for SCORM Run-Time Data Model</b>
First Page of Test	Launch	Send Initialize Begin timer function Retrieve Scaled passing score from the LMS Retrieve success status If success status equals passed, deny re-entry If completion status equals incomplete, retrieve stored test item data and bookmark and send learner to bookmark If completion status equals not attempted, send incomplete and send Commit
Learner progresses through learner performance test	Next, Back buttons	Send cmi.interactions to LMS Store result in array Send Commit
	Close Button	Send Bookmark Store test item data for future session Send Commit Send Terminate
Last Page - Submit Answers	Calc score	Calculate score, store in LMS and display to learner
	Lesson Status	If learner fails, send success status to failed If learner passes, send success status of passed Send results array to the LMS Send Commit
	Close button	Send Terminate

Figure 9.21.3a

## 9.22 Differences in SCORM v1.2 Data Model and the SCORM 2004 Data Model

SCORM v1.2 DATA MODEL	SCORM 2004 DATA MODEL
N/A	cmi._version
cmi.core.student_id	cmi.learner_id
cmi.core.student_name	cmi.learner_name
cmi.core._children	N/A
cmi.core.lesson_location	cmi.location
cmi.core.credit	cmi.credit *(LMS specific; SCORM doesn't require that an LMS support this element)
cmi.core.lesson_status	cmi.completion_status AND cmi.success_status
cmi.core.entry	cmi.entry
cmi.core.score._children	cmi.score._children
cmi.core.score.raw	cmi.score.raw (No longer impacts status; New data element of cmi.score.scaled now impacts status)
cmi.core.score.max	cmi.score.max
cmi.core.score.min	cmi.score.min
cmi.core.total_time	cmi.total_time
cmi.core.lesson_mode	cmi.mode *(LMS specific; no mechanism in place to determine the mode of a SCO)
cmi.core.exit	cmi.exit
cmi.core.session_time	cmi.session_time
cmi.suspend_data	No change
cmi.launch_data initialized from <adlcp:datafromlms>	cmi.launch_data initialized from <adlcp:dataFromLMS>
N/A	cmi.comments_from_learner._children
N/A	cmi.comments_from_learner._count
cmi.comments	cmi.comments_from_learner.n.comment
N/A	cmi.comments_from_learner.n.location
N/A	cmi.comments_from_learner.n.timestamp
N/A	cmi.comments_from_lms._children
N/A	cmi.comments_from_lms._count
cmi.comments_from_lms	cmi.comments_from_lms.n.comment *(outside the scope of SCORM)
N/A	cmi.comments_from_lms.n.location *(outside the scope of SCORM)
N/A	cmi.comments_from_lms.n.timestamp *(outside the scope of SCORM)
cmi.student_data._children	N/A
cmi.student_data.mastery_score initialized from <adlcp:masteryscore>	cmi.scaled_passing_score initialized from <imsss:minNormalizedMeasure>
cmi.student_data.max_time_allowed initialized from <adlcp:maxtimeallowed>	cmi.max_time_allowed initialized from <imsss:attemptAbsoluteDurationLimit>
cmi.student_data.time_limit_action initialized	cmi.time_limit_action initialized from

<b>SCORM v1.2 DATA MODEL</b>	<b>SCORM 2004 DATA MODEL</b>
from <adlcp:timelimitaction>	<adlcp:timeLimitAction>
cmi.objectives._children	No change
cmi.objectives._count	N/A
cmi.objectives.n.id	N/A
cmi.objectives.n.score._children	No change
N/A	cmi.objectives.n.score.scaled
cmi.objectives.n.score.raw	No change
cmi.objectives.n.score.min	No change
cmi.objectives.n.score.max	No change
cmi.objectives.n.status	N/A
N/A	cmi.objectives.n.success_status
N/A	cmi.objectives.n.completion_status
N/A	cmi.objectives.n.progress_measure
N/A	cmi.objectives.n.description
N/A	cmi.progress_measure
cmi.student_preference._children	cmi.learner_preference._children
cmi.student_preference.audio	cmi.learner_preference.audio_level
cmi.student_preference.language	cmi.learner_preference.language *(outside the scope of SCORM)
cmi.student_preference.speed	cmi.learner_preference.delivery_speed *(outside the scope of SCORM)
cmi.student_preference._text	cmi.learner_preference.audio_captions
cmi.interactions._children	No change
cmi.interactions._count	No change
cmi.interactions.n.id	No change
cmi.interactions.n.time	cmi.interactions.n.timestamp
cmi.interactions.n.type	No change
cmi.interactions.n.objectives._count	No change
cmi.interactions.n.objectives.n.id	No change
cmi.interactions.n.correct_responses._count	No change
cmi.interactions.n.correct_responses.n.pattern	No change
cmi.interactions.n.weighting	No change
cmi.interactions.n.student_response	cmi.interactions.n.learner_response
cmi.interactions.n.result	No change
cmi.interactions.n.latency	No change
N/A	cmi.interactions.n.description
N/A	cmi.completion_threshold

Figure 9.22a

\*See beginning of the Programming Examples section (13) for more information.

### 9.23 Data Types for Formatting Values

Each of the data model elements has a defined data type and is formatted according to one of the following data types:

<b>Data Type</b>	<b>Definition of Data Type</b>
Characterstring	A string of characters that are defined in ISO 10646, which is equivalent to the Unicode Standard.
Localized string type	A characterstring that has an indicator of the language of the characterstring.
Language Type	A data type to represent a language defined by ISO 639-1 and 639-2 for the langcode and ISO 3166-1 for the subcode.
Long Identifier Type	A unique characterstring that conforms to the syntax defined for Universal Resource Identifiers (URI), refer to Request for Comments (RFC) 2396.
Short Identifier Type	A unique characterstring that conforms to the syntax defined for Universal Resource Identifiers (URI), refer to RFC 2396.
Integer	A member of the set of positive whole numbers, negative whole numbers and zero.
State	Some data model elements have a defined set of states. Each state is found to a reserved token. These are further defined in the Data Model Element Implementation Requirements section in each data model Run-Time Environment explanation.
Real	
Time	
Time Interval	

Figure 9.23a

## 9.24 Metadata Examples

Content Organization metadata is located in the <organizations> section within the manifest file, and SCO metadata is located within the <resource> sections that are defined as having a scormType of "sco" within the manifest file.

SCORM requires that all manifest files contain the schema and schema version to which the metadata is created. This XML binding is shown in the following example:

```
<manifest>
  <metadata>
    <schema>ADL SCORM</schema> (REQUIRED)
    <schemaversion>2004 3RD EDITION</schemaversion>
(REQUIRED)
  </metadata>
  <organizations>
    ...
  </organizations>
</manifest>
```

Figure 9.24a

### 9.24.1 XML Binding for Separate Metadata Files



**9.24.1-1 Business Rule (Army): Metadata must be in separate XML files and referenced from within the manifest file using the XML binding for separate metadata file references.**

The XML binding to be used within the manifest that references the relative path for the separate XML metadata files is shown in the following example.

XML binding for separate metadata files:

```
<metadata>
  <adlcp:location>[relative path from the imsmanifest.xml file to the
  metadata file]</adlcp:location>
</metadata.>
```

Figure 9.24.1a

## 9.24.2 Relative Path Example

Following is an example of a relative path from the 'imsmanifest.xml' file:

If the physical file structure is:

- unitsafety folder
  - metadata folder
    - sco1\_metadata.xml
    - sco2\_metadata.xml
  - sco1 folder
  - sco2 folder
  - images folder
  - references folder
  - glossary folder
- imsmanifest.xml file
- imscp\_v1p1.xsd
- adlseq\_v1p3.xsd
- imsss\_v1p0.xsd
- adlcp\_v1p3.xsd
- adlnav\_v1p3.xsd

If the metadata file name is sco1\_metadata.xml and located in the 'metadata' folder, then, using this scenario, the <adlcp:location> tag within the manifest would be 'metadata/sco1\_metadata.xml', which is the path from the location of the 'imsmanifest.xml' file to the 'sco1\_metadata.xml' file. In the above file structure, any SCO can be extracted and reused in another content aggregation.

In essence, a metadata reference is contained within the manifest file and the actual metadata XML file is located within the physical files.

## 9.24.3 Content Organization Metadata Referenced on the Manifest

Content Organization Metadata is located in the <organization> section of the manifest.

Following is an example of the placement of the Content Organization metadata reference within the manifest:

```
<organization>      (opening tag for organization)
  <item>...</item>
  <item>...</item>
  <item identifier="B10">
    <item>...</item>
    <item>...</item>
    <item>...</item>
  </item>
  <metadata>      (Content Organization metadata for <organization>)
    <adlcp:location>metadata/co_unitsafety.xml</adlcp:location>
  </metadata>
</organization> (closing tag for organization)
```

Figure 9.24.3a

Content Organization metadata as shown here is 'co\_unitsafety.xml'.

#### 9.24.4 SCO Metadata Referenced on the Manifest

The SCO metadata reference is located in the <resource> section within the manifest file.

Following is an example of the placement of the SCO Metadata reference:

```
<resource identifier="A23" type="webcontent" adlcp:scormType="sco"
href="launchfile.htm">
  <metadata>      (SCO metadata)
    <adlcp:location>metadata/unit.xml</adlcp:location>
  </metadata>
  <file>...</file>
  <file>...</file>
</resource>
```

Figure 9.24.4a

#### 9.24.5 Impact of File Path Offset Option on Relative Paths in the Manifest

Within SCORM, there is an optional value named 'XML Base' that can be applied to relative paths (for example, 'unit1/safety/chemical/index.html') in the manifest file. The XML Base is used to designate a file path offset to the relative path. For example, the relative path of 'unit1/safety/chemical/index.html' can be shortened to an xml:base of 'unit1/safety/' and the relative path of 'chemical/index.html'.

This attribute can be applied in any combination of three ways:

- Apply to the entire manifest file
- Apply to the entire resources section
- Apply to an individual resource tag

Each XML Base value impacts each relative path. Following is an example of the XML Base being used in all three ways:

```
<manifest identifier="Army_88H10" xml:base="unit1/">
<organizations>...</organizations>
<resources xml:base="safety/">
  <resource identifier="R_SC012" type="webcontent" scormType="sco"
href=chemical/index.html" xml:base="chemical/">
    <metadata>
      <adlcp:location>chemical/sco33.xml</adlcp:location>
    </metadata>
    <file href="index.html"/>
    <file href="page2.html"/>
    <file href="images/graphic1.html"/>
  </resource>
</resources>
</manifest>
```

Figure 9.24.5a

The 'graphic1.html' file is located at 'unit1/safety/chemical/images/graphic1.html' relative to the courseware root folder, which is the location of the manifest file. This file path offset is taken into account in any software parsing the manifest, and programmers must be aware of its existence. A trailing slash is required to be at the end of any XML Base value.

## 9.25 Developing the SCORM Metadata Files

Here is a macro example of how a separate metadata file named 'sco1.xml' should appear:

```
<?xml version="1.0"?>
<lom xmlns="http://ltsc.ieee.org/xsd/LOM"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://ltsc.ieee.org/xsd/LOM lomStrict.xsd">

  <general>...</general>
  <lifeCycle>...</lifeCycle>
  <metaMetadata>...</metaMetadata>
  <technical>...</technical>
  <rights>...</rights>
  <classification>...</classification>
  <classification>...</classification>
  <classification>...</classification>
  <classification>...</classification>
  <classification>...</classification>
</lom>
```

Figure 9.25a

Following is a Cross-Reference of Metadata Values:

<b>Metadata Required</b>	<b>XML Metadata Tag for Programmer</b>
Catalog Identifier	general.identifier.catalog (value = ATIA)
Entry Identifier	general.identifier.entry (value = TBD)
Title of Learning Resource	general.title
Language of Learning Resource	general.language
Description of Learning Resource	general.description
Keywords	general.keyword
Type of Metadata	general.aggregationLevel
Version of Learning Resource	lifeCycle.version
Status of Package Submittal	lifeCycle.status (value = final)
Proponent's Role	lifeCycle.contribute.role (value = publisher)
Proponent Name, Address, E-mail	lifeCycle.contribute.entity
Date of Submittal	lifeCycle.contribute.date
Metadata Catalog Identifier	metaMetadata.identifier.catalog (value = TBD)
Metadata Entry Identifier	metaMetadata.identifier.entry (value = TBD)
Metadata Specification Used to Create this Metadata	metaMetadata.metadataSchema (value = LOMv1.0, value = SCORM_CAM_1.3, value = ADL_v1.0)
Language of Metadata file	metaMetadata.language
File Format (MIME types)	technical.format
Cost of Learning Resource	rights.cost (value = none)
Copyright and Other Restrictions	rights.copyrightAndOtherRestrictions (value = none)
MOS and Skill Level	classification (value = "discipline")
SQI	classification (value = "discipline")
ASI	classification (value = "discipline")
Task Numbers and Task Descriptions	classification (value = "educational objective")
Learning Objectives (Action, Condition, Standard)	classification (value = "educational objective")
508 Compliant	classification (value = "accessibility restrictions")
Security Level (Foreign disclosure)	classification (value = "security level")
ADL-R Requirement	Classification (value = "collection")

Figure 9.25b

### 9.25.1 Catalog and Entry Identifier

The catalog where this learning resource is stored is designated with the tag `general.identifier.catalog` and the entry is designated with the tag `general.identifier.entry`.

The `general.identifier` metadata element represents a mechanism for assigning a globally unique label and contains two sub-elements: `catalog` and `entry`. These two sub-elements will designate the

catalog and entry identification used by the Government (ATSC) to store the courseware after delivery. The catalog element must be "ATIA" and the entry element must be "TBD".

Following is an example of designating the catalog and entry identifier:

```
<general>
...
  <identifier>
    <catalog>ATIA</catalog>
    <entry>TBD</entry>
  </identifier>
...
</general>
```

Figure 9.25.1a

### 9.25.2 Title of Learning Resource

The title of the learning resource is designated with the tag `general.title`.

The `general.title` metadata element describes the name given to the learning resource. For SCOs or aggregations, this value must contain the same title used in the manifest file.

Following is an example of `general.title`:

```
<general>
  <title>
    <string>Introduce Basic M60 Machine Gun Training</string>
  </title>
...
</general>
```

Figure 9.25.2a

### 9.25.3 Language of Learning Resource

The language (English, Spanish, German, etc.) of the learning resource is designated with the tag `general.language`.



**ADL Reference:** The `general.language` metadata element is required for SCOs and Content Organizations. This element must use the language code as defined by ISO 639:1988 and, if needed, the language subcode from ISO 3166-1997.



**Note:** SCORM 2004 3<sup>rd</sup> Edition defines the `general.language` element as being able to contain a value of "none", which indicates no lingual content.

The language of the learning resource content is specified in this element with the two-character country code.

Following are several sites that list the language codes:

- <http://www.unicode.org/onlinedat/languages.html>
- <http://www.oasis-open.org/cover/iso639a.html>
- <http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt>
- <http://alis.isoc.org/langues/iso639.en.htm>

Following is an example of general.language:

```
<general>
  ...
  <language>en</language>           (English)
  <language>de</language>          (German)
  ...
</general>
```

Figure 9.25.3a

#### 9.25.4 Description of Learning Resource

The description of the learning resource is designated with the tag `general.description`.

The 'general.description' metadata element must contain a general description of the object for Content Organizations and SCOs. This element must exclude any reference to hierarchy (lesson, phase, module, etc.).

Following is an example of `general.description`:

```
For Content Organizations:
<general>
...
  <description>
    <string>Basic instruction on U.S. Army Infantry M60 Machine Gun
    laying using field expedients, range card preparation, maintenance,
    function check performance, loading, unloading, malfunction corrections
    and target engaging
    </string>
  </description>
...
</general>

For SCOs:
<general>
...
  <description>
    <string>Explanation of the field expedient method of laying an
    M60 on preselected targets using the notched stake/tree crotch method
    </string>
  </description>
...
</general>
```

Figure 9.25.4a

### 9.25.5 Keywords

Keywords are designated with multiple tags of `'general.keyword'`.

The `'general.keyword'` metadata element contains keywords or phrases describing this learning resource. One or more keyword elements are derived from description. Enter words and phrases that accurately and precisely define the object.

It is up to the content developer to determine what is represented as this string value.

Following is an example of `'general.keyword'`:

```
<general>
...
  <keyword><string>Infantry</string></keyword>
  <keyword><string>M60 laying using field expedients</string></keyword>
  <keyword><string>Notched Stakes</string></keyword>
...
</general>
```

Figure 9.25.5a

### 9.25.6 Type of Metadata

Since there are five types of metadata, the type of metadata must be designated with the tag 'general.aggregationLevel' to determine proper testing requirements. The value is converted to a numeric value being "2" for a SCO or "3" for a Content Organization.

The general.aggregationlevel metadata element must designate which of the two types of resources the metadata is describing that the Army requires: SCO metadata or Content Organization metadata.



**ADL Reference:** The SCORM specification states that this data element container describes the "functional granularity of the learning resource." The vocabularies defined for this element are restricted vocabularies as follows: "1" the smallest level of aggregation, for example, raw media data or fragments; "2" a collection of level 1 learning objects, for example, a lesson; "3" a collection of level 2 learning objects, for example a course; "4" the largest level of granularity, for example, a set of courses that lead to a certificate.

Since there are four types of learning resources contained in SCORM, the Army has interpreted the above vocabulary to mean the following:

- "1" indicates that an Asset is being described
- "2" indicates that a SCO is being described
- "3" indicates that a Content Organization is being described (Course)
- "4" indicates that a Content Aggregation (Package) is being described

Following is an example of the metadata tag for SCO level metadata:

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

Figure 9.25.6a

### 9.25.7 Version of Learning Resource

The version number of the learning resource is designated with the tag lifeCycle.version.

The lifeCycle.version metadata element describes the edition of the learning resource. Initial version will be 1.0. Corrections will change minor version number; enhancements/updates will change major version number. This element upon initial final delivery to the Army must contain the value "1.0".

Example: 1.0 for the initial final delivery:

```
<lifeCycle>
  ...
  <version>
    <string>1.0</string>
  </version>
  ...
</lifeCycle>
```

Figure 9.25.7a

### 9.25.8 Status of Package Submittal

The submittal status of "final" for the learning resource is designated with the tag `lifeCycle.status` tag and shown in the following example:

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

Figure 9.25.8a

### 9.25.9 Proponent's Role

The role of the proponent as contributor is designated as "publisher" with the 'lifeCycle.contribute.role' tag and shown in the following example:

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

Figure 9.25.9a

### 9.25.10 Proponent's Name and Address

The 'lifeCycle.contribute.entity' metadata element identifies the organization (proponent) contributing to this resource according to the vCard specification located at [PDI Product Developers Information](#). This value must contain the full name, address, school code, and e-mail of the proponent institution according to the vCard specification and designated with the 'lifeCycle.contribute.entity' tag.

Example of the vCard format is as follows:

```
<lifeCycle>
  ...
  <contribute>
    <entity>BEGIN:VCARD
VERSION:2.1N:U.S. Army Infantry School
ORG:U.S. Army;Army Infantry School;Fort Benning
NOTE:071
ADR;DOM;WORK:Suite 650;6751 Constitution Loop;Fort Benning;GA;31905-
4502;U.S.
EMAIL;INTERNET:soldierinfo@benning.army.mil END:VCARD
    </entity>
  </contribute>
  ...
</lifeCycle>
```

Figure 9.25.10a

The 'NOTE' field has been designated as the method for identifying the school code. A listing of all school codes are contained in TRADOC Reg 350-70, Appendix C, section C-2.

### 9.25.11 Date of Submittal

The 'lifeCycle.contribute.date' metadata container element describes the date of the submission. This element must contain the date that the object was approved final using the SCORM designated format of YYYY-MM-DD.

The date of submittal is designated with the 'lifeCycle.contribute.date' tag and shown in the following example:

```
<lifeCycle>
  ...
  <contribute>
    <date>
      <datetime>2004-07-10</datetime>
    </date>
  </contribute>
</lifeCycle>
```

Figure 9.25.11a

### 9.25.12 Meta-Metadata Catalog and Entry Identifier

The 'metaMetadata.identifier' container element represents a mechanism for assigning a globally unique label that identifies the metadata record that describes the SCORM Content Model Component. This container element contains two sub-elements of <catalog> and <entry>.

<Catalog> represents the name or designator of the identification or cataloging scheme for the entry. <Entry> represents the value of the identifier within the identification or cataloging scheme that designates or identifies the metadata.

The Meta-Metadata catalog identifier is designated with the tag 'metaMetadata.identifier.catalog'. The Meta-Metadata entry identifier is designated with the tag 'metaMetadata.identifier.entry'.

The catalog element requires the value of "TBD" and the entry element requires the value of "TBD" as shown in the following example:

```
<metaMetadata>
  <identifier>
    <catalog>TBD</catalog>
    <entry>TBD</entry>
  </identifier>
  ...
</metaMetadata>
```

Figure 9.25.12a

### 9.25.13 Meta-Metadata Schema

The Meta-Metadata schema is designated with the element 'metaMetadata.metadataSchema'.

The 'metaMetadata.metadataSchema' metadata element identifies the name and version of the authoritative specification used to create this metadata instance. The Army requires three elements: (1) an element with a value of "LOMv1.0", (2) an element with the value of "SCORM\_CAM\_v1.3", and (3) an element with the value of "ADLv1.0".



**Note:** Requiring these three values will ensure that Army metadata will be valid under both SCORM 2004 2<sup>nd</sup> Edition and SCORM 2004 3<sup>rd</sup> Edition.

```
<metaMetadata>
  ...
  <metadataSchema>LOMv1.0</metadataSchema>
  <metadataSchema>SCORM_CAM_v1.3</metadataSchema>
  <metadataSchema>ADLv1.0</metadataSchema>
</metaMetadata>
```

Figure 9.25.13a

The metadata instances for all of the SCORM metadata must conform to both the conformance requirements of the LOM and the SCORM CAM.

#### 9.25.14 Language of the Metadata File

The language (English, Spanish, German, etc.) of metadata file is designated as 'metaMetadata.language' to indicate the language of all <string> elements within the metadata file. If this value is provided, then it is not necessary to indicate a language attribute for the <string> elements.

The 'metaMetadata.language' element identifies the language contained within the metadata tags of the XML file. Use the same language codes as the general.language element.

```
<metaMetadata>
  ...
  <language>en</language>
</metaMetadata>
```

Figure 9.25.14a

#### 9.25.15 File Formats

The 'technical.format' metadata element identifies all the technical data types of this resource by using the Multipurpose Internet Mail Extensions (MIME). This element must contain all MIME types that are used in the learning resource being described.

MIME types are used to identify the software needed to access the resource. Browsers use MIMEs in the same way they use legends. If a browser received some content with a unique file extension, it looks in its list of MIME types to help it identify the content. The syntax of MIME types is as follows: type + "/" + subtype. These types are then associated with a file extension. For more details on MIME types, go to:

- [MHONARC.org](http://MHONARC.org)
- [IANA.org](http://IANA.org) for an official list maintained by the Internet Assigned Numbers Authority (IANA).

Following is an example of 'technical.format':

```
<technical>
...
  <format>text/html</format>
  <format>text/plain</format>
  <format>image/gif</format>
  <format>image/jpeg</format>
  <format>application/x-shockwave-flash</format>
...
</technical>
```

Figure 9.25.15a

### 9.25.16 Cost of Learning Resource

The 'rights.cost' metadata container element describes whether use of the resource requires payment. This element must have the <source> element equal to "LOMv1.0" and the <value> element equal to "no".

Following is an example of rights.cost:

```
<rights>
  <cost>
    <source>>LOMv1.0</source>
    <value>no</value>
  </cost>
...
</rights>
```

Figure 9.25.16a

### 9.25.17 Copyright and Other Restrictions

The 'rights.copyrightAndOtherRestrictions' metadata element indicates whether copyright or other restrictions apply to the use of this resource. This value must be the <source> element equal to "LOMv1.0" and the <value> element equal to "no".

Following is an example of 'rights.copyrightAndOtherRestrictions':

```

<rights>
  ...
  <copyrightAndOtherRestrictions>
    <source>>LOMv1.0</source>
    <value>>no</value>
  </copyrightAndOtherRestrictions>
</rights>

```

Figure 9.25.17a

### 9.25.18 Classification

The classification container element describes where this resource is placed within a particular classification system. To define multiple classifications, there may be multiple instances of this category. This element is required for Content Organization and SCO metadata. This classification container element must have three (3) sub-elements per classification instance, namely, <purpose>, <description>, and <keyword>. Include one instance each for MOS & Skill Level, Tasks, Learning Objectives, Accessibility Restrictions, Security Level, and Collection. If courseware is classified by a SQI or ASI or other classification, then one instance each must also be provided.

The 'classification.purpose' best practice vocabulary relates to Army terms as follows:

Best Practice Vocabulary	In Army Terms
Discipline	MOS & Skill Level
Discipline	SQI
Discipline	ASI
Educational objective	Task Numbers and Task Descriptions
Educational objective	Learning Objectives (Action, Condition and Standard)
accessibility restrictions	508 compliant or not
security level	Foreign Disclosure
collection	DOD

Figure 9.25.18a

For an MOS and Skill Level, enter the MOS and Skill Level and textual description of the MOS for which the object was designed. Following is an example of this classification instance:

```

<classification>
  <purpose>
    <source>>LOMv1.0</source>
    <value>discipline</value>
  </purpose>
  <description><string language="en-US">11C2 Indirect Fire Infantryman</string>
  </description>
  <keyword><string language="en-US">Indirect Fire Infantryman</string></keyword>
</classification>

```

Figure 9.25.18b

For an SQI, enter the SQI and textual description for which the object was designed. Following is an example of this classification instance:

```
<classification>
  <purpose>
    <source>LOMv1.0</source>
    <value>discipline</value>
  </purpose>
  <description><string language="en-US">E Mountaineer</string>
  </description>
  <keyword><string language="en-US">Mountaineer</string></keyword>
</classification>
```

Figure 9.25.18c

For an ASI, enter the ASI and textual description for which the object was designed. Following is an example of this classification instance:

```
<classification>
  <purpose>
    <source>LOMv1.0</source>
    <value>discipline</value>
  </purpose>
  <description><string language="en-US">Q6 Long Range Surveillance
Leader</string>
  </description>
  <keyword><string language="en-US">Long Range Surveillance
Leader</string></keyword>
</classification>
```

Figure 9.25.18d

For Tasks, enter the list of task numbers and titles of the critical tasks for which this SCO/Organization provides training or support. Following is an example of this classification instance:

```
<classification>
  <purpose>
    <source>LOMv1.0</source>
    <value>educational objective</value>
  </purpose>
  <description><string language="en-US">071-312-3003 Lay An M60 Machine gun
Using Field Expedients; 071-312-3007 Prepare A Range Card For An M60 Machine
gun; 071-312-3025 Main An M60 Machine Gun</string>
  </description>
  <keyword><string language="en-US">Range Card </string></keyword>
</classification>
```

Figure 9.25.18e

For Learning Objectives, enter the Action, Condition, and Standard for which this SCO/Organization provides training or support. Following is an example of this classification instance:

```
<classification>
  <purpose>
    <source><langstring xml:lang="x-none">LOMv1.0</langstring></source>
    <value><langstring xml:lang="x-none">educational
objective</langstring></value>
  </purpose>
  <description><string language="en-US">Action: Lay An M60 Machine gun Using
Field Expendients; Condition: Given Interactive Multimedia Instruction;
Standard: The Standards are met when the learner has completed the IMI lesson
and achieved a passing score on a separately administered test.</string>
  </description>
  <keyword><string language="en-US">M60 Machine gun using Field
Expendients</string></keyword>
</classification>
```

Figure 9.25.18f

For accessibility, enter whether or not this object is 508 compliant. The values for <description> are "508 Compliant" or "Not 508 Compliant" and the values for <keyword> are "508", "PL508", or "accessibility restrictions". Following is an example of a classification instance:

```
<classification>
  <purpose>
    <source>LOMv1.0</source>
    <value>accessibility restrictions</value>
  </purpose>
  <description><string language="en-US">Not 508 Compliant</string>
  </description>
  <keyword><string language="en-US">Not 508</string></keyword>
</classification>
```

Figure 9.25.18g

For Foreign Disclosure, the following is an example of a classification instance:

```
<classification>
  <purpose>
    <source>LOMv1.0</source>
    <value>security level</value>
  </purpose>
  <description><string language="en-US">FD1</string>
  </description>
  <keyword><string language="en-US">FD1</string></keyword>
</classification>
```

Figure 9.25.18h

For the ADL-R, the following is an example of a minimum classification instance:

```
<classification>
  <purpose>
    <source>ADL-Rv1.0</source>
    <value>collection</value>
  </purpose>
  <taxonPath>
    <source>
      <string>ADL/DOD Content Category Taxonomy</string>
    </source>
    <taxon>
      <entry>
        <string>DOD</string>
      </entry>
    </taxon>
  </taxonPath>
</classification>
```

Figure 9.25.18i

## 9.26 Setting up the SCORM Metadata Schemas

The metadata schemas are required to be placed in the root folder of the content package, according to the SCORM specification pertaining to all SCORM schemas.

Following is a representation of how the metadata schemas are to be structured in the courseware package root folder:

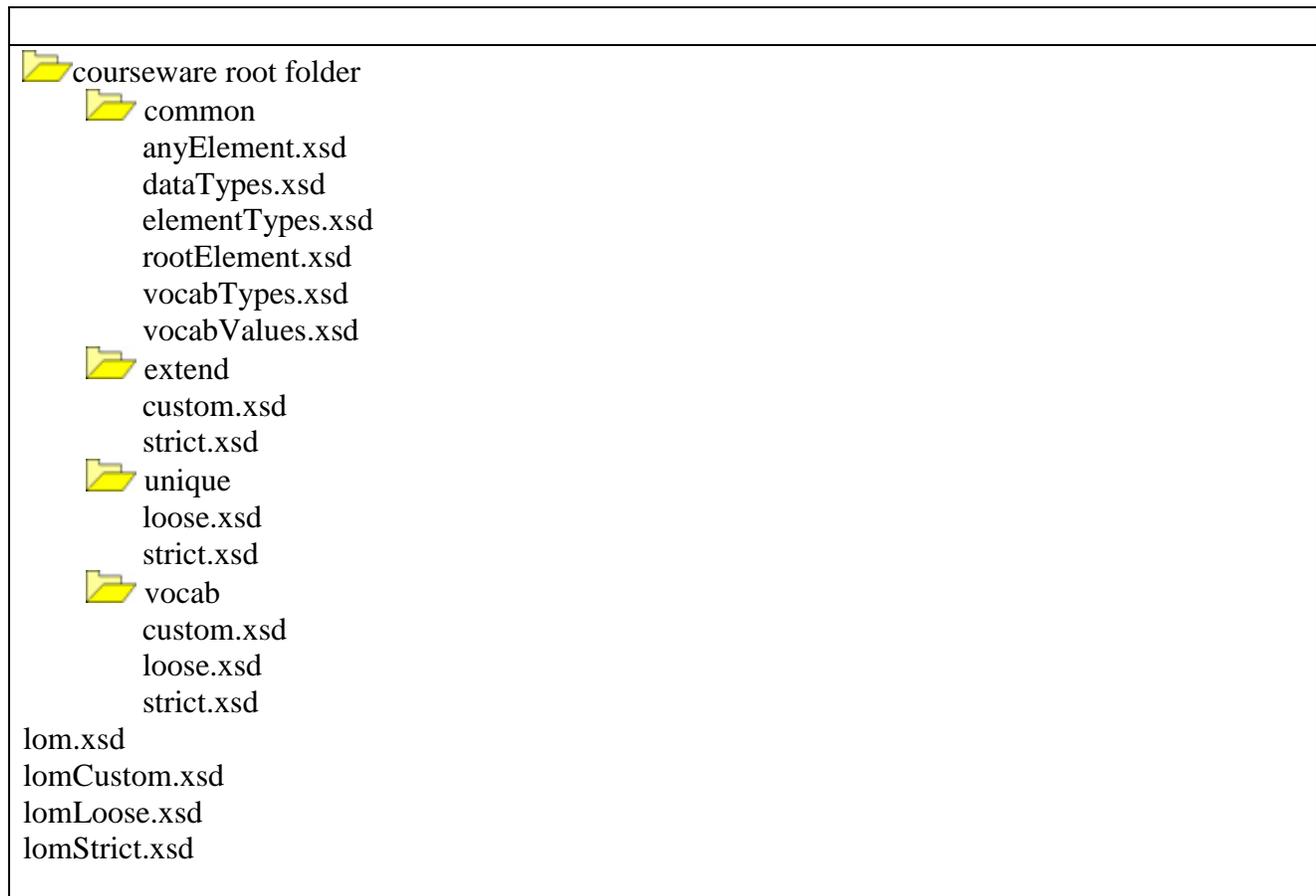


Figure 9.26a



**9.26-1 Business Rule (Army): The metadata schemas that must reside in the root folder must also be located in the folder(s) where the metadata resides. If a folder contains metadata, then the metadata schemas must be copied into this folder.**

Business Rule 9.26-1 exists as a safeguard. Certain XML processors will try to locate the XML schema documents for the metadata by scanning the folder where the metadata resides. This action may cause some errors with XML parsing if the schema files are not present.

The schemas for all three LOM Profiles can be downloaded from:

<http://ltsc.ieee.org/xsd/lomv1.0/20040413/>.

## 9.27 Packaging and Delivery

SCORM indicates that the purpose of Content Packaging is to provide a standardized way to exchange digital learning resources between different systems or tools. The Content Aggregation Package contains a manifest file describing the entire package and all physical files, both of which

should be compressed into a Package Interchange File (PIF) in accordance with the applicable delivery order.



**Note:** SCORM 2004 3<sup>rd</sup> Edition (or 4<sup>th</sup> Edition either) does not currently recommend use of (sub)manifests due to confusion on the cases for using (sub)manifests, requirements on syntax, and behavior processing. All previously defined requirements for (sub)manifests have been removed from SCORM documentation.



**9.27-1 Business Rule (Army):** All SCOs (content and graded assessments) for a particular unit of instruction (any hierarchical level) must be contained in one content package. This SCORM package must contain a manifest (imsmanifest.xml) file and all of the SCORM and extension schemas in the root of the package. All physical files required for the courseware must be referenced locally and contained within the content package and disclosed on the imsmanifest.xml file. Metadata is supplied as separate files and these files must be contained within the content package and disclosed on the manifest file.

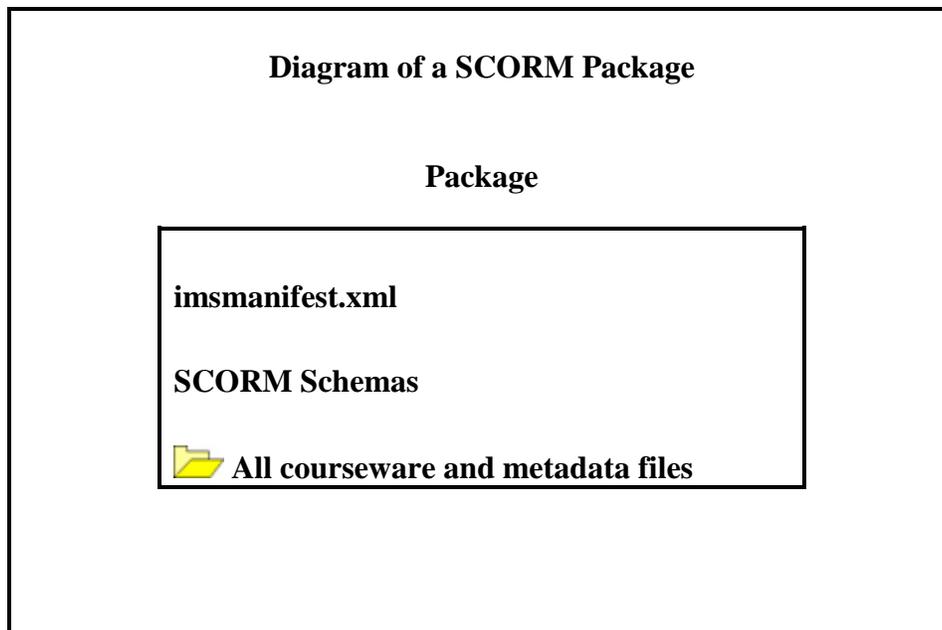


Figure 9.27a

Under SCORM 2004, the LMS does not require separating out assessments from the content. Instructional strategies are now enforced by the sequencing component of SCORM.

Note: There are currently no known package size constraints in the ALMS (Saba 2005). However, anything over 500 MB will have to be loaded by an administrator. This is due to bandwidth issues of loading content remotely, not actual LMS constraints. Content over 1 GB has been successfully loaded. All uploading of content to the production LMS will be done by an administrator of the LMS. Courseware managers may only upload content to the Common Test Environment (CTE).

### 9.27.1 Creating the Manifest File

The manifest file is the most important file in the package. Specific data on the manifest file is read into the Learning Management System (LMS). The LMS uses this file to determine the table of contents to display and launch file for each SCOs. Content repositories use this file in aggregating and de-aggregating packages. Metadata is determined and accessed by using the information in this file.

The activity tree is a representation of the table of contents, which is XML-tagged within the manifest file, and is created in the LMS when the SCORM package is launched.

For examples of coding a manifest, refer to the Programming Examples, [XML Examples on the Manifest](#) section (9.1).

#### Two Types of Manifest Files

There are two types of manifest files: Content Aggregation Manifest file and Resource Package Manifest file. A Content Aggregation Manifest contains an <organizations> section, which describes a distinct course structure such as a table of contents. A Resource Package does not contain an <organizations> section and, therefore, no table of contents, and only provides a method of transferring learning resources that are not within a learning context.

The Army does not anticipate receiving Resource Packages as courseware deliverables. However, resource packages can be used for the submission of raw media files or other files.

The diagram following shows the differences between the two different types of manifest files:

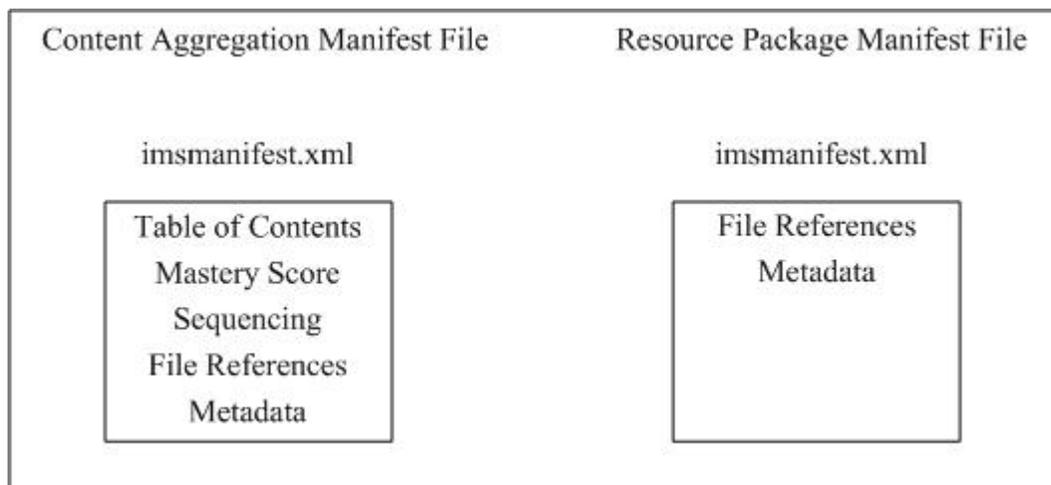


Figure 9.27.1.1a

## 9.27.2 How to Create a Content Package per the SCORM Implementation Guide (<http://www.adlnet.gov>)

1. Determine whether you want to package the files as a PIF. For a course delivered via the Web, package files as a PIF. For the CD-ROM version of the course, a non-PIF file structure may be used.
  - PIF – Package Interchange File is a representation of the content package components using the PKZIP Version 2.04g archive format (zip). The PIF provides a concise Web delivery format that can be used to transport content packages between systems (zipped).
  - Non-PIF – File structure to be used on a CD-ROM or other file system.
2. Place the imsmanifest.xml file at the root level of the package.
3. Place all schemas referenced by the manifest at the root level of the package. These metadata schemas must also be located in the folder(s) where the metadata resides. If a folder contains metadata, then the metadata schemas must be copied into this folder.
4. Place all physical files needed by the packaged courses, lessons, etc. where you refer to them in your manifest.
5. If using a PIF, compress and save using the PKZIP Version 2.04g standard.
6. Test the newly created package using the Content Package test of the Conformance Test Suite.
7. If the Content Package passes the conformance test, import it into the LMS for use.



**Best Practice:** It is recommended that the Resource Validator test be run using the newly created PIF (that has been exploded) to ensure the PIF accurately reflects the courseware.

To create a PIF, everything in the large oval shape on the RIGHT side of the following figure would be zipped up using a Microsoft® Windows interface to PKZIP such as WinZip. Notice that the imsmanifest.xml file will be in the root of the package.

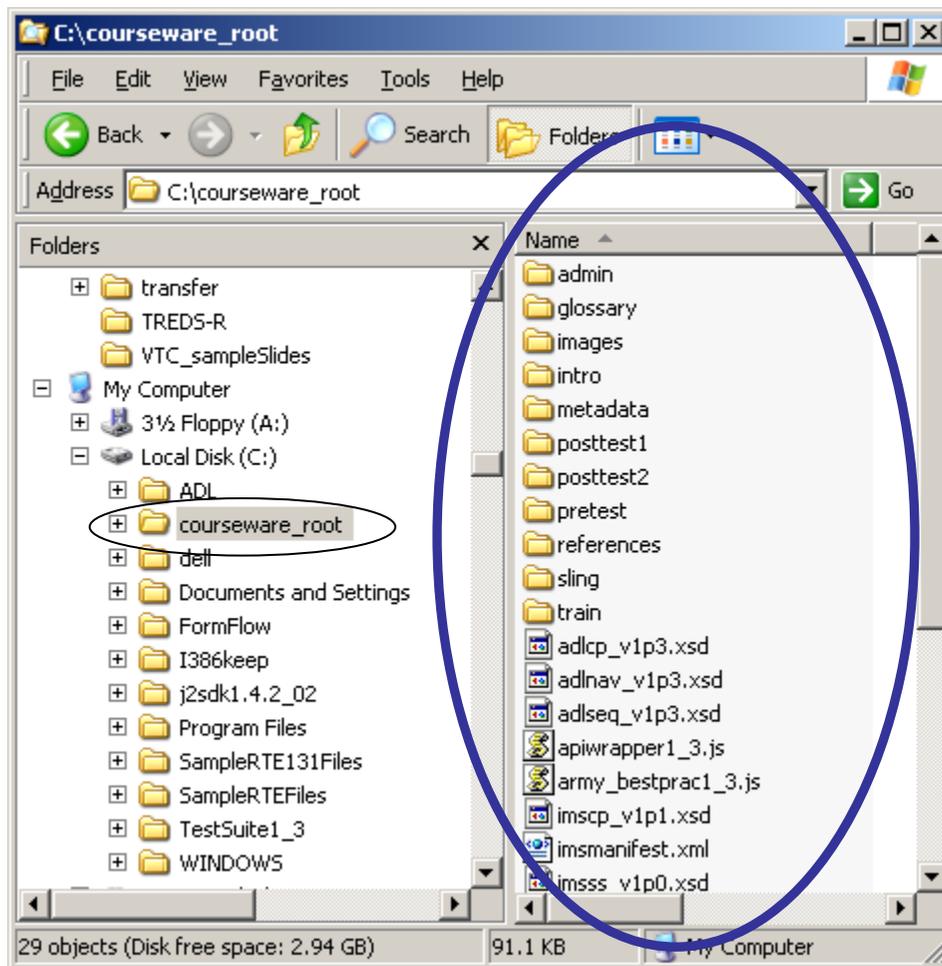


Figure 9.27.2a

WinZip instructions for creating a PIF and other helpful information can be found at <http://www.atsc.army.mil/itsd/imi/documents/PIF.doc>.

 **9.27.2-1 Business Rule (Army): SCORM 2004 compliant courseware must be delivered to the Government as a PIF file. This PIF file will be in the form of a compressed (.zip) file using the PKZIP Version 2.04g standards that are conformant to RFC1951.. Other contract deliverables (for example, original source files, SCORM testing log files, answer keys, loading instructions, ALO XML file, etc.) must be delivered as separate files external to the courseware PIF file(s) and I/A/W ATSC Acceptance Criteria.**

It is important that **only** the courseware files, metadata files, imsmanifest.xml file, and the schema files (examples shown in Figure 9.27.2a) be included in the PIF. Other files or other contract

deliverables should not be included in the PIF. Wire-frames should also be delivered in a separate PIF.

The Government expects a logical folder structure and separate CD-ROMs to easily identify contract deliverables. Refer to the most current delivery order template for packaging specific contract deliverables.

### **9.27.3 Final Courseware Packaging and Delivery Requirements**

The current guidance on final courseware delivery for Web-based courseware, stated in the delivery order, will create duplication of lessons. The product will likely be delivered/fielded incrementally as individual lessons are completed and pass the testing process. It will also be submitted (with corrections) as a final deliverable in three distinct versions: as part of the final package for the entire course (with sequencing); as one package for each lesson with pretests and posttests (with sequencing); and as one package for each lesson (without sequencing) and test inclusion at the proponent's discretion.

LMS capabilities may influence packaging and delivery as future discoveries are made and lessons learned are applied. For example, an LMS may have a feature that allows the ability to identify a test. If this is the case, perhaps only one delivery of a test would be sufficient for more than one packaging scenario. Based on whether a learner is taking the test for credit, an LMS may have the capability of serving up/not serving up the test.

## 10. Test Item Data (Interactions) Table

Following is a test item data table for the developer to fill out for the programmer:

Is this a timed assessment?

<input type="checkbox"/>	Yes
<input type="checkbox"/>	No

Question	Question Type	Correct Responses	Weight (if applicable)
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

Following is a populated example test item data table:

Is this a timed assessment?

	Yes
X	No

Question	Question Type	Correct Response(s)	Weight (if applicable)
1	true/false	true	.25
2	multiple choice	A, C	.50
3	true/false	false	.25
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

## 11. Metadata Tables

Following is a metadata table for the training developer to fill out for the programmer:

<b>Metadata Required</b>	<b>Metadata Tag Information for Programmer (from Developer)</b>
Catalog Identifier	ATIA
Entry Identifier	TBD
Title of Learning Resource	
Language of Learning Resource	
Description of Learning Resource	
Keywords	
Type of Metadata	Choose one: SCO (2) or Content Organization (3)
Version of Learning Resource	
Status of Package Submittal	
Proponent's Role	
Proponent Name School Code Address E-mail	
Date of Submittal	
Metadata Catalog Identifier	TBD
Metadata Entry Identifier	TBD
Metadata Specification Used to Create this Metadata	LOMv1.0 SCORM_CAM_1.3.1 ADLv1.0
Language of Metadata file	
File Format (MIME types) (That is.html, .jpg, .gif, .swf, .avi, .mpeg, etc.)	
Cost of Learning Resource	
Copyright and Other Restrictions	
MOS and Skill Level	
SQI	
ASI	
Task Numbers and Task Descriptions	
Learning Objectives (Action, Condition, Standard)	
508 Compliant	
Security Level (Foreign disclosure)	

Following is a populated example of a metadata table:

<b>Metadata Required</b>	<b>Metadata Tag Information for Programmer (from Developer)</b>
Catalog Identifier	ATIA
Entry Identifier	TBD
Title of Learning Resource	Introduce Basic M60 Machine Gun Training
Language of Learning Resource	English
Description of Learning Resource	Basic instruction on U.S. Army Infantry M60 Machine Gun laying using field expedients, range card preparation, maintenance, function check performance, loading, unloading, malfunction corrections, and target engaging
Keywords	Infantry; M60 laying using field expedients; Notched Stakes
Type of Metadata	2
Version of Learning Resource	1.0 (for final delivery)
Status of Package Submittal	final
Proponent's Role	publisher
Proponent Name School Code Address	U.S. Army Infantry School 071 6751 Constitution Loop, Suite 650 Fort Benning, GA 31905-4502
E-mail	soldierinfo@benning.army.mil
Date of Submittal	2004-07-10
Metadata Catalog Identifier	TBD
Metadata Entry Identifier	TBD
Metadata Specification Used to Create this Metadata	LOMv1.0 SCORM_CAM_1.3.1 ADLv1.0
Language of Metadata file	English
File Format (MIME types) (That is.html, .jpg, .gif, .swf, .avi, .mpeg, etc.)	Text: .html Image: .gif, .jpeg Applications: Shockwave, Flash
Cost of Learning Resource	no
Copyright and Other Restrictions	no
MOS and Skill Level	11C2 Indirect Fire Infantryman
SQI	E Mountaineer
ASI	Q6 Long Range Surveillance Leader
Task Numbers and Task Descriptions	071-312-3003 Lay An M60 Machine Gun Using Field Expedients 071-312-3007 Prepare A Range Card For An M60 Machine Gun 071-312-3025 Man An M60 Machine Gun

Metadata Required	Metadata Tag Information for Programmer (from Developer)
Learning Objectives (Action, Condition, Standard)	Action: Lay An M60 Machine Gun Using Field Expedients Condition: Given Interactive Multimedia Instruction Standard: The Standards are met when the learner has completed the IMI lesson and achieved a passing score on a separately administered test.
508 Compliant	Not 508 Compliant
Security Level (Foreign disclosure)	FD1