

SCORM 1.2 SUPPORT GUIDE
for
The NAVY Integrated Learning Environment



ILE SCORM 1.2 SUPPORT GUIDE



TABLE OF CONTENTS

INTRODUCTION..... 4

RUN-TIME ENVIRONMENT 4

LMS-SPECIFIC IMPLEMENTATION 5

CONTENT PACKAGING 6

META-DATA 6

ROLL UP & SCORING 7

ASSESSMENT ROLL UP SCENARIOS 12

PRETEST SEQUENCING 18





Introduction

This document describes the recommended guidance that must be provided for Navy developers creating learning content in alignment with the SCORM 1.2 specification. The technical details for support of the SCORM 1.2 specification by the Navy’s LMS (Learning Management System) is addressed in terms of Run-Time Environment, Content Packaging, and Meta Data. This document also explains how an aggregation of multiple SCOs can be rolled up and configured by the LMS using the intended scoring logic and lesson status provided by any number of SCOs. Finally, scenarios of aggregate scoring examples will be provided.

Run-Time Environment

The Navy’s LMS is SCORM Version 1.2 Run-time Environment Conformant (LMS-RTE3). The three aspects of the Run-Time Environment are Launch, Application Program Interface (API), and the Data Model. The Launch and API are clearly defined in the SCORM 1.2 specification. The optional implementation for some aspects of the data model was not clearly defined in the SCORM 1.2 specification. The implementation for course credit and lesson mode is LMS-specific. The table below (Figure A) details exactly which SCORM 1.2 data model elements were implemented in the Navy LMS. **Y = “Yes”**; **N = “No”**

Figure A: SCORM 1.2 DATA MODEL ELEMENTS

Mandatory Elements:	Implemented
cmi.core._children	Y
cmi.core.student_id	Y
cmi.core.student_name	Y
cmi.core.lesson_location	Y
cmi.core.credit	Y
cmi.core.lesson_status	Y
cmi.core.entry	Y
cmi.core.score._children	Y
cmi.core.score.raw	Y
cmi.core.total_time	Y
cmi.core.exit	Y
cmi.core.session_time	Y
cmi.suspend_data	Y
cmi.launch_data	Y
Optional Elements:	Implemented
cmi.core.score.max	Y
cmi.core.score.min	Y
cmi.core.lesson_mode	Y
cmi.comments	Y
cmi.comments_from_lms	Y
cmi.objectives._children	Y
cmi.objectives._count	Y
cmi.objectives.n.id	Y
cmi.objectives.n.score._children	Y
cmi.objectives.n.score.raw	Y



cmi.objectives.n.score.max	Y
cmi.objectives.n.score.min	Y
cmi.objectives.n.status	Y
cmi.student_data._children	Y
cmi.student_data.mastery_score	Y
cmi.student_data.max_time_allowed	Y
cmi.student_data.time_limit_action	Y
cmi.student_preference._children	Y
cmi.preference.audio	Y
cmi.student_preference.language	Y
cmi.student_preference.speed	Y
cmi.student_preference.text	Y
cmi.interactions._children	Y
cmi.interactions._count	Y
cmi.interactions.n.id	Y
cmi.interactions.n.objectives._count	Y
cmi.interactions.n.objectives.n.id	Y
cmi.interactions.n.time	Y
cmi.interactions.n.type	Y
cmi.interactions.n.correct_responses._count	Y
cmi.interactions.n.correct_responses.n.pattern	Y
cmi.interactions.n.weighting	Y
cmi.interactions.n.student_response	Y
cmi.interactions.n.result	Y
cmi.interactions.n.latency	Y

LMS-specific Implementation

The Navy’s LMS is SCORM Version 1.2 Run-time Environment Conformant (LMS-RTE3). Many aspects of the SCORM 1.2 specification were identified as LMS-specific. The following sections will clarify how the Navy’s LMS implemented various aspects of the SCORM 1.2 specification.

COURSE CREDIT

Each course on the LMS is credited to the student. As of SCORM 1.2, there was not a defined way to signify that the learning content can be taken for credit or no-credit. Implementation is LMS-specific. The default vocabulary value for each SCORM 1.2 course is “credit”.

LESSON MODE

As of SCORM 1.2 there was not a defined way to signify that learning content can be taken in different modes. Implementation is LMS-specific. Therefore, the default vocabulary value for each course is “normal”. However, this is internally controlled by the LMS and the cmi.core.lesson_mode value does not change to anything other than “normal.” Each completed course can be launched in a simulated “review” mode after it has become a complete transcript. However, this is controlled on a course-by-course basis through the LMS, and not by control of any one SCO. For more information on how the LMS treats each attempt on a SCO, please read the Rollup & Scoring section.



LESSON STATUS

Content can set the following values for `cmi.core.lesson_status`:

- Completed: the student experienced all of the elements in the SCO. No associated raw score should be sent.
- Incomplete: the SCO was begun but not finished.
- Passed: necessary score was achieved. Student is considered to have completed the SCO and passed. This should only be set for those SCOs associated with sending a passing raw score.
- Failed: the SCO was not passed. Student is considered to have completed the SCO and failed. This should only be set for those SCOs associated with sending a failing raw score.

BEST ATTEMPT

The LMS keeps track of multiple attempts on a SCO. This means that when a SCO is completed (with a status of "completed", "passed" or "failed"), the LMS will record a new attempt if any completed SCO is relaunched during the current session of activity (before exiting and returning to the LMS catalog). The status and score from the best attempt are always displayed in the course's table of contents, and the best attempts are also used for the final rollout logic.

SCO EXITING

The Navy E-Learning catalog hosts several offerings from developed by many different sources. Each SCORM 1.2 course is typically designed with a unique custom navigation and interface. The Navy's LMS forces each SCO to be launched within a frameset window. The top frame of the frameset hosts the API adapter. In order to close this frameset, each SCO must execute the `top.window.close()` JavaScript method. Furthermore, it is a SCORM 1.2 best practice to utilize the `LMSCommit()` function in the event of an ungraceful exit (e.g. the browser window is closed by the learner instead of using the internal exit button within the course interface).

Content Packaging

A content package must be able to stand-alone; that is, it must contain all the information needed to use the contents for learning when it has been unpacked. Content packages are not required to be incorporated into a Package Interchange File. A package may also be distributed on a CD-ROM or other removable media without being compressed into a single file. This section defines the conformance requirements for SCORM Content Packages. There are currently two defined Content Packaging Application Profiles as defined in Section 2.3 of the SCORM Content Aggregation Model:

1. Content Aggregation Package – a packaged containing a collection of organized (content structure) learning resources.
2. Resource Package – a package containing a collection of learning resources.

Navy developers must provide the learning resources within a Content Aggregation Package or as a separate Resource Package. The SCORM 1.2 Resource Package Profile defines a mechanism for packaging learning resources (e.g. Assets and SCOs) without having to provide a specific organization, learning context, or curricular taxonomy. Packaging learning resources will provide a common means for populating the LCMS repository.

Meta-Data

Metadata should only reference one item, so if you have multiple versions or revisions to an item, each iteration should have its own unique metadata. You can accomplish this by using different unique identifiers or the version field in the Life Cycle element. The primary purpose of metadata is as a tool to enable those seeking assets and content relevant to their requirements to locate them efficiently and effectively. In most instances, the instructional designer or developer should



define the metadata fields for the programmer in advance. The details of Meta-data are provided in the SCORM. In general, guidance is provided for meta-data to be applied to Content Aggregations, SCOs, and Assets in a consistent manner so that they can be searched upon within a repository or shared across other distributed systems to facilitate reuse. Navy content developers are required to provide all of the mandatory elements for Content Aggregation, SCO, and Asset Meta-Data as part of the final deliverable for a SCORM 1.2 conformant product. Content Aggregation, SCO, and Asset Meta-Data must be aligned with the IEEE LTSC LOM standard.

Roll up & Scoring

The process of determining the tracking status of a parent activity based on the tracking status of the child is referred to in the IMS simple sequencing specification as "roll up". This section provides the definition of a roll up scoring weight value that determines if and how a particular activity's score is weighted relative to its sibling activities' scores in determining the score of the parent activity. The Normalized Score for a course follows these rules:

1. If the values of the normalized score for all of the child activities that participate in the rollup are "null", the value of the normalized score for the course is "null".
2. Otherwise, the value of the normalized score for the course (parent activity) is the weighted average of the values of the normalized score for the child activities that participate in the rollup, normalized to a range of 0 to 100.
3. When the student is in their first attempt at a SCO the `cmi.core.score.raw` value is set to "" (empty string) or "NULL". Additionally, a value of "NULL" indicates that the score cannot be ascertained or has not been set in `cmi.core.score.raw`.
4. If the content developer does not want a SCO included in the final calculation of the final score for the course then no value should be set or a value of "" (empty string) must be set as the `cmi.core.score.raw` value for the SCO.

Example:

Here is an example of how the LMS could roll up a course score with the following 7 SCO scores:

```
rt_ch_grade
SCO 1 = "NULL"
SCO 2 = "NULL"
SCO 3 = "NULL"
SCO 4 = "NULL"
SCO 5 = "NULL"
SCO 6 = "NULL"
SCO 7 = "80"
```

The score is rolled up to $80 / 1 = 80$. Since the LMS ignores NULL scores, the score would roll up to 80.

RAW SCORE

The raw score for a SCO may be internally calculated in any manner that makes sense to the developer, and represents a final, normalized score for the SCO. For instance, it could reflect the percentage of objectives complete, it could be the raw score on a multiple-choice test, or it could indicate the number of correct first responses to embedded questions in a SCO. The LMS initializes the raw score to "" or "NULL". If the content developer does not want a SCO included in the final calculation of the final score for the course then the raw score for a SCO should not be set. The `cmi.core.score.raw` can be a normalized value between 0 and 100. However, the content developer should not set "0" as a score unless it is an earned score to be averaged with other SCOs for a final grade in the course.



Example Return/Set Values:

SCO 1 = "NULL"

SCO 2 = "80"

SCO 3 = "90"

SCO 4 = "0"

Scenario: The first SCO is an introduction to the course and not send a score. SCO 2, SCO 3, and SCO 4 are actual lessons with embedded tests at the end of each lesson of instruction. The student scores "80" on SCO 1, "90" on SCO 2, and misses every question on SCO 3 for a score of "0". Given the course were configured to ignore NULL scores, the course score would roll up to 56.6.

ROLLUP CONFIGURATION (ASSESSMENT SCORE LOGIC)

The LMS supports the configuration of scoring logic at the *course level*. The configuration of scoring logic for a course with multiple *launchable objects* (SCOs, AUs) can be used in conjunction with the course *roll up configuration*. The score logic is based on the following options:

1. **Score Provided by Course** – This option should be selected, when a course will return the overall course score. This is often the case when a course author creates a single launchable object (SCO, AU) course. In this case, the single object is the parent for all other objects that make up the course and the overall course score should reflect the value returned by the single launchable object (SCO, AU). Moreover, this option should be selected for a launchable object (SCO, AU) course that has no scoring capability.
2. **Average Score of All Units** – The final course score is calculated by averaging the total scores returned over the total number of launchable objects (SCOs, AUs). This option assumes that each launchable object (SCOs, AUs) sends a score and that all of the scores should be averaged.
3. **Average Score of All Units With Scores** – This option will calculate the final course score by averaging the total scores returned by the number of launchable objects (SCOs, AUs) that returned a score. Though this option is likely the most common for multiple launchable objects (SCOs, AUs) that have more than one assessment, it should be noted that the only draw back is that students can increase their average score just by completing the launchable objects (SCOs, AUs) that they know count for a score.
4. **Fixed Average** – This option provides the LMS administrator to indicate a fixed number that total of returned scores should be divided by. If the LMS Administrator knows the number of launchable objects (SCOs, AUs) that will return a score, then by recording it, the average score will be most accurate.



The following image (Figure A; below) demonstrates an appropriate layout for a user to configure the appropriate scoring logic at the course level.

Figure A

The screenshot shows a web-based configuration interface for ILE SCORM 1.2. At the top, there are two dropdown menus labeled 'Actions' and 'Subregions', each with a 'Go' button. Below these is a toolbar with icons for file operations (new, copy, delete, info, undo, redo, help). A tabbed interface is visible with tabs for 'Main', 'Description', 'E-Learning' (selected), 'Compliance', 'Event Defaults', 'Lending Library', and 'Version'.

The 'E-Learning' tab contains the following settings:

- E-Learning meta-data available**: (with a lock icon)
- Launchable**:
- Document Launch URL**:
- Special Instructions**:
- Days to Finish CBT**: (# of days)
- CBT Launch Interface Type**:
- Virtual Classroom Type**:
- Scoring Logic**:
- Number Of Scoring Objects**:
 -
 -
 -
- Completion Threshold**:
- Completion Status Logic**:

At the bottom of the interface, there is a 'Data entry mode' checkbox (unchecked) and two buttons: 'Save' and 'Cancel'.



ROLLUP CONFIGURATION (AGGREGATE LOGIC)

Just like the Course Score Rollup (Aggregate) Logic, there may be different methods to determine if a student has completed a course or not. The LMS will need to provide configuration options at the course level in order for the LMS to determine the appropriate Completion Status to set on a Course Transcript. As indicated above, current e-learning standards treat each Learning Object independently and have not considered the logic needed when multiple objects are placed together to form a course. The following options will be used by the LMS in order to determine the completion status to set for a transcript.

1. **Status Provided by Course** – When selected, the LMS will always prompt the user upon exiting the course in order to determine if he has completed the course or not. Based on the users response, the LMS will set the appropriate History Transcript Status.
2. **Complete when all units complete** – When selected, the LMS will only record a completed transcript status when the learner has completed all objects that make up the course.
3. **Complete when threshold score met** – When selected, the LMS will compare the overall Course Score to the Course Mastery Score. If the Course Score is greater than the Course Mastery Score, than the history transcript status will be updated to Completed, otherwise it will indicate Not Completed. When this option is selected, the LMS must check to ensure a Mastery Score has been recorded for the course.
4. **Status Provided by Student** – When selected, the LMS will expect an overall course completion status to be returned from the course. If this option is selected and the course does not return an overall course completion status, then the LMS will Prompt the Learner and default to option 1.

COMPLETION THRESHHOLD SCORE

Also included in the above figure is Course Completion Threshold Score. This value lets the course administrator set a Completion Threshold Score for the entire course. Currently, the e-learning standards (SCORM, AICC) have a concept of Completion Threshold Score; however, this value applies to each object that makes up a course. It is important to keep in mind that the SCORM model treats each SCO as an individual object. There is no dependencies or interaction between these objects. Moreover, when these objects are placed together to form a course, there is no required or defined mechanism overall course Completion Threshold score. Within this request, the Course Completion Threshold Score will be used in to determine Course Completion Status.



The following image demonstrates an appropriate layout for a user to configure the appropriate Completion Status logic at the course level.

Figure B

The screenshot shows a web-based configuration interface for ILE SCORM 1.2. At the top, there are navigation elements: 'Actions' and 'Subregions' dropdown menus, each with a 'Go' button. Below this is a toolbar with icons for file operations and help. A tabbed interface is visible, with the 'E-Learning' tab selected. The main content area contains several configuration options:

- E-Learning meta-data available**: (with a lock icon)
- Launchable**:
- Document Launch URL**:
- Special Instructions**:
- Days to Finish CBT**: (# of days)
- CBT Launch Interface Type**:
- Virtual Classroom Type**:
- Scoring Logic**:
- Number Of Scoring Objects**:
- Completion Threshold**:
- Completion Status Logic**:

A dropdown menu is open for the 'Completion Status Logic' field, showing the following options:

- Status provided by course
- Complete when all units complete
- Complete when threshold score met** (highlighted)
- Status provided by student

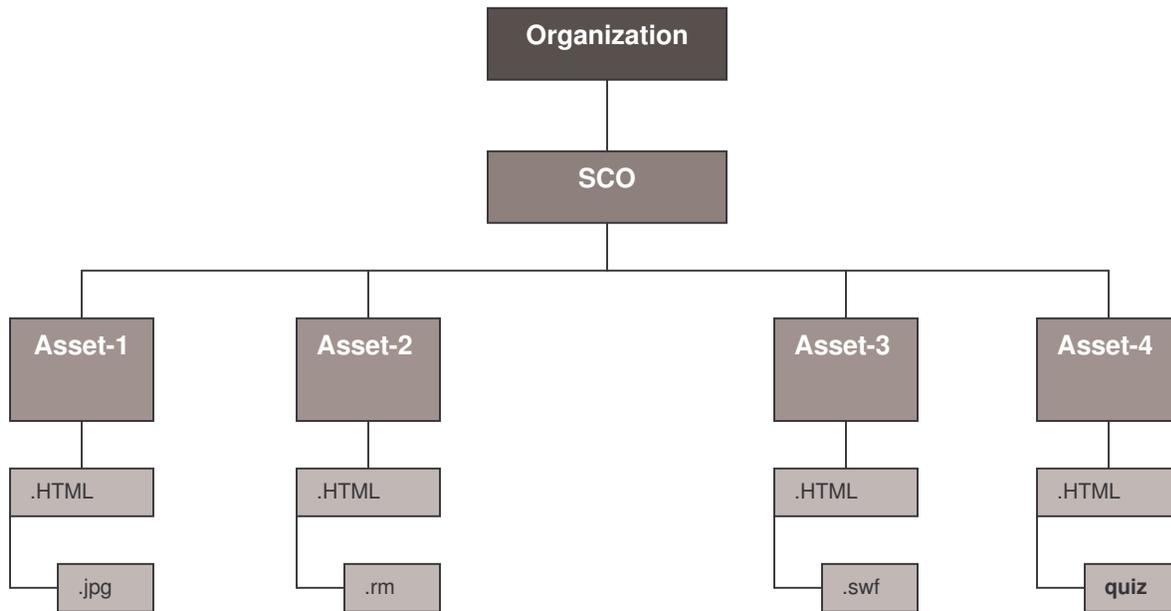
At the bottom left, there is a 'Data entry mode' checkbox which is currently unchecked.



ASSESSMENT ROLL UP Scenarios

Scenario 1: Single SCO Course, One Assessment

This scenario represents a SCO composed of multiple pages of assets. Each asset consists of several other assets. The SCO also contains an assessment that appears as an asset. The rollup will result in one score divided by the total number of SCOs (1) that report a score.

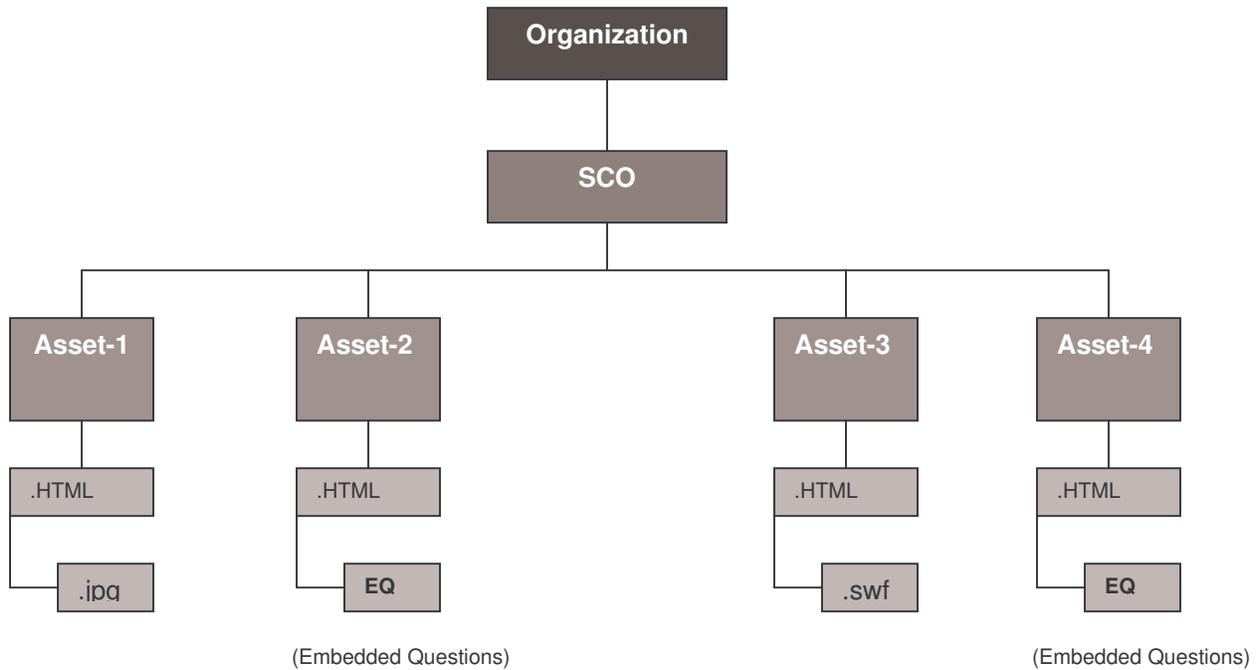


Scenario 1 Rules:	
SCO Behavior	LMS Function
1. To complete the organization, the learner must complete the SCO.	Roll up: All: satisfied, completed
2. To complete the SCO, the learner must pass the assessment in Asset-4 within the SCO.	One score is reported; No additional SCORM function.



Scenario 2: Single SCO, Multiple Assessments

This scenario represents a single SCO composed of multiple pages of assets. Each asset consists of several other assets. The SCO also contains multiple assessments that appear as assets. The rollup will also result in one score divided by the total number of SCOs (1) that report a score because SCORM 1.2 requires that only one score “cmi.core.score.raw” can be reported for each SCO. Therefore, if there are multiple tests within one SCO, the SCO’s internal logic must average the number of embedded questions in order to provide one final “raw” score for the SCO.

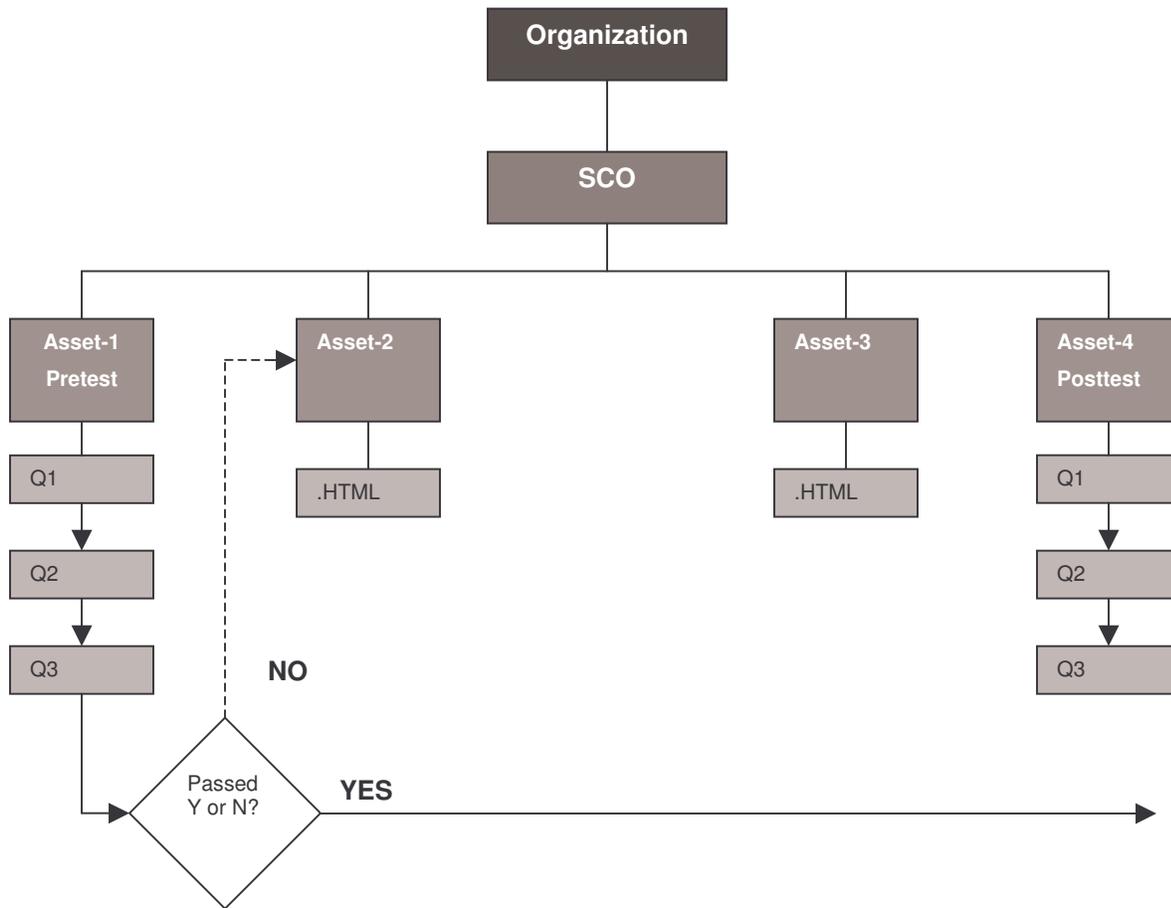


Scenario 2 Rules:	
SCO Behavior	LMS Function
1. To complete the organization, the learner must complete or pass the SCO.	Roll up: All: satisfied, completed
2. To complete or pass the SCO, the learner completes the embedded questions within the SCO.	One score is reported; No additional SCORM function.



Scenario 3: Single SCO, Pretest & Posttest Sequencing (Using Assets)

This scenario represents a SCO composed of multiple pages of assets. Each asset could consist of several other assets. The SCO also contains a pretest and a posttest that are represented as assets. The rollup will result in one final score. The content developer determines the internal sequencing logic.



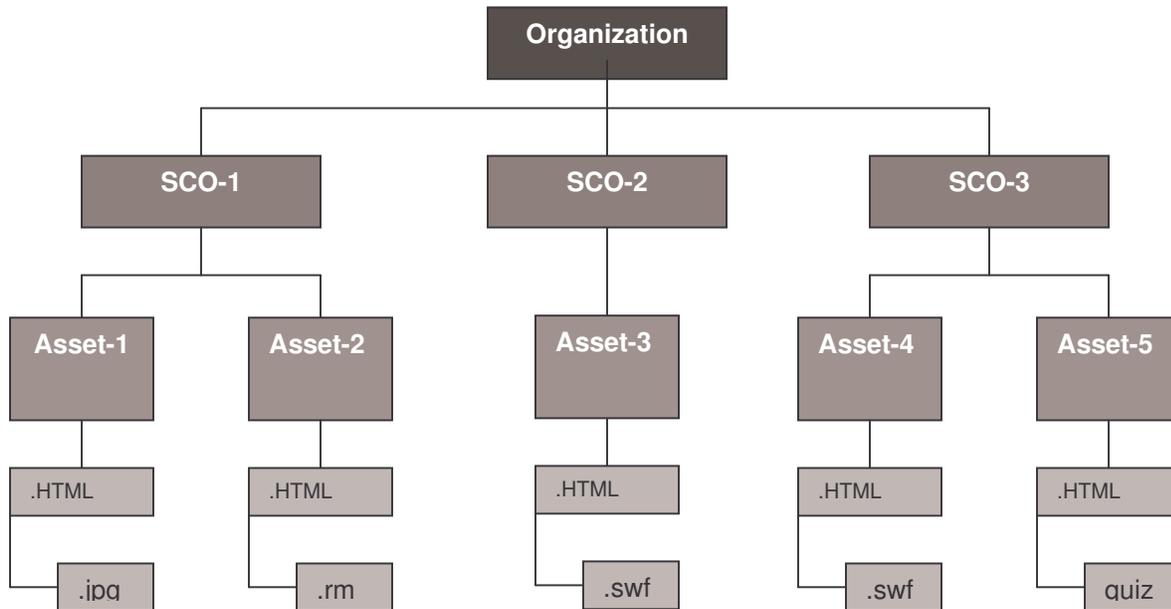
If the learner passed the pretest, send a passed status and the pretest score as the final score for the SCO. If the learner failed the pretest then send the learner to the instruction and posttest.

Scenario 3 Rules:	
SCO Behavior	LMS Function
1. To complete the organization, the learner must complete the SCO.	Roll up: All: satisfied, completed
2. To complete the SCO, the learner must pass the pretest or posttest.	One score is reported; No additional SCORM function.



Scenario 4: Multiple SCOs, One Assessment

This scenario represents three SCOs each with multiple pages of assets. Each asset could consist of several other assets. The last SCO contains an assessment that appears as an asset. The rollup will result in one score divided by the total number of SCOs (1) that report a score.

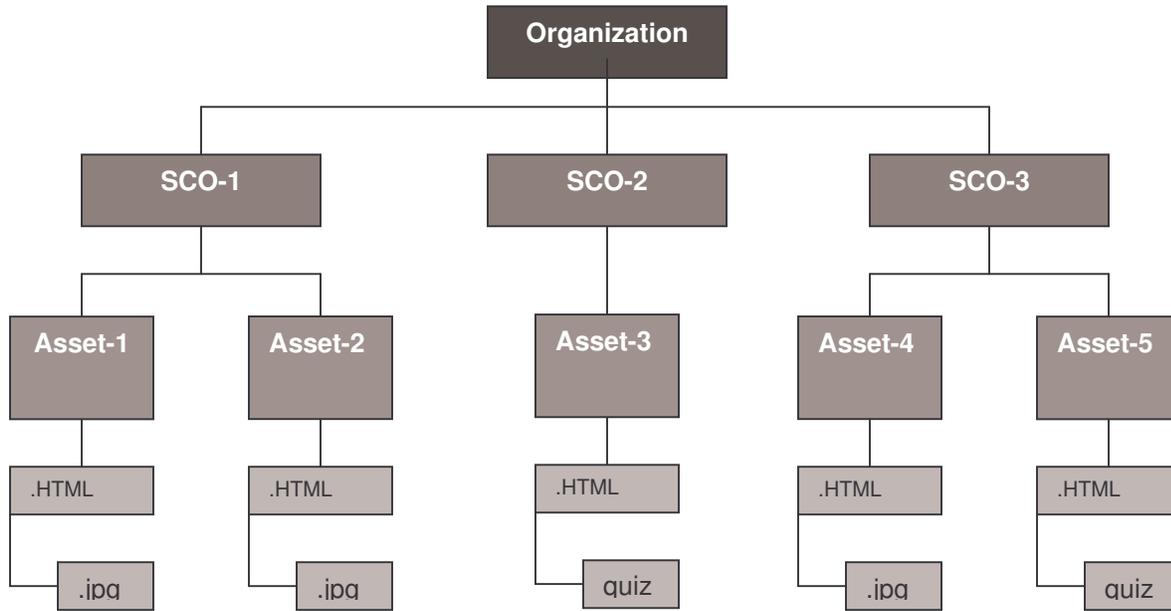


Scenario 4 Rules:	
Behavior	LMS Function
1. To complete the organization, the learner must complete the SCO.	Roll up: All: satisfied, completed
2. To complete the SCO, the learner must complete all screens of instruction and pass the assessment in SCO-3.	One score is reported; No additional SCORM function.



Scenario 5: Multiple SCOs, Multiple Assessments

This scenario represents a SCO composed of multiple pages of assets. Each asset consists of several other assets. Two of the SCOs each contain an assessment that appears. The rollup will result in one cumulative score divided by the total number of SCOs (2) that report a score.

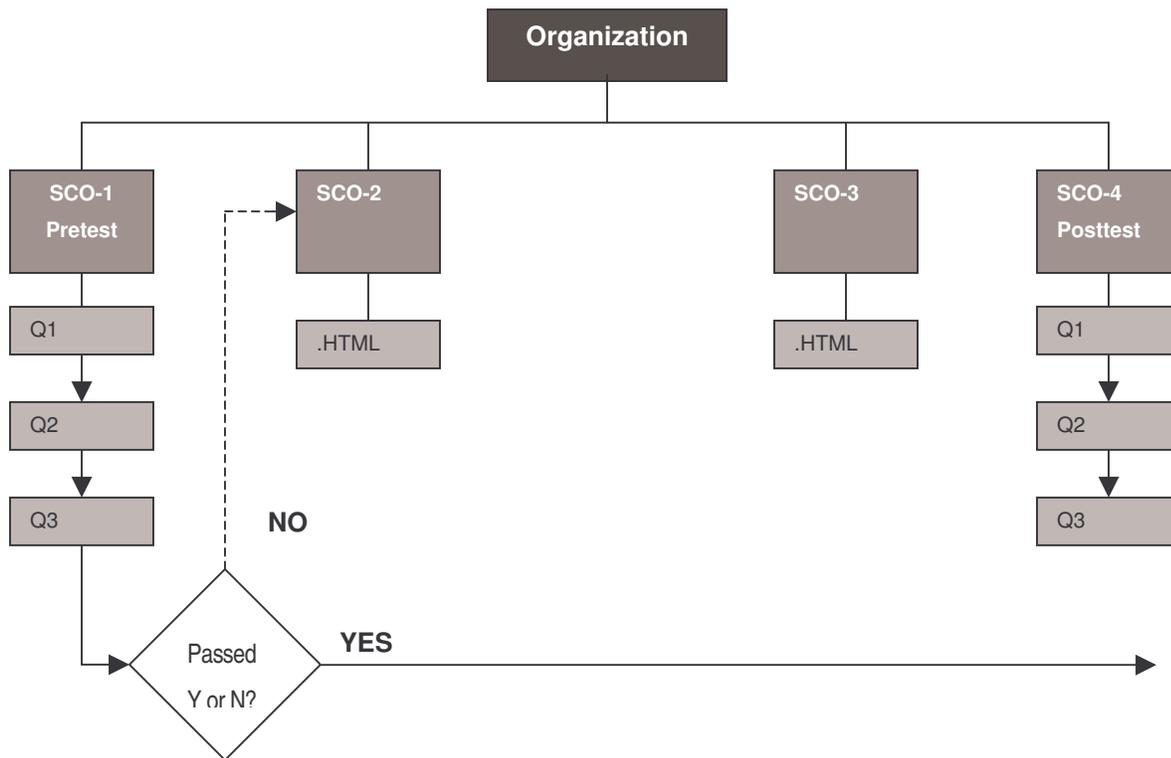


Scenario 5 Rules:	
Behavior	LMS Function
1. To complete the organization, the learner must complete the SCOs.	Roll up: All: satisfied, completed
2. To complete each SCO, the learner must complete all screens of instruction and pass the assessments within each SCO.	One score is rolled up based on all SCOs that report a score



Scenario 6: Multiple SCOs, Pretest Sequencing

This scenario represents an aggregation comprised of multiple SCOs. The aggregation contains a pretest and a posttest that is represented as separate SCOs.



If the learner passed the pretest, send a passed status and the pretest score as the final score for the SCO. If the learner failed the pretest only send a failed status and no score. Then inform the learner to take the instruction and posttest. For more information on Pretest sequencing, read the next section.

Scenario 6 Rules:	
SCO Behavior	LMS Function
1. To complete the organization, the learner must complete the SCO.	Roll up: All: satisfied, completed
2. To complete the SCO, the learner must pass the pretest or posttest.	One score is reported; No additional SCORM function.



Pretest Sequencing

The Navy and most of the E-Learning industry have the capability to 'test out' of a multi SCO course or aggregation for quite some time. By 'testing out' we mean if they pass the pretest in a multi-sco aggregation, then they don't have to experience any more of content of the course (in a multi-SCO course this would mean they would only have to pass the pretest). In the previous scenarios and in most E-Learning environments, pretest sequencing was only possible by creating one large SCO containing all lessons of instruction with the pretest and posttest logic 'hard-wired.' This new ability is not related to the new version of SCORM 1.3 aka SCORM 2004 (which will provide more extensive sequencing and navigation capabilities). More extensive sequencing will be defined in the new version of SCORM 2004, but in the meantime Navy content developers have the option to provide a form of pretest sequencing. This capability is possible because of a rollup customization that was implemented in the LMS upon the Navy's request in early 2003. This customization was requested to accommodate the deficiency of sequencing capabilities in the SCORM 1.2 specification.

There are two conditions by which instructors & developers are now able to provide basic pretest/posttest completion sequencing. The first condition requires a simple configuration of the course within the LMS admin tool. The second condition requires that a certain conditional rule be applied within the pretest & posttest SCO logic. By applying a rule when creating a pretest/posttest course, and properly configuring the rollup feature of the LMS, the course is able to provide useful pretest functionality (the ability to test-out of an entire course by passing the pretest) to each student. The rule that must be applied to a pretest requires that the pretest SCO send a raw score with a passed status only if they pass the pretest, and a failed status with no score if they fail the pretest. The student can take the remaining instruction and/or posttest if they failed the pretest. Successful completion or passing of the pretest will complete the course and generate a certificate without requiring the user to launch the remaining SCOs. Navigating to the instructional content is voluntary, but completion of the pretest and/or posttest is mandatory in order for the course to generate a certificate of completion. These rollup configuration settings can be set for any aggregation of content objects or on a course-by-course basis, and will allow the user to complete a course or 'test-out' when a threshold score is achieved. Since the configuration for each course allows strict averaging of the scores, the pretest can ONLY send a raw score when the student passes in order for this functionality to work.

The following content aggregation scenario was tested to prove the validity of this capability:

1st sco: pretest (*SCO lesson status set to passed/failed, raw score ONLY sent if passed*)

2nd sco: instruction

3rd sco: instruction

4th sco: instruction

5th sco: posttest (*SCO lesson status set to passed/failed, raw score sent*)



The pretest sco would require conditional logic similar to this:

```
var masteryScore = LMSGetValue("cmi.student_data.mastery_score");

/*if the final score calculated for an assessment is greater than or equal to the passing score then
set the raw score and the lesson status to passed */

If (finalScore >= masteryScore) {

    LMSSetValue("cmi.core.score.raw", finalScore);
    LMSSetValue("cmi.core.lesson_status", "passed");

/* else if the final calculated score for an assessment is less than the passing score then ONLY
set the lesson status */

} else if (finalScore < masteryScore) {

    LMSSetValue(cmi.core.lesson_status, "failed");

}
```

The posttest sco would need to have some conditional logic similar to this:

```
var masteryScore = LMSGetValue("cmi.student_data.mastery_score");

/*if the final score calculated for an assessment is greater than or equal to the passing score then
set the raw score and the lesson status to passed */

If (finalScore >= masteryScore) {

    LMSSetValue(cmi.core.score.raw, finalScore);
    LMSSetValue(cmi.core.lesson_status, "passed");

/* else if the final calculated score for an assessment is less than the passing score then set the
raw score and the lesson status*/

} else if (finalScore < masteryScore) {

    LMSSetValue(cmi.core.score.raw, finalScore);
    LMSSetValue(cmi.core.lesson_status, "failed");

}
```

The above logic examples are could be altered depending upon how the content was authored.