



Advanced Distributed Learning Initiative

Sharable Content Object
Reference Model
(SCORM®) 2004 2nd Edition
Addendum

Version 1.0

September 15, 2004



This page intentionally left blank.

Advanced Distributed Learning

SCORM 2004 2nd Edition Addendum Version 1.0

Available at **ADLNet.org**
(<http://www.adlnet.org/>)

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

This page intentionally left blank.

Table of Contents

INTRODUCTION.....	1-1
1.1. Purpose	1-3
SCORM 2004 DEFECT ADDENDA.....	2-1
2.1. Handling of Invalid SetValue() Requests for Data Model Element Collections.....	2-3
2.2. Ambiguous Pseudo-Code in Case #4 of Choice Sequencing Request Process	2-5
2.3. Misevaluation of Traversal Direction.....	2-11
2.4. Measure Rollup Should not be Applied to Leaf Activities	2-14
2.5. Invalid Default Value defined for the measureSatisfactionIfActive attribute	2-16
2.6. Incorrect SPM for the <dataFromLMS> Element.....	2-17
2.7. Handling of Reserved Delimiters	2-18
2.8. Deprecating the adlcp:persistState Attribute	2-23
SCORM 2004 CLARIFICATION/ENHANCEMENT ADDENDA	3-1
3.1. Ambiguous information defined in the language_type Data Type.....	3-3
3.2. Clarification of Learner Session Initialization Requirements	3-4
3.3. Setting the Current Activity to None.....	3-6

This page intentionally left blank.

Introduction

This page intentionally left blank.

1.1. Purpose

The purpose of this document is to track all reported issues with SCORM 2004 that would require updates to the SCORM 2004 2nd Edition documentation suite. This document captures those issues and describes the corrections needed to address them. The information contained in this document supersedes information contained in the SCORM 2004 2nd Edition documentation suite. Vendors should adhere to all changes to SCORM 2004 2nd Edition as described in this document. The SCORM 2004 Conformance Test Suite will be updated to reflect the changes described in this document.

This document should be used in conjunction with the SCORM 2004 2nd Edition until a new edition of SCORM 2004 is published by ADL. This document will be updated to include additional corrections should they become known.

Please submit any additional known issues with the SCORM 2004 2nd Edition to the ADL Technical Team via the Help & Info Center on ADLNet.org.

This document is divided into three major sections. Section 1 explains the purpose of the SCORM 2004 2nd Edition addendum. Section 2 describes addenda that are related to defects in the SCORM 2004 2nd Edition. The types of defects that fall into this category are those that may impact conformance to SCORM 2004, depending on current implementations. Section 3 describes addenda that are clarifications or enhancements that do not impact SCORM 2004 conformance.

This page intentionally left blank.

SCORM 2004 Defect Addenda

This page intentionally left blank.

2.1. Handling of Invalid SetValue() Requests for Data Model Element Collections

The SCORM Run-Time Environment (RTE) Version 1.3.1 does not clearly describe how to handle invalid `SetValue()` requests invoked against data model elements in a “to-be” created record of a data model element collection. More specifically, there is no specific statement of when a data model element record should be created in response to a `SetValue()` request invoked against one of its children, increasing the count of its containing collection.

For example, assume that the value for `cmi.comments_from_learner._count` is zero (the SCO has not set any comments from learner) and the SCO attempts to make the following `SetValue()` request:

```
SetValue("cmi.comments_from_learner.0.comment", "{lang=}")
```

This `SetValue()` request is invalid because the value provided for the language delimiter is not a valid `language_type` – it cannot be an empty string (refer to the SCORM RTE Version 1.3.1, Section 4.1.1.7: *Data Types* for requirements of a valid `language_type`). In this case, the LMS is required to return `false` and set the API error code to 406 – `Data Model Element Type Mismatch`.

This issue is experienced when continuing the example. SCORM does not clearly define how an LMS should respond to the following `GetValue()` requests when they are invoked immediately following the previous `SetValue()` request.

```
GetValue("cmi.comments_from_learner.0.comment")
```

```
GetValue("cmi.comments_from_learner._count")
```

2.1.1. Rationale for Change

Invalid `SetValue()` requests (regardless of the data model element) always result in an LMS returning `false` and setting the API error code. The LMS is required to not alter the state of the data model element’s value. In the case where these `SetValue()` requests are invoked against data model elements in a “to-be” created record, not altering the state of the data model element implies that no record is created and that the containing collection’s size does not change.

2.1.2. SCORM Update

Section 4.1.1.3 Handling Collections found in the SCORM Run-Time Environment Version 1.1.3 will be updated to include the following required behavior:

Failure to set a data model element in a “to-be” created data model record does not result in any data being persisted and does not increase the containing collection size.

The following child data model elements are affected by this change. If any of these elements is set successfully, it would cause a newly created record to be added to the containing collection, increasing the collection’s count by 1.

- `cmi.objectives.n.id`
- `cmi.interactions.n.id`
- `cmi.interactions.n.objectives.n.id`
- `cmi.comments_from_learner.n.comment`
- `cmi.comments_from_learner.n.location`
- `cmi.comments_from_learner.n.timestamp`

When an LMS receives an invalid `SetValue()` request against one of these data model elements, the LMS will return `false` and set the appropriate API error code. The size of containing collection shall not be incremented. If a `_count` request is made prior to the invalid `SetValue()` request and a `_count` request is made immediately after the invalid `SetValue()` request, then the value returned by both calls shall be identical (i.e., the collection size has not changed).

An update will be made to define that if an LMS receives a `GetValue()` request immediately following an invalid `SetValue()` request, then the behavior shall be:

- (1) Return an empty characterstring (“”)
- (2) Set the API Error Code to 301 - General Get Failure

2.2. Ambiguous Pseudo-Code in Case #4 of Choice Sequencing Request Process

This addendum addresses a discrepancy found in Choice Sequencing Request Process (Section SB.2.9) of the SCORM Sequencing and Navigation (SN) Version 1.3.1. The discrepancy can be found in Case #4 which starts on line 11. Case #4 deals with the Choice Sequencing Request scenario where the target activity is an ancestor of the Current Activity. In this case, the Activity Tree traversal will be up the “active” path rather than across the tree; however, the pseudo code includes an evaluation of Constrained Choice attribute that requires an ambiguous traversal either Backward or Forward in the Activity Tree relative to the constrained activity.

2.2.1. Rationale for Change

When the target of a choice navigation request is an ancestor of the current activity (Case #4), the Constrained Choice attribute will have no effect. In Case #4, only the target or its immediate “next” activity (if the target is a cluster – Section SB.2.9, line 14), could possibly be identified for delivery, which is the intended effect of the Constrained Choice Attribute. In this case, the evaluation of Constrained Choice adds no value and potentially leads to an ambiguous traversal direction.

2.2.2. SCORM Update

The evaluation of the Constrained Choice attribute is unnecessary and ambiguous when performed during Case #4 of the Sequencing Choice Process. The Choice Sequencing Request Process (Section SB.2.9) will be updated to delete the pseudo-code that references the constrained activity in Case #4 (this includes lines 11.3, 11.4.2.*, and 11.5.*).

The updated Choice Sequencing Process is reproduced in whole for easy reference.

Choice Sequencing Request Process [SB.2.9] (for a target activity; may return a delivery request; may change the <i>Current Activity</i> ; may return an exception code):		
Reference: Activity is Active AM.1.1; Activity is Suspended AM.1.1; Available Children AM.1.1; Check Activity Process UP.5; Choice Activity Traversal Subprocess SB.2.4; Current Activity AM.1.2; End Attempt Process UP.4; Flow Subprocess SB.2.3; Sequencing Control Mode Choice SM.1; Sequencing Control Choice Exit SM.1; Sequencing Rules Check Process UP.2; Terminate Descendent Attempts Process UP.3; <i>adlseq:constrainedChoice SCORM SN</i> ; <i>adlseq:preventActivation SCORM SN</i>		
1.	If there is no target activity Then	There must be a target activity for choice
1.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: <i>SB.2.9-1</i>)	Nothing to deliver
	End If	
2.	If the target activity is not the root of the activity tree Then	

2.1.	If the <i>Available Children</i> for the parent of the target activity does not contain the target activity Then	The activity is currently not available
2.1.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-2)	Nothing to deliver
	End If	
	End If	
3.	Form the activity path as the ordered series of activities from the root of the activity tree to the target activity, inclusive	
4.	For each activity in the activity path	
4.1.	Apply the <i>Sequencing Rules Check Process</i> to the activity and the <i>Hide from Choice sequencing rules</i>	Cannot choose something that is hidden
4.2.	If the <i>Sequencing Rules Check Process</i> does not return <i>Nil</i> Then	
4.2.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-3)	Nothing to deliver
	End If	
	End For	
5.	If the target activity is not the root of the activity tree Then	
5.1.	If the <i>Sequencing Control Mode Choice</i> for the parent of the target activity is <i>False</i> Then	Confirm that control mode allow 'choice' of the target
5.1.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-4)	Nothing to deliver
	End If	
	End If	
6.	If the <i>Current Activity</i> is <i>Defined</i> Then	Has the sequencing session already begun?
6.1.	Find the common ancestor of the <i>Current Activity</i> and the target activity	
7.	Else	
7.1.	Set common ancestor is the root of the activity tree	No, choosing the target will start the sequencing session
	End If	
8.	Case: <i>Current Activity</i> and target activity are identical	Case #1 - select the current activity
8.1.	Break All Cases	Nothing to do in this case
	End Case	
9.	Case: <i>Current Activity</i> and the target activity are siblings	Case #2 - same cluster; move toward the target activity
9.1.	Form the activity list as the ordered sequence of activities from the <i>Current Activity</i> to the target activity, exclusive of the target activity	We are attempted to walk toward the target activity. Once we reach the target activity, we don't need to test it.

9.2.	If the activity list is <i>Empty</i> Then	Nothing to choose
9.2.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: <i>SB.2.9-5</i>)	Nothing to deliver
	End If	
9.3.	If the target activity occurs after the <i>Current Activity</i> in preorder traversal of the activity tree Then	
9.3.1.	traverse is <i>Forward</i>	
9.4.	Else	
9.4.1.	traverse is <i>Backward</i>	
	End If	
9.5.	For each activity on the activity list	
9.5.1.	Apply the <i>Choice Activity Traversal Subprocess</i> to the activity in the traverse direction	
9.5.2.	If the <i>Choice Activity Traversal Subprocess</i> returns <i>False</i> Then	
9.5.2.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: the exception identified by the <i>Choice Activity Traversal Subprocess</i>)	Nothing to deliver
	End If	
	End For	
9.6.	Break All Cases	
	End Case	
10.	Case: <i>Current Activity</i> and common ancestor are the same Or <i>Current Activity</i> is Not Defined	Case #3 - path to the target is forward in the activity tree
10.1.	Form the activity path as the ordered series of activities from the common ancestor to the target activity, exclusive of the target activity	
10.2.	If the activity path is <i>Empty</i> Then	
10.2.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: <i>SB.2.9-5</i>)	Nothing to deliver
	End If	
10.3.	For each activity on the activity path	
10.3.1.	Apply the <i>Choice Activity Traversal Subprocess</i> to the activity in the <i>Forward</i> direction	
10.3.2.	If the <i>Choice Activity Traversal Subprocess</i> returns <i>False</i> Then	
10.3.2.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: the exception identified by the <i>Choice Activity Traversal Subprocess</i>)	Nothing to deliver
	End If	
10.3.3.	If <i>Activity is Active</i> for the activity is <i>False</i> And (the activity is Not the common ancestor And <i>adlseq:preventActivation</i> for the activity is <i>True</i>) Then	If the activity being considered is not already active, make sure we are allowed to activate it
10.3.3.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: <i>SB.2.9-6</i>)	Nothing to deliver
	End If	
	End For	
10.4.	Break All Cases	
	End Case	
11.	Case: Target activity is the common ancestor of the <i>Current Activity</i>	Case #4 - path to the target is backward in the

		activity tree
11.1.	Form the activity path as the ordered series of activities from the <i>Current Activity</i> to the target activity, inclusive	
11.2.	If the activity path is <i>Empty</i> Then	
11.2.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-5)	Nothing to deliver
	End If	
11.3.	For each activity on the activity path	
11.3.1.	If the activity is not the last activity in the activity path Then	
11.3.1.1.	If the <i>Sequencing Control Choice Exit</i> for the activity is <i>False</i> Then	Make sure an activity that should not exit will exit if the target is delivered.
11.3.1.1.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-7)	Nothing to deliver
	End If	
	End If	
	End For	
11.4.	Break All Cases	
	End Case	
12.	Case: Target activity is forward from the common ancestor activity	Case #5 - target is a descendent activity of the common ancestor
12.1.	Form the activity path as the ordered series of activities from the <i>Current Activity</i> to the common ancestor, inclusive	
12.2.	If the activity path is <i>Empty</i> Then	
12.2.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-5)	Nothing to deliver
	End If	
12.3.	Set constrained activity to <i>Undefined</i>	
12.4.	For each activity on the activity path	Walk up the tree to the common ancestor
12.4.1.	If the activity is not the last activity in the activity path Then	
12.4.1.1.	If the <i>Sequencing Control Choice Exit</i> for the activity is <i>False</i> Then	Make sure an activity that should not exit will exit if the target is delivered
12.4.1.1.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-7)	Nothing to deliver
	End If	
	End If	
12.4.2.	If constrained activity is <i>Undefined</i> Then	Find the closest constrained activity to the current activity
12.4.2.1.	If <i>adlseq:constrainedChoice</i> for the activity is <i>True</i> Then	
12.4.2.1.1.	Set constrained activity to activity	
	End If	
	End If	
	End For	

12.5.	If constrained activity is <i>Defined</i> Then	
12.5.1.	If the target activity is <i>Forward</i> in the activity tree relative to the constrained activity Then	
12.5.1.1.	traverse is <i>Forward</i>	'Flow' in a forward direction to see what activity comes next
12.5.2.	Else	
12.5.2.1.	traverse is <i>Backward</i>	'Flow' in a backward direction to see what activity comes next
	End If	
12.5.3.	Apply the <i>Choice Flow Subprocess</i> to the constrained activity in the traverse direction	
12.5.4.	Set activity to consider to the activity identified by the <i>Choice Flow Subprocess</i>	
12.5.5.	If the target activity is Not an available descendent of the activity to consider And (the target activity is Not the constrained activity Or the target activity is Not the activity to consider) Then	Make sure the target activity is within the set of 'flow' constrained choices
12.5.5.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: <i>SB.2.9-8</i>)	
	End If	
	End If	
12.6.	Form the activity path as the ordered series of activities from the common ancestor to the target activity, exclusive of the target activity	
12.7.	If the activity path is <i>Empty</i> Then	
12.7.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: <i>SB.2.9-5</i>)	Nothing to deliver
	End If	
12.8.	If the target activity is forward in the activity tree relative to the <i>Current Activity</i> Then	Walk toward the target activity
12.8.1.	For each activity on the activity path	
12.8.1.1.	Apply the <i>Choice Activity Traversal Subprocess</i> to the activity in the <i>Forward</i> direction	
12.8.1.2.	If the <i>Choice Activity Traversal Subprocess</i> returns <i>False</i> Then	
12.8.1.2.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: the exception identified by the <i>Choice Activity Traversal Subprocess</i>)	Nothing to deliver
	End If	
12.8.1.3.	If <i>Activity is Active</i> for the activity is <i>False</i> And (the activity is Not the common ancestor And <i>adlseq:preventActivation</i> for the activity is <i>True</i>) Then	If the activity being considered is not already active, make sure we are allowed to activate it
12.8.1.3.1.	Exit <i>Choice Sequencing Request Process</i> (Delivery Request: <i>n/a</i> ; Exception: <i>SB.2.9-6</i>)	Nothing to deliver
	End If	

	End For	
12.9.	Else	
12.9.1.	For each activity on the activity path	
12.9.1.1.	If <i>Activity is Active</i> for the activity is <i>False</i> And (the activity is <i>Not</i> the common ancestor And <i>adlseq:preventActivation</i> for the activity is <i>True</i>) Then	If the activity being considered is not already active, make sure we are allowed to activate it
12.9.1.1.1.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-6)	Nothing to deliver
	End If	
	End For	
	End If	
12.10.	Break All Cases	
	End Case	
13.	If the target activity is a leaf activity Then	
13.1.	Exit Choice Sequencing Request Process (Delivery Request: the target activity; Exception: n/a)	
	End If	
14.	Apply the <i>Flow Subprocess</i> to the target activity in the <i>Forward</i> direction with consider children equal to <i>True</i>	The identified activity is a cluster. Enter the cluster and attempt to find a descendent leaf to deliver
15.	If the <i>Flow Subprocess</i> returns <i>False</i> Then	Nothing to deliver, but we succeeded in reaching the target activity - move the current activity
15.1.	Apply the <i>Terminate Descendent Attempts Process</i> to the common ancestor	
15.2.	Apply the <i>End Attempt Process</i> to the common ancestor	
15.3.	Set the <i>Current Activity</i> to the target activity	
15.4.	Exit Choice Sequencing Request Process (Delivery Request: n/a; Exception: SB.2.9-9)	Nothing to deliver
16.	Else	
16.1.	Exit Choice Sequencing Request Process (Delivery Request: for the activity identified by the <i>Flow Subprocess</i>; Exception: n/a)	
	End If	

2.3. Misevaluation of Traversal Direction

This addendum addresses an error discovered in the Flow Activity Traversal Subprocess (SB.2.2) of the SCORM SN Version 1.3.1. The error prevents the proper reversal of the flow traversal direction during the evaluation of a *Previous* sequencing request. This case only occurs when a *Forward Only* cluster is encountered and all of its children are skipped during the evaluation of a *Previous* sequencing request. In this case, the Flow Activity Traversal Subprocess needs to switch direction twice, once to move forward through the *Forward Only* cluster and then once again to continue to move backward (honoring the original request) from the *Forward Only* cluster to its predecessor.

2.3.1. Rationale for Change

This error is a bug in the SCORM Sequencing pseudo code. A strict implementation of the SCORM Sequencing pseudo code would prevent an LMS from recognizing a change in traversal direction and from successfully passing the SCORM Sequencing Conformance Test Case CM-3b.

2.3.2. SCORM Update

Line 3.3.1 of the Flow Activity Traversal Subprocess (SB.2.2) will be updated. The currently referenced (local variable) “traversal direction” will be updated to “previous traversal direction.”

The updated Flow Activity Traversal Subprocess is reproduced in whole for easy reference.

Flow Activity Traversal Subprocess [SB.2.2] (for an activity, a traversal direction, and a previous traversal direction; returns the ‘next’ activity in a directed traversal of the activity tree and True if the activity can be delivered; may return an exception code):		
Reference: Check Activity Process UP.5; Flow Activity Traversal Subprocess SB.2.2; Flow Tree Traversal Subprocess SB.2.1; Sequencing Control Flow SM.1; Sequencing Rules Check Process UP.2		
1.	If <i>Sequencing Control Flow</i> for the parent of the activity is <i>False</i> Then	Confirm that ‘flow’ is enabled
1.1.	Exit <i>Flow Activity Traversal Subprocess</i> (Deliverable: <i>False</i> ; Next Activity: the activity; Exception: <i>SB.2.2-1</i>)	
	End If	
2.	Apply the <i>Sequencing Rules Check Process</i> to the activity and its <i>Skipped</i> sequencing rules	
3.	If the <i>Sequencing Rules Check Process</i> does not return <i>Nil</i> Then	Activity is skipped, try to go to the ‘next’ activity
3.1.	Apply the <i>Flow Tree Traversal Subprocess</i> to the activity in the traversal direction and the previous traversal direction with consider children equal to <i>False</i>	
3.2.	If the <i>Flow Tree Traversal Subprocess</i> does not identify an activity	

	Then	
3.2.1.	Exit Flow Activity Traversal Subprocess (Deliverable: False; Next Activity: the activity; Exception: exception identified by the Flow Tree Traversal Subprocess)	
3.3.	Else	
3.3.1.	If the previous traversal direction is Backward And the Traversal Direction returned by the Flow Tree Traversal Subprocess is Backward Then	Make sure the recursive call is considers the correct direction
3.3.1.1.	Apply the Flow Activity Traversal Subprocess to the activity identified by the Flow Tree Traversal Subprocess in the traversal direction and a previous traversal direction of n/a	Recursive call – make sure the ‘next’ activity is OK
3.3.2.	Else	
3.3.2.1.	Apply the Flow Activity Traversal Subprocess to the activity identified by the Flow Tree Traversal Subprocess in the traversal direction and a previous traversal direction of previous traversal direction	Recursive call – make sure the ‘next’ activity is OK
	End If	
3.3.3.	Exit Flow Activity Traversal Subprocess - (Return the results of the recursive Flow Activity Traversal Subprocess)	Possible exit from recursion
	End If	
	End If	
4.	Apply the Check Activity Process to the activity	Make sure the activity is allowed
5.	If the Check Activity Process returns True Then	
5.1.	Exit Flow Activity Traversal Subprocess (Deliverable: False; Next Activity: the activity; Exception: SB.2.2-2)	
	End If	
6.	If the activity is not a leaf node in the activity tree Then	Cannot deliver a non-leaf activity; enter the cluster looking for a leaf
6.1.	Apply the Flow Tree Traversal Subprocess to the activity in the traversal direction and a previous traversal direction of n/a with consider children equal to True	
6.2.	If the Flow Tree Traversal Subprocess does not identify an activity Then	
6.2.1.	Exit Flow Activity Traversal Subprocess (Deliverable: False; Next Activity: the activity; Exception: exception identified by the Flow Tree Traversal Subprocess)	
6.3.	Else	
6.3.1.	If the traversal direction is Backward And the traversal direction returned by the Flow Tree Traversal Subprocess is Forward Then	Check if we are flowing backward through a forward only cluster - must move forward instead
6.3.1.1.	Apply the Flow Activity Traversal Subprocess to the activity identified by the Flow Tree Traversal Subprocess in the Forward direction with the previous traversal direction of Backward	Recursive call – Make sure the identified activity is OK
6.3.2.	Else	
6.3.2.1.	Apply the Flow Activity Traversal Subprocess to the activity	Recursive call –

	identified by the <i>Flow Tree Traversal Subprocess</i> in the traversal direction and a previous traversal direction of <i>n/a</i>	Make sure the identified activity is OK
	End If	
6.3.3.	Exit <i>Flow Activity Traversal Subprocess</i> - (Return the results of the recursive <i>Flow Activity Traversal Subprocess</i>)	Possible exit from recursion
	End If	
	End If	
7.	Exit <i>Flow Activity Traversal Subprocess</i> (Deliverable: <i>True</i> ; Next Activity: the activity; Exception: <i>n/a</i>)	Found a leaf

2.4. Measure Rollup Should not be Applied to Leaf Activities

This addendum addresses a side effect discovered in the Overall Rollup Process (RB.1.5) found in the SCORM SN Version 1.3.1. The side effect discovered prevents scores reported by SCOs from being utilized during sequencing evaluations. Section 4.6.1 of the SCORM SN Version 1.3.1 defines when the Overall Rollup Process is applied through its extended rollup process and some rules constraining how that rollup is evaluated. Each time the Overall Rollup Process is invoked, it applies the various rollup subprocesses to each activity along the “active path” – the path from the Current Activity (typically a leaf) and the root of the Activity Tree. For each activity, even a Current Activity that happens to be a leaf, the Measure Rollup Process (RB.1.1) is applied. However, the purpose of the Measure Rollup Process is to aggregate (rollup) the measures of the target activity’s children. If none of the activity’s children have a measure or if the activity has no children, the resulting measure is “unknown.”

2.4.1. Rationale for Change

A strict implementation of the SCORM Sequencing pseudo code would result in all delivered leaf activities having a measure of “unknown” because the Measure Rollup Process would interpret the lack of *counted measures* as an indication that the rolled up measure should be “unknown” (RB.1.1, line 5.2.1). For sequencing purposes, the “unknown” value would be used instead of any measure provided by a SCO (cmi.score.scaled), defeating the intent of mapping SCORM Run-Time Data to the associated activity’s tracking data. The net result would prevent an LMS from successfully passing several of the measure-based SCORM Sequencing Conformance Test Cases (MS-*).

2.4.2. SCORM Update

Although Section 4.6.1 of the SCORM SN Version 1.3.1 states that “Rollup rules have no effect if defined on a leaf activity – there is nothing to rollup”, this instruction could be interpreted as applying only to Rollup Rule Descriptions, as defined in the Sequencing Definition Model, and not to general measure rollup. To ensure that there is no ambiguity in defined sequencing behaviors, the following bullet will be added to the SCORM SN Version 1.3.1 in Section 4.6.1:

- Measure rollup is not applied to leaf activities.

In addition, a clause will be added to the Overall Rollup Process (RB.1.5) that applies to line 3.1. The clause will ensure that the Measure Rollup Process (RB.1.1) is not applied to leaf activities. This change is normative behavior and will enforce that all implementation will utilize the measure reported by a SCO for sequencing purposes.

The updated Overall Rollup Process is reproduced in whole for easy reference.

Overall Rollup Process [RB.1.5] (for an activity; may change the tracking information for the activity and its ancestors):		
Reference: Activity Progress Rollup Process RB.1.3; Measure Rollup Process RB.1.1; Objective Rollup Process RB.1.2; Tracked SM.11; Tracking Model TM		
1.	Form the activity path as the ordered series of activities from the root of the activity tree to the activity, inclusive, in reverse order.	
2.	If the activity path is <i>Empty</i> Then	
2.1.	Exit <i>Overall Rollup Process</i>	Nothing to rollup
	End If	
3.	For each activity in the activity path	
3.1.	If the activity has children Then	Only apply Measure Rollup to non-leaf activities
3.1.1.	Apply the <i>Measure Rollup Process</i> to the activity	Rollup the activity's measure
	End If	
3.2.	Apply the appropriate <i>Objective Rollup Process</i> to the activity	Apply the appropriate behavior described in section RB.1.2, based on the activity's defined sequencing information
3.3.	Apply the <i>Activity Progress Rollup Process</i> to the activity	Apply the appropriate behavior described in section RB.1.3, based on the activity's defined sequencing information
	End For	
4.	Exit <i>Overall Rollup Process</i>	

2.5. Invalid Default Value defined for the `measureSatisfactionIfActive` attribute

This addendum address an error found in the SCORM Content Aggregation Model (CAM) Version 1.3.1. Section 5.1.11 `<rollupConsiderations>` Element contains an optional attribute named `measureSatisfactionIfActive`. This attribute is defined with an incorrect default value. The value defined in the CAM is `false`. The default value of the `measureSatisfactionIfActive` should be `true`. The `adlseq_v1p3.xsd` correctly identifies the default value as `true`.

2.5.1. Rationale for Change

This change is being made to correctly identify the default value of the `measureSatisfactionIfActive` attribute.

2.5.2. SCORM Update

The `measureSatisfactionIfActive` attribute defined for the `<rollupConsiderations>` element found in Section 5.1.11 `<rollupConsiderations>` Element will be updated to change the default value of the `measureSatisfactionIfValid` to `true`. The section will be updated as follows:

From:

- `measureSatisfactionIfActive` (optional, default value = `false`). This attribute indicates if the measure should be used to determine satisfaction during rollup when the activity is active. XML Data Type: `xs:boolean`.

To:

- `measureSatisfactionIfActive` (optional, default value = `true`). This attribute indicates if the measure should be used to determine satisfaction during rollup when the activity is active. XML Data Type: `xs:boolean`.

2.6. Incorrect SPM for the <dataFromLMS> Element

This addendum addresses an error in the definition of the Smallest Permitted Maximum (SPM) for the ADL Content Packaging extension element <adlcp:dataFromLMS>. The CAM Version 1.3.1 incorrectly defines the SPM for the <adlcp:dataFromLMS> element's value as 4096 characters.

2.6.1. Rationale for Change

The ADL Content Packaging Extension element, <adlcp:dataFromLMS>, is used to initialize the `cmi.launch_data` SCORM Run-Time Environment Data Model element. The `cmi.launch_data` data model element has a defined SPM of 4000 characters. The IEEE Data Model Standard defines the SPM of the Launch Data element (i.e., maps to the `cmi.launch_data` model element) as 4000 characters. Since the SCORM defines that the <adlcp:dataFromLMS> element is used to initialize the `cmi.launch_data` data model element, then the SPMs should match.

2.6.2. SCORM Update

The Data Type section of the <dataFromLMS> element found in Section 3.4.1.14 <dataFromLMS> Element will be updated to change the SPM for the element's value from 4096 characters to 4000 characters. The section will be updated as follows:

From:

<p>Data Type: The <dataFromLMS> element is represented as a characterstring element. The characterstring has an SPM of 4096 characters.</p>

To:

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

2.7. Handling of Reserved Delimiters

This addendum addresses various issues involving the delimiters used by the SCORM Run-Time Environment Data Model dot-notation binding:

- Discrepancy between, the types of delimiters, and
- Discrepancy between the requirements on the syntax and placement of delimiters.

2.7.1. Rationale For Change

This change is being made to remove the discrepancies between the types of delimiters, syntax requirements for each type and placement requirements for each type. The current text is confusing and does not clearly address these issues.

2.7.2. SCORM Update

The following updates will be made to Section 4.1.1.6 Reserved Delimiters:

- Table 4.1.1.6a will be broken up to resolve the ambiguity between the two types of delimiters (Property and Separator delimiters).
- The section will be updated to describe the Property Delimiter Syntax Requirements, Property Delimiter Placement Syntax, Separator Delimiter Syntax Requirements and Separator Placement Syntax.

As mentioned above, Table 4.1.1.6a, will be broken up into two tables. Table 4.1.1.6a will be updated to describe the Reserved Property Delimiters as follows:

Table 4.1.1.6a: Reserved Property Delimiters

Reserved Delimiter Syntax	Default Value	Example
{lang=<language_type>}	{lang=en}	{lang=en}
{case_matters=<boolean>}	{case_matters=false}	{case_matters=true} {case_matters=false}
{order_matters=<boolean>}	{order_matters=true}	{order_matters=true} {order_matters=false}

Because these delimiters are not required, the default value shall be assumed for those cases where the delimiter is not specified. If the delimiters are used in the

characterstring, there are other requirements on placement of the delimiter and the delimiter syntax.

Property Delimiter Syntax Requirements: The delimiter shall be treated as a constant set of characters with the following format::

```
delimiter ::= "{" + name + "=" + value + "}"
```

NOTE: The "{" and "}" are required to indicate the beginning and ending portions of a delimiter. The "=" is required to separate the `name` and `value` pieces of the delimiter. The absences of these required characters will cause the delimiter to not be recognized by the system; instead, the set of characters will be treated as part of the underlying characterstring data value.

The `name` represents the identifier of the delimiter. The `name` is represented by a set of reserved tokens:

- `lang`
- `case_matters`
- `order_matters`

For the `{lang=<language_type>}` delimiter to be recognized as the language for the characterstring, it must be placed at the beginning of the characterstring being qualified. If the `{lang=<language_type>}` delimiter is not the first set of characters, then the default language shall be assumed.

The `name` token must be represented as-is (i.e., one of the following: `lang`, `case_matters` or `order_matters`). Any derivatives of these tokens (e.g., padding the token names with whitespace) will result in an unrecognized delimiter and the set of characters will be treated as part of the underlying characterstring.

The `value` indicates the value for the named delimiter. The `value` portion of the delimiter is restricted to the following:

- `lang`: Restricted to the value represented by a `language_type` (Refer to Section 4.1.1.7: *Data Types* for requirements of a `language_type`).
- `case_matters`: Restricted to either `true` or `false`
- `order_matters`: Restricted to either `true` or `false`

NOTE: If the value does not meet its named delimiter's type requirements, then the delimiter is improperly formed and the characterstring does not meet the requirements of its type (i.e., causing a 406 - Data Type Mismatch error to occur).

Valid Examples:

- `SetValue("cmi.comments_from_learner.0.comment", "{lang=en}Characterstring in the English language")`
- `SetValue("cmi.interactions.0.correct_response.0.pattern", "{lang=en}{case_matters=true}Characterstring in the English language where the case matters")`

-
- SetValue("cmi.comments_from_learner.0.comment", "{lang =fr}Characterstring in the English language")

There is no delimiter qualifying this Characterstring - "{lang =fr}" is not considered a language delimiter because it contains whitespace, which is not lexically equivalent to the lang reserved delimiter. In this case, the overall Characterstring includes {lang =fr} and is still considered valid; its default language is English ("en"). If this SetValue call is invoked, the LMS shall set the data model element associated with the call to

"{lang =fr}Characterstring in the English language", set the error code to 0 - No error and return true.

- SetValue("cmi.comments_from_learner.0.comment", "{case_matters=invalid}Characterstring in the English language")

There are no delimiters qualifying this Characterstring - "{case_matters=invalid" is not part of the defined format for cmi.comments_from_learner.n.comment. In this case the overall Characterstring includes {case_matters=invalid} and is still considered valid; its default language is English ("en"). If this SetValue call is invoked, the LMS shall set the data model element associated with the call to

"{case_matters=invalid}Characterstring in the English language", set the error code to 0 - No error and return true.

Invalid Examples:

- SetValue("cmi.interaction.0.correct_response.0.pattern", "{case_matters=invalid}{lang=en}Characterstring in the English language")

Assuming that the type of cmi.interaction.0 is fill-in, the case matters delimiter is invalid because it requires a value of true or false. This example uses "invalid." Because the delimiter is improperly formed, an LMS should not set the data model element associated with the call, set the error code to 406 - Data Type Mismatch, and return false.

- SetValue("cmi.comments_from_learner.0.comment", "{lang= fr}Characterstring in the French language")

In this example, the lang delimiter is invalid because its value includes whitespace, which is not part of a valid language string. Because the delimiter is improperly formed an LMS should not set the data model element associated with the call, set the error code to 406 - Data Type Mismatch, and return false.

Property Delimiter Placement Requirements: The delimiters are required to be placed in specific positions within the characterstring. In those cases where a combination of delimiters may be used, the order of the delimiters is described by the data model element. If a default value is used (implied by the absence of a delimiter) for one of the delimiters in the set of delimiters, then the order should still be preserved. The delimiters

shall be concatenated together with no whitespace permitted between the delimiters. For example:

- {case_matters=true}{order_matters=true}

No whitespace or other characters are permitted prior to the first delimiter identified in the characterstring. If there are no delimiters, which implies that the default values are being used, then the value represents the characterstring used for the data model element.

NOTE: If any whitespace or other character is found at the beginning of the characterstring, then this will cause the set of characters to be treated as part of the underlying characterstring.

As mentioned above, Table 4.1.1.6a, will be broken up into two tables. Table 4.1.1.6b will be updated to describe the Reserved Separator Delimiters as follows:

Table 4.1.1.6b: Reserved Separator Delimiter

Reserved Delimiter Syntax	Default Value	Example
[.]	Not applicable, needs to be provided	Used to separate a pair of values that are related for an interaction: 1[.]a
[,]	Not applicable, needs to be provided	Used to separate a set of values for an interaction's collection: 1[.]a[,]2[.]c[,]3[.]b
[:]	Not applicable, needs to be provided	Used to represent a separator between a range of numeric values: 1[:]100 - a range where the numeric value is between 1 and 100 (inclusive)

Separator Delimiter Syntax Requirements: The delimiter shall be treated as a constant set of characters with the following format:

delimiter ::= "[" + reserved_character + "]"

NOTE: The “[“ and “]” are required to indicate the beginning and ending portions of the delimiter. The absences of these required characters will cause the delimiter to not be recognized by the system; instead, the set of characters will be treated as part of the underlying characterstring.

The `reserved_character` represents the defined separator. The `reserved_character` is represented by a set of reserved tokens:

- .
- ,
- :

NOTE: The `reserved_character` token must be represented as-is (i.e., one of the following: `[.]`, `[,]` or `[:]`). Any derivatives of these tokens (e.g., padding the `reserved_character` tokens with whitespace) will result in an unrecognized delimiter and the set of characters will be treated as part of the underlying characterstring.

Separator Delimiter Placement Requirements: The delimiters are required to be placed in specific positions within the characterstring. The delimiter is used to separate various data values defined by a particular data model element. For more information on the data model elements that use these separator delimiters refer to Section 4.2 SCORM Run-Time Environment Data Model.

2.8. Deprecating the `adlcp:persistState` Attribute

This addendum addresses the deprecation of the `adlcp:persistState` attribute and its associated run-time behavior.

2.8.1. Rationale For Change

This change is being made to resolve potential run-time behavior discrepancies related to the initialization, management, and persistence of Run-Time Data associated with a new learner attempt on a SCO whose associated learning resource has been tagged with an `adlcp:persistState` attribute equal to `true`.

2.8.2. SCORM Update

It has been determined through a review of the use cases related to the `adlcp:persistState` attribute, that the intention of the `adlcp:persistState` attribute has been superseded by the development of various e-learning standards and specifications. The removal of the `adlcp:persistState` attribute will ensure that complimentary emerging technologies and specifications can incorporate a more comprehensive solution to the existing use cases without burdening LMSs and content developers with, potentially complex and confusing, legacy support.

The following updates will be made to the SCORM CAM Version 1.3.1:

- Section 3.4.1.21 <resource> Element will be updated to remove the `adlcp:persistState` attribute definition and declaration.
- Code Illustration 3-19 will be updated to remove the use of the `adlcp:persistState` attribute.
- Table 3.5.3.a will be updated to remove the `adlcp:persistState` attribute from the table.

The following updates will be made to the SCORM RTE Version 1.3.1:

- Section 2.1.1.2. will be removed from the SCORM.
- References to using Persist State will be removed from Sections 4.2.17. and 4.2.23.

The following updates will be made to the ADL Content Packaging Extension XML Schema Definition (XSD) file – `adlcp_v1p3.xsd`:

-
- Upon the formal release of an updated version of the SCORM documentation suite, the adlcp_v1p3.xsd will be updated to remove the attribute declaration of the adlcp:persistState.

SCORM 2004 Clarification/Enhancement Addenda

This page intentionally left blank.

3.1. Ambiguous information defined in the language_type Data Type

This addendum addresses a clarification to the language found in Section 4.1.1.7 Data Types of the SCORM RTE Version 1.3.1. The last paragraph found in the language_type section is ambiguous and is not needed in this section. This section is used to describe the specifics of the data types used by the SCORM RTE.

3.1.1. SCORM Update

The `cmi.learner_preference.language` data model element, found in Section 4.2.13 Learner Preference, will be updated to add the additional information found in Section 4.1.17 to further describe the special case of the `language_type` being an empty characterstring. The Data Model Element Implementation Requirements section will be updated as follows:

From:

Data Model Element Implementation Requirements:

- **Data Type:** language_type (SPM 250)
- **Value Space:** iso-646 [4]
- **Format:** Refer to Section 4.1.1.7: *Data Types* for more information on the requirements for the format of the language_type data type. The default language shall be "" (empty characterstring).

To:

Data Model Element Implementation Requirements:

- **Data Type:** language_type (SPM 250) or empty characterstring
- **Value Space:** iso-646 [4]
- **Format:** Refer to Section 4.1.1.7: *Data Types* for more information on the requirements for the format of the language_type data type. The default language shall be "" (empty characterstring).

3.2. Clarification of Learner Session Initialization Requirements

This addendum addresses a clarification in the behavior of the `cmi.session_time` and `cmi.exit` RTE Data Model elements found in Section 4.2.21 Session Time and Section 4.2.8 Exit respectively, of the SCORM RTE Version 1.3.1. The clarification is centered on the how to handle the `cmi.session_time` and `cmi.exit` value between learner sessions.

3.2.1. SCORM Update

The LMS Behavior Requirements section of the `cmi.session_time` and `cmi.exit` data model element found in Section 4.2.21 Session Time and Section 4.2.8 Exit respectively, will be updated to clarify behavior on handling the `cmi.session_time` value between learner sessions as follows:

Section 4.2.8 Exit

From:

LMS Behavior Requirements:

- The data model element is mandatory and shall be implemented by an LMS as write-only.
- The value is completely controlled by the SCO. The SCO is responsible for setting this value. If the SCO does not set the `cmi.exit` data model element, then the default value (empty characterstring – “”) shall be used. If the LMS receives a request to get the `cmi.exit` value, then the LMS shall adhere to the requirements listed below for API Implementation Requirements.

To:

LMS Behavior Requirements:

- The data model element is mandatory and shall be implemented by an LMS as write-only.
- The value is completely controlled by the SCO. The SCO is responsible for setting this value. If the SCO does not set the `cmi.exit` data model element, then the default value (empty characterstring – “”) shall be used. If the LMS receives a request to get the `cmi.exit` value, then the LMS shall adhere to the requirements listed below for API Implementation Requirements.
- If there are additional learner sessions within a learner attempt, the `cmi.exit` becomes uninitialized (i.e., reinitialized to its default value of (“”) - empty characterstring) at the beginning of each additional learner session within the learner attempt.

Section 4.2.21 Session Time

From:

LMS Behavior Requirements:

- The data model element is mandatory and shall be implemented by an LMS as write-only.
- Since this data model element is implemented by the LMS as write-only, the LMS is not responsible for initializing this data model element. It is the responsibility of the SCO to manage this value. The LMS is only responsible for accepting a SetValue() call to this data model element and perform the accumulation with *cmi.total_time*.
- Since a SCO is not required to set a value for this data model element (not required to keep track of the session time), an LMS shall keep track of session time from the time the LMS launches the SCO. If the SCO reports a different session time, then the LMS shall use the session time as reported by the SCO instead of the session time as measured by the LMS.

To:

LMS Behavior Requirements:

- The data model element is mandatory and shall be implemented by an LMS as write-only.
- Since this data model element is implemented by the LMS as write-only, the LMS is not responsible for initializing this data model element. It is the responsibility of the SCO to manage this value. The LMS is only responsible for accepting a SetValue() call to this data model element and perform the accumulation with *cmi.total_time*.
- Since a SCO is not required to set a value for this data model element (not required to keep track of the session time), an LMS shall keep track of session time from the time the LMS launches the SCO. If the SCO reports a different session time, then the LMS shall use the session time as reported by the SCO instead of the session time as measured by the LMS.
- If there are additional learner sessions within a learner attempt, the *cmi.session_time* becomes uninitialized at the beginning of each additional learner session within the learner attempt.

3.3. Setting the Current Activity to None

This addendum addresses a clarification in the SCORM SN Version 1.3.1 to explicitly define when an LMS should set the Global State Information (SN Version 1.3.1, Section 4.2.1.6) Current Activity element to *None* (or *undefined*).

3.3.1. SCORM Update

The first paragraph of Section 2.3 Starting and Stopping a Sequencing Session will be updated as follows:

From:

A Sequencing Session is the time from when an attempt on the root activity of an Activity Tree begins until that attempt ends. The SCORM Sequencing Behaviors only specify which navigation requests can begin a sequencing session, but they do not specify when or how those navigation requests are triggered. Generally, the LMS will issue a *Start* navigation request in recognition of some system event, e.g., a login, begin course, etc. It is recommended, if the previous sequencing session ended due to a *Suspend All* navigation request, the LMS should issue a *Resume All* navigation request instead of a *Start*.

To:

A Sequencing Session is the time from when an attempt on the root activity of an Activity Tree begins until that attempt ends; outside of the context of a Sequencing Session the Current Activity is considered to be *undefined*. The SCORM Sequencing Behaviors only specify which navigation requests can begin a Sequencing Session, but they do not specify when or how those navigation requests are triggered. Generally, the LMS will issue a *Start* navigation request in recognition of some system event, e.g., a login, begin course, etc. It is recommended, if the previous sequencing session ended due to a *Suspend All* navigation request, the LMS should issue a *Resume All* navigation request instead of a *Start*.

The first set of steps defined in Section 4.3.1 Sequencing Loop will be updated as follows:

From:

Begin Sequencing Session

- (1) The learner initiates access to the LMS (e.g., accesses the system, logs in, etc.) and establishes a context within a particular unit of instruction (e.g., selects a course, a content organization, etc.).
- (2) The LMS initiates a sequencing process by issuing a *Start*, *Resume All*, or *Choice* navigation request.

-
- (3) The Navigation Behavior translates the *Start*, *Resume All*, or *Choice* navigation request into the appropriate sequencing request and processes it. The sequencing session “officially” begins when an activity is identified for delivery – one successful pass through the following Sequencing Loop.

To:

Begin Sequence Session^{**}:

- (1) The learner initiates access to the LMS (e.g., accesses the system, logs in, etc.) and establishes a context within a particular unit of instruction (e.g., selects a course, a content organization, etc.).

^{**} Prior to the beginning of a sequencing session, the Current Activity shall be considered to be *None* (or *undefined*).

- (2) The LMS initiates a sequencing process by issuing a *Start*, *Resume All*, or *Choice* navigation request.
- (3) The Navigation Behavior translates the *Start*, *Resume All*, or *Choice* navigation request into the appropriate sequencing request and processes it. The sequencing session “officially” begins when an activity is identified for delivery – one successful pass through the following Sequencing Loop.

This page intentionally left blank.

Document Revision History

Release Date	Description of Change
09/15/2004	<p data-bbox="621 369 1089 401">Initial Draft. Added the following addenda:</p> <ul data-bbox="621 428 1295 1024" style="list-style-type: none"><li data-bbox="621 428 1295 485">• Addendum 2.1: Handling of Invalid SetValue() Requests for Data Model Element Collections<li data-bbox="621 491 1295 548">• Addendum 2.2: Ambiguous Pseudo-Code in Case #4 of Choice Sequencing Request Process<li data-bbox="621 554 1295 585">• Addendum 2.3: Miscalculation of Traversal Direction<li data-bbox="621 592 1295 648">• Addendum 2.4: Measure Rollup Should not be Applied to Leaf Activities<li data-bbox="621 655 1295 711">• Addendum 2.5: Invalid Default Value defined for the measureSatisfactionIfActive attribute<li data-bbox="621 718 1295 774">• Addendum 2.6: Incorrect SPM for the <dataFromLMS> Element<li data-bbox="621 781 1295 812">• Addendum 2.7: Handling of Reserved Delimiters<li data-bbox="621 819 1295 875">• Addendum 2.8: Deprecating the adlcp:persistState Attribute<li data-bbox="621 882 1295 938">• Addendum 3.1: Ambiguous information defined in the language_type Data Type<li data-bbox="621 945 1295 1001">• Addendum 3.2: Clarification of Learner Session Initialization Requirements<li data-bbox="621 1008 1295 1024">• Addendum 3.3: Setting the Current Activity to None