Advanced Distributed Learning Initiative

Sharable Content Object Reference Model (SCORM™)

Version 1.2

The SCORM Run-Time Environment

October 1, 2001

This page intentionally left blank.

Advanced Distributed Learning
Sharable Content Object Reference Model
Version 1.2
The SCORM Run-Time Environment

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

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

This page intentionally left blank.

Editor
Philip Dodds (ADL)

Key Contributing Editors (ADL)

Ron Ball	Jeff Krinock
William Capone	Lori Morealli
Jeff Falls	Douglas Peterson
Dexter Fletcher	Jonathan Poltrack
Alan Hoberney	Chris Snyder
Paul Jesukiewicz	Schawn Thropp
Kirk Johnson	Bryce Walat
Mary Krauland	Jerry West

Partial List of Contributors:

Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE) (http://www.ariadne-eu.org/)
Erik Duval
Eddy Forte
Florence Haenny
Ken Warkentyne

Aviation Industry CBT (Computer-Based Training) Committee (AICC) (http://www.aicc.org/)
Jack Hyde
Bill McDonald
Anne Montgomery

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

IMS Global Learning Consortium, Inc. (http://www.imsglobal.org/)
Thor Anderson
Steve Griffin
Mark Norton
Ed Walker

(At Large)

Bob Alcorn	Mike Pettit
Lenny Greenberg	Dan Rehak
Chris Moffatt	Tom Rhodes
Boyd Nielsen	Tyde Richards
Claude Ostyn	Roger St. Pierre
Chantal Paquin	Kenny Young

…and many others.

This page intentionally left blank.

Table of Contents

This page intentionally left blank.

SECTION 3
The SCORM^{TM} Run-Time Environment

This page intentionally left blank.

3.1. Run-Time Environment Overview

A goal of the SCORM^{TM} is that learning resources be reusable and interoperable across multiple Learning Management Systems (LMS). For this to be possible, there must be a common way to start learning resources, a common mechanism for learning resources to communicate with an LMS and a predefined language or vocabulary forming the basis of the communication. As illustrated in figure 3.1a, these three aspects of the Run-Time Environment are Launch, Application Program Interface (API) and Data Model.

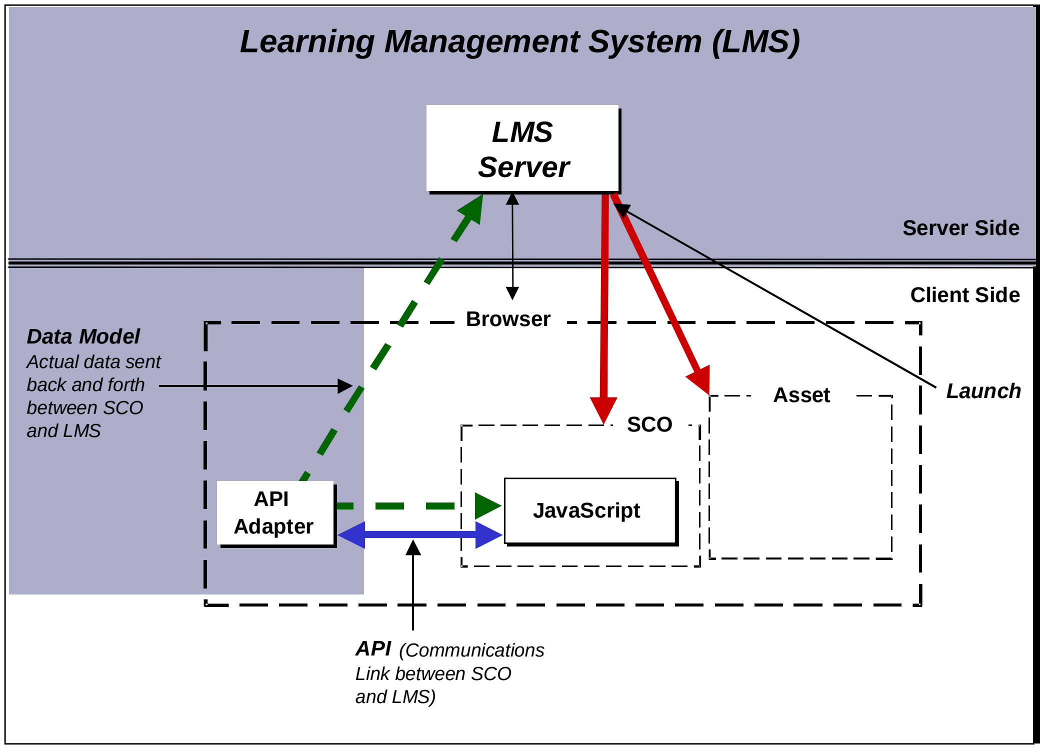

Learning Management System (LMS)
LMS Server
Server Side
Client Side
Browser
Data Model
Actual data sent back and forth between SCO and LMS
Asset
Launch
SCO
API Adapter
JavaScript
API (Communications Link between SCO and LMS)

Figure 3.1a: Launch, API and Data Model as they apply to the SCORM Run-Time Environment.

The *Launch* mechanism defines a common way for LMSs to start Web-based learning resources. This mechanism defines the procedures and responsibilities for the establishment of communication between the delivered learning resource and the LMS. The communication protocols are standardized through the use of a common API.

The *API* is the communication mechanism for informing the LMS of the state of the learning resource (e.g., initialized, finished or in an error condition), and is used for getting and setting data (e.g., score, time limits, etc.) between the LMS and the Sharable Content Object (SCO).

A *Data Model* is a standard set of data elements used to define the information being communicated, such as, the status of the learning resource. In its simplest form, the data model defines elements that both the LMS and SCO are expected to “know” about. The LMS must maintain the state of required data elements across sessions, and the learning content must utilize only these predefined data elements if reuse across multiple systems is to occur.

3.2. Launch

A common launch scheme enables consistency of learning resource delivery behavior across LMSs without specifying the underlying LMS implementation. Note that in this context the term "LMS" is used to describe systems that include the function of managing delivery of learning resources. This launch scheme addresses delivery of Web-enabled learning resources in the form of SCOs and Assets within the context of a learning experience.

3.2.1. Launching of Learning Resources

As described in the Content Aggregation Model, the SCORM Content Model is made up of three components:

- Asset
- SCO
- Content Aggregation

The two SCORM Content Model components that can be launched by an LMS are Assets and SCOs. There are different launching requirements depending on the type of learning resource being launched. The launching mechanism defines the common way for LMSs to start learning resources. The procedures and responsibilities for the establishment of communication between the delivered learning resource and the LMS vary depending on the type of SCORM learning resource being launched.

It is the responsibility of the LMS to manage the sequencing and navigation between learning resources, based on the content structure defined in a content package. LMSs may adaptively determine sequencing based on the fulfillment of defined prerequisites of learning resources. The progression through learning resources that comprise a particular learning experience may be sequential, non-sequential, user-directed, or adaptive, depending on the capabilities of the LMS. At this time the SCORM does not address the standardization of sequencing and navigation between learning resources. This also means that there is no guideline in place for the look and feel of visual components related to sequencing. This will be addressed in a future version of the SCORM.

For example, the LMS may render a menu that allows user directed navigation through a content aggregation. The menu may appear as a series of hyperlinks whose targets contain the corresponding launch locations of the learning resources that appear in the menu.

Alternatively, the LMS may contain, or make use of a server-side delivery mechanism that adaptively determines the sequence in which learning resources are launched based on learner performance. In this case, the server-side component responsible for delivery would directly or indirectly serve the appropriate learning resource based on the specified launch location for the appropriate learning resource.

It is the responsibility of the LMS (or delivery component/service thereof), based on some event, to determine which learning resource is to be launched. The LMS may launch the next learning resource in the sequence defined in the content structure, launch a user selected SCO, or determine which SCO to launch based on student performance in an adaptive fashion. Upon determining the appropriate learning resource to launch, the LMS uses the URL defined by the learning resource's launch location, defined in the content package, to navigate to, or replace the currently displayed learning resource with the learning resource found at the launch location.

The LMS may implement the launch in any manner desired and may delegate the actual launch responsibility to the client or server portion of the LMS as needed. The actual launch must be accomplished using the HTTP protocol. Ultimately, the learning resource identified by the launch location in a content package is launched and delivered to the client browser.

3.2.1.1. Asset

For learning resources that represent Assets, the SCORM launch model only requires that an LMS launch the Asset using the HTTP protocol. Since an Asset does not need to communicate, using the API and Data Model, back to the LMS there is no need for an Asset to search for the API Adapter provided by an LMS.

3.2.1.2. Sharable Content Object (SCO)

For learning resources that represent SCOs, the SCORM launch model requires that an LMS only launch one SCO at a time and that only one SCO is active at a time. The launch model also requires that only LMSs may launch SCOs. SCOs may not launch other SCOs.

The LMS must launch the SCO in a browser window that is a child window or a child frame of the LMS window that exposes the API Adapter as a Document Object Model (DOM)[27] Object. The API Adapter must be provided by the LMS.

It is the responsibility of the SCO to recursively search the parent and/or opener window hierarchy until the API Adapter is found. Once the API Adapter has been found the SCO may initiate communication with the LMS.

3.3. Application Program Interface (API)

3.3.1. API Overview

The SCORM is based directly on the run-time environment functionality defined in AICC's CMI001 Guidelines for Interoperability[4] document. ADL collaborated with AICC members and participants to develop a common *Launch* and *API* specification and to adopt Web-based data elements. The following sections provide an overview of the key elements of the AICC API specification as they relate to the SCORM.

3.3.2. Description of the SCO to LMS Communication API

The use of a common API fulfills many of the SCORM's high-level requirements for interoperability and reuse. It provides a standardized way for SCOs to communicate with LMSs, yet it shields the particular communication implementation from the content developer. In its simplest terms, an API is merely a set of predefined functions that the SCO can rely on being available. An API hides implementation details from SCOs and thus promotes reuse and interoperability. An API Adapter is a piece of functional software that implements and exposes the functions of the API. How the insides of an API Adapter are implemented should not matter to content developers provided they use the same public interface. The LMS need only provide an API Adapter that implements the functionality of the API and exposes its interface to the client SCO.

A key aspect of the API is that it is a communication mechanism that allows the SCO to communicate with the LMS. It is assumed that once the SCO is launched it can then "get" and "set" information with an LMS. All communication between the API Adapter and the SCO is initiated by the SCO. There is currently no supported mechanism for LMSs to initiate calls to functions implemented by a SCO. The functions of the API Adapter object are threefold:

- *Execution State*
  Two of the API functions, *LMSInitialize("")* and *LMSFinish("")*, handle execution state.
- *State Management*
  The API has three functions that are used to handle errors. These three API functions are: *LMSGetLastError()*, *LMSGetErrorString(errornumber)* and *LMSGetDiagnostic(parameter).*
- *Data Transfer*
  The remaining three API functions are used to transfer data to and from an LMS: *LMSGetValue(data model element), LMSSetValue(data model element, value)* and *LMSCommit("").* Note that the API is designed to get and set data

values that are separately defined by an external data model. The AICC specification defines one such data model, called “cmi”. Other data models could be developed and used with this API as well.

3.3.2.1. SCO To LMS Communications API Details

The following table defines the SCO to LMS Communications API in detail.

Execution State	

LMSInitialize	**Description:** This function indicates to the API Adapter that the SCO is going to communicate with the LMS. It allows the LMS to handle LMS specific initialization issues. It is a requirement of the SCO that it call this function before calling any other API functions.
	
	Syntax: LMSInitialize(parameter)
	
	Parameter: "" An empty string must be passed for conformance to this standard. Values other than "" are reserved for future extensions.
	
	Return Value: String representing a boolean.
	• "true" result indicates that the LMSInitialize("") was successful
	• "false" result indicates that the LMSInitialize("") was unsuccessful
	If a return value of "false" is returned, then this signifies to the SCO that the LMS is in an unknown state and that any additional API calls will not be processed by the LMS.
	
	Example:
	`var result = LMSInitialize("")`
	`if (result == "false")`
	`{`
	`   // Do some error handling`
	`}`
	`else`
	`{`
	`   // continue with the execution of the SCO`
	`}`
	
	The SCO tells the API Adapter that the content wants to establish communication with the LMS. A typical return value is "true".

LMSFinish	**Description:** The SCO must call this when it has determined that it no longer needs to communicate with the LMS, if it successfully called LMSInitialize at any previous point. This call signifies two things: 1. The SCO can be assured that any data set using LMSSetValue() calls has been persisted by the LMS. 2. The SCO has finished communicating with the LMS. **Syntax:** LMSFinish(parameter) **Parameter:** "" An empty string must be passed for conformance to this standard. Values other than "" are reserved for future extensions. **Return Value:** String representing a boolean. • "true" result indicates that the LMSFinish("") was successful • "false" result indicates that the LMSFinish("") was unsuccessful

	If a return value of "true" is returned, then the SCO may no longer call any other API functions. If a return value of "false" is returned, then this signifies to the SCO that the LMS is in an unknown state and that any additional API calls may or may not be processed by the LMS. **Examples:** `var result = LMSFinish("");`
Data Transfer	
LMSGetValue	**Description:** This function allows the SCO to obtain information from the LMS. It is used to determine: • Values for various categories (groups) and elements in the data model • The version of the data model supported • Whether a specific category or element is supported • The number of items currently in an array or list of elements The complete data element name and/or keywords are provided as a parameter. The current value of the requested data model parameter is returned. Only one value -- always a string -- is returned for each call. **Syntax:** LMSGetValue(parameter) **Parameter:** datamodel.group.element Returns the value of the named element datamodel._version The _version keyword is used to determine the version of the data model supported by the LMS. datamodel.element._count The _count keyword is used to determine the number of elements currently in an array. The count is the total number of elements in the array, not the index number of the last position in the array. datamodel.element._children The _children keyword is used to determine all of the elements in a group or category that are supported by the LMS. **Return Value:** All return values are strings. LMSGetValue(datamodel.group.element) The return value is a string representing the current value of the requested element or group. LMSGetValue(datamodel._version) The return value is a string representing the version of the data model supported by the LMS. LMSGetValue(datamodel.group._children) The return value is a comma-separated list of all of the element names in the specified group or category that are supported by the LMS. If an element has no children, but is supported, an empty string ("") is returned. An empty string ("") is also returned if an element is not supported. A subsequent request for last error [LMSGetLastError()] can determine if the element is not supported. The error "401 Not implemented error" indicates the element is not supported. LMSGetValue(datamodel.group._count) The return value is an integer that indicates the number of items

	currently in an element list or array.
	
	Examples:
	`var value = LMSGetValue("cmi.core.student_name")`
	A typical return value might be "Hyde, Jackson".
	
	`var value = LMSGetValue("cmi.core.lesson_status")`
	A typical return value might be "incomplete".
	
	`var value = LMSGetValue(“cmi._version”)`
	The current draft standard for the IEEE document defining the CMI data model is entitled *Draft Standard for Computer Managed Instruction,* and has an ID of P1484.11.2 and a version number of 3.4. This call returns the version number of that IEEE document which is 3.4.
	
	`var value = LMSGetValue("cmi.student_preferences._children")`
	
	This is a request for category support information. One typical return value would be, "audio,speed,text". If there is an empty string (“”) returned, student_preferences are not supported. An additional API call to determine the last error could verify this.

LMSSetValue	**Description:** This function allows the SCO to send information to the LMS. The API Adapter may be designed to immediately forward the information to the LMS, or it may be designed to forward information based on some other approach. This function is used to set the current values for various categories (groups) and elements in the data model. The data element name and its group are provided as a parameter. The newly desired value of the data element is included as the second parameter in the call. Only one value is sent with each call. **Syntax:** LMSSetValue(parameter, value) **Parameter:** This is the name of a fully qualified element defined in the data model. The argument is case sensitive. The argument is a string enclosed in quotes. The following represents some forms this parameter may take. datamodel.element This is the name of a category or group defined in the Data Model. An example is "cmi.comments". datamodel.group.element This is the name of an element defined in the Data Model. An example is "cmi.core.lesson_status". datamodel.group.n.element The value of the sub-element in the nth-1 member of the element array (zero-based indexing is used). **Value:** This is a string that must be convertible to the data type defined in this standard for the data model element identified in the first parameter. **Return Value:** String representing a boolean • "true" result indicates that the LMSSetValue() was successful • "false" result indicates that the LMSSetValue() was unsuccessful **Examples:** `var result = LMSSetValue("cmi.core.score.raw","95");`

	Sets the cmi.core.score.raw to a value of "95" A subsequent call to LMSGetValue("cmi.core.score.raw") must return "95"
LMSCommit	**Description:** If the API Adapter is caching values received from the SCO via an LMSSetValue(), this call requires that any values not yet persisted by the LMS be persisted. In some implementations, the API Adapter may persist set values as soon as they are received, and not cache them on the client. In such implementations, this API call is redundant and would result in no additional action from the API Adapter. This call ensures to the SCO that the data sent, via an LMSSetValue() call, will be persisted by the LMS upon completion of the LMSCommit(). **Syntax:** LMSCommit(parameter) **Parameter:** `""`. An empty string must be passed for conformance to this standard. Values other than `""` are reserved for future extensions. **Return Value:** String representing a boolean • "true" result indicates that the LMSCommit(`""`) was successful • "false" result indicates that the LMSCommit(`""`) was unsuccessful If a return value of "false" is returned, then this signifies to the SCO that the LMS is in an unknown state and that any additional API calls may or may not be processed by the LMS. **Examples:** `var result = LMSCommit("");` Requires that any cached values, previously set via SCO calls to LMSSetValue(), that have not been persisted by the LMS be persisted.
State Management	
LMSGetLastError	**Description:** The SCO must have a way of assessing whether or not any given API call was successful, and if it was not successful, what went wrong. This function returns an error status code resulting from the previous API call. Each time an API function is called (with the exception of this one, LMSGetErrorString, and LMSGetDiagnostic -- the error functions), the error code is reset. The SCO may call the error functions any number of times to retrieve the error code, and the code cannot change until the next API call is made. **Syntax:** LMSGetLastError() **Parameter:** None **Return Value:** The return values are Strings that can be converted to integer numbers that identify errors falling into the following categories: 100's General errors 200's Syntax errors 300's LMS errors 400's Data model errors The following codes are available for error messages: 0 No error 101 General exception 201 Invalid argument error

	202 Element cannot have children 203 Element not an array – cannot have count 301 Not initialized 401 Not implemented error 402 Invalid set value, element is a keyword 403 Element is read only 404 Element is write only 405 Incorrect Data Type Additional codes TBD **Examples:** `var errorCode = LMSGetLastError();`
LMSGetErrorString	**Description:** This function enables the content to obtain a textual description of the error represented by the error code number. **Syntax:** LMSGetErrorString(errornumber) **Parameter:** An integer number representing an error code. **Return Value:** A string that represents the verbal description of an error. **Examples:** `var errorString = LMSGetErrorString("403");` errorString should contain "Element is read only"
LMSGetDiagnostic	**Description:** This function enables vendor-specific error descriptions to be developed and accessed by the content. These would normally provide additional detail regarding the error. **Syntax:** LMSGetDiagnostic(parameter) **Parameter:** The parameter may take one of two forms. • An integer number representing an error code. This requests additional information on the listed error code. • “”. An empty string. This requests additional information on the last error that occurred. **Return Value:** The return value is a string that represents any vendor-desired additional information relating to either the requested error or the last error. **Examples:** `var moreInfo = LMSGetDiagnostic("403");` moreInfo could contain more vendor specific information on the “Element is read only” error

3.3.2.2. SCO To LMS Communications API Adapter State Transition

The SCO to LMS Communications API Adapter traverses through several states (Figure 3.3.2.2a), for a given instance of a SCO, at run-time. The states of the API Adapter specify the responses of the API Adapter to specific input events. During each of these states there are different activities that a SCO may go through. The states encountered by the API are: Not Initialized, Initialized and Finished.

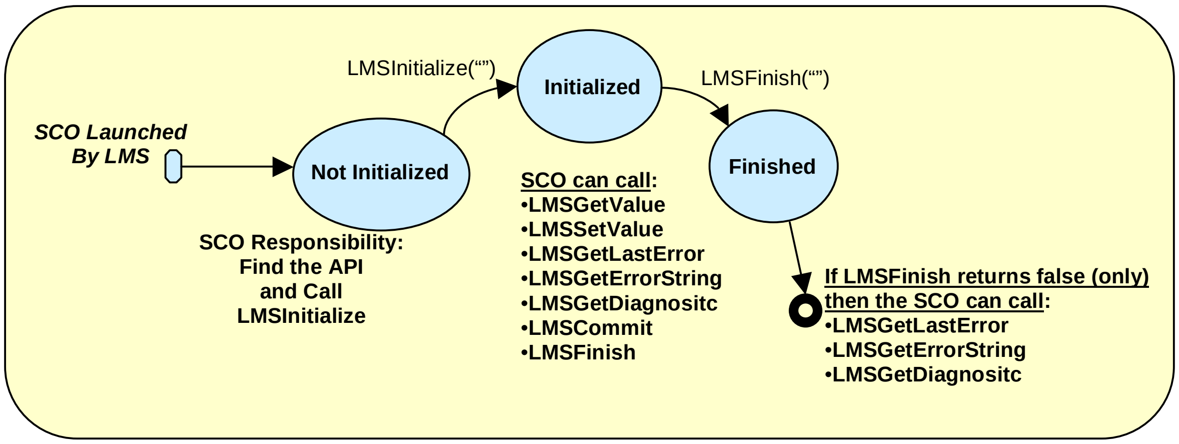

SCO Launched By LMS
Not Initialized
LMSInitialize(“”)
Initialized
LMSFinish(“”)
Finished
SCO Responsibility: Find the API and Call LMSInitialize
SCO can call: •LMSGetValue •LMSSetValue •LMSGetLastError •LMSGetErrorString •LMSGetDiagnositc •LMSCommit •LMSFinish
If LMSFinish returns false (only) then the SCO can call: •LMSGetLastError •LMSGetErrorString •LMSGetDiagnositc

Figure 3.3.2.2a: API Adatper State Transitions

Not Initialized: This describes the state that the SCO is in between the actual launching of the SCO and before the LMSInitialize(“”) API function is invoked by the SCO. During this state it is the SCOs responsibility to find the API Adapter provided by the LMS. Once the API Adapter is found by the SCO, the SCO is permitted to invoke the following API function calls:

- LMSInitialize(“”)
- LMSGetLastError()
- LMSGetErrorString()
- LMSGetDiagnostic()

Initialized: This describes the state that the SCO is in once the LMSInitialize(“”) API function call is invoked and before the LMSFinish(“”) API function call is invoked. If the SCO is in the Initialized state it is permitted to invoke all of the API function calls except for LMSInitialize(“”).

Finished: This describes the state that the SCO is in once the LMSFinish(“”) API function call is invoked. If the API Adapter returns “false” to the SCOs attempt to call LMSFinish(“”) the SCO is permitted to invoke the following API function calls:

- LMSGetLastError()
- LMSGetErrorString()

- LMSGetDiagnostic()

If the LMS API Adapter returns “false” there is no guarantee that the LMS API Adapter will respond appropriately to any API function call.

3.3.3. API Error Code Usage

The SCO must have a way of assessing whether or not any given API function call was successful, and if it was not successful, what went wrong. The *LMSGetLastError()* function returns an error code that can be used to determine the type of error raised by the previous API function call.

Syntax: LMSGetLastError()

Parameters: None

Return Type: String – values can be converted into integer numbers that are identified in the following table.

Code	Description	Usage
“0”	No error	No errors encountered. Successful API call.
“101”	General Exception	Used to indicate general exceptions.
“201”	Invalid argument error	To be used when there is a call to a SCORM Run-Time Environment data model element that does not exist. To be used when an invalid argument is passed via the API
	Ex. *LMSGetValue(“cmi.core.zip_code”)* “cmi.core.zip_code” is not a valid CMI Data Model element. Ex. *LMSInitialize(“init”)* The LMSInitialize(“”) expects an empty string argument.	
“202”	Element cannot have children	To be used when LMSGetValue() is called on any data model category or element that does not support _children.
	Ex. *LMSGetValue(“cmi.student_id._children”)*	
“203”	Element not an array. Cannot have count.	To be used when an LMSGetValue() is called on any data model category or element that does not support _count.
	Ex. *LMSGetValue(“cmi.core._count”)*	
“301”	Not initialized	To be used when there is a call to any API function call before LMSInitialize(“”) is called.

"401"	Not implemented error	To be used when a call is made to a Data Model element that is not supported by the LMS or if another Data Model is used outside of the SCORM Run-Time Environment Data Model.
	Ex. *LMSGetValue("cmi.objectives.0.id")* The "cmi.objectives.0.id" element is an optional element. If the LMS does not support this element then the LMS should return an empty string ("") and set the error code to 401. Ex. *LMSGetValue("xyz.score.result")* The "xyz.score.result" is not a valid SCORM Run-Time Environment Data Model element. Since it is not a valid data model element ("xyz" vs. "cmi"), the LMS should set the error code to "401" and return an empty string.	
"402"	Invalid set value, element is a keyword	To be used when an LMSSetValue() call is invoked on a keyword.
	Ex. *LMSSetValue("cmi.core._children","student_id,student_name")* "cmi.core._children" is a keyword.	
"403"	Element is read only.	To be used when an LMSSetValue() call is invoked on an element that is read only.
	Ex. *LMSSetValue("cmi.core.student_id","JoeStudent")* "cmi.core.student_id" is a read only CMI element	
"404"	Element is write only	To be used when an LMSGetValue() call is invoked on an element that is write only
	Ex. *LMSGetValue("cmi.core.exit")* "cmi.core.exit" is a write only CMI element	
"405"	Incorrect Data Type	To be used when an attempt is made to set an element with the incorrect data type.
	Ex. *LMSSetValue("cmi.core.score.raw","eighty five")* "eighty five" is not the correct data type for the "cmi.core.score.raw". The correct data type needs to be a CMIDecimal ("85") or a CMIBlank (""). Ex. *LMSSetValue("cmi.core.lesson_status","Not Attempted")* "Not Attempted" is not a valid vocabulary member of the cmi.core.lesson_status element. Vocabularies are case sensitive and must match identically.	

3.3.4. API General Rules

The following list summarizes general usage rules for the API:

- The function names are all case sensitive, and must always be expressed exactly as shown.
- The function parameters or arguments are case sensitive. All parameters are lower case.
- Each call to an API function, other than the error handling functions, resets the error code.

3.3.5. LMS Responsibility

3.3.5.1. API Adapter

The SCORM requires that an LMS supply an API Adapter that implements the required API functionality described in the previous section. This adapter must shield SCOs from the particular adapter implementation details so that the SCOs need not have any knowledge of the underlying communication infrastructure, and instead rely solely on the existence of a standardized LMS Application Program Interface. The requirements for using the API Adapter are as follows:

- The LMS must launch the SCO in a browser window that is a child window or a child frame of the LMS window that contains the API Adapter.
- The API Adapter must be provided by the LMS.
- The only supported mechanism for API interaction from SCOs is through ECMAScript (JavaScript) calls.
- The API Adapter must be accessible via the DOM[27] as an object named "API".

As an example, an API Adapter might be implemented as a Java applet that may have a signature like this:

```
public class API extends Applet
{
  public String LMSInitialize( String parameter )
  { . . .}
  public String LMSGetValue( String element )
  { . . }
  public String LMSSetValue( String element, String value )
  {. . .}
  public String LMSCommit( String parameter )
  {. . .}
  public String LMSFinish( String parameter )
  {. . .}
  public String LMSGetLastError()
  {. . .}
  public String LMSGetErrorString( String errorCode )
  {. . .}
  public String LMSGetDiagnostic( String errorCode )
  {. . .}
}
```

Note that an API Adapter can be implemented in other programming languages such as C++ and loaded, for example, as a browser plug-in. The API Adapter implementation is expected to be LMS specific; the above code fragment is only an example approach. There are many ways to implement an LMS API Adapter.**SCO Responsibility**

3.3.6.1. Find API

It is the responsibility of the SCO to, at a minimum, issue LMSInitialize("") and LMSFinish("") API calls. In order to do this, the content must be able to locate the API

Adapter that is presented by the LMS. It is the responsibility of the LMS to place an API Adapter in the DOM window hierarchy so that the SCO can recursively search the parent and/or opener window hierarchy to find the API. It is the responsibility of the content to find and establish communication with the LMS's API Adapter. How the SCO chooses to do this is not mandated by the SCORM.

The following code example represents an algorithm that the SCO could use to locate the LMS's API Adapter. The use of these functions is not a requirement imposed by the SCORM and the SCO may use other approaches to locate the LMS's API Adapter.

```
<SCRIPT LANGUAGE=JAVASCRIPT >
var findAPITries = 0;

function findAPI(win)
{
   // Check to see if the window (win) contains the API
   // if the window (win) does not contain the API and
   // the window (win) has a parent window and the parent window
   // is not the same as the window (win)
   while ( (win.API == null) &&
           (win.parent != null) &&
           (win.parent != win) )
   {
      // increment the number of findAPITries
      findAPITries++;

      // Note: 7 is an arbitrary number, but should be more than sufficient
      if (findAPITries > 7)
      {
         alert("Error finding API -- too deeply nested.");
         return null;
      }

      // set the variable that represents the window being
      // being searched to be the parent of the current window
      // then search for the API again
      win = win.parent;
   }
   return win.API;
}

function getAPI()
{
   // start by looking for the API in the current window
   var theAPI = findAPI(window);

   // if the API is null (could not be found in the current window)
   // and the current window has an opener window
   if ( (theAPI == null) &&
        (window.opener != null) &&
        (typeof(window.opener) != "undefined") )
   {
      // try to find the API in the current window's opener
      theAPI = findAPI(window.opener);
   }
   // if the API has not been found
   if (theAPI == null)
   {
      // Alert the user that the API Adapter could not be found
      alert("Unable to find an API adapter");
   }
   return theAPI;
}
```

3.4. Data Model

3.4.1. Data Model Overview

The purpose of establishing a common data model is to make sure that a defined set of information about SCOs can be tracked by different LMS environments. If, for example, it is determined that tracking a student's score is a general requirement, then it is necessary to establish a common way for content to report scores to LMS environments. If SCOs use their own unique scoring representations, learning management systems may not know how to receive, store or process the information.

There are a number of data models under development in various communities and standards organizations. These draft data model specifications attempt to functionally group information sets to be exchanged between SCOs and LMS environments. Examples include: student profile information, question and test interactions, state information, assessment, etc. As of the release of this version of the SCORM, these draft data model sets are still under development and have not been widely implemented or tested.

3.4.1.1. The SCORM Run-Time Environment Data Model

The data model in this section is defined as the SCORM Run-Time Environment Data Model derived directly from the AICC CMI Data Model described in the AICC CMI Guidelines for Interoperability[4]. The AICC CMI Data Model was chosen for inclusion in the SCORM since it is well defined and has been implemented in the past. It is expected that in the future new data model sets will be adopted and incorporated into the SCORM. It is assumed that the data model elements defined in this version of the SCORM will map to new data model sets when defined and adopted. A mapping from the current data model to new elements should provide a relatively smooth migration path to future data models.

To identify the data model in use, all of the names of the elements described in this section start with "cmi". This signals implementers that these elements are part of the AICC CMI Data Model. Alternative data models, as developed, will start with a different designation (e.g., adl.*elementName* instead of cmi.*elementName*).

During the test and evaluation phase of the SCORM, members of the AICC and IEEE decided to substantially reduce the number of elements in the AICC CMI Data Model. This was done both to ease the transition to new data models under development, and because many of the elements removed had not actually been implemented by most implementers. A list of elements removed from the AICC CMI Data Model that appeared in the SCORM Version 1.0 are summarized in Appendix C. The SCORM Run-Time Environment Data Model contained herein matches the reduced AICC CMI Data Model set.

3.4.1.2. The SCORM Run-Time Environment Data Model General Rules

The following list summarizes general usage rules for the data model:

- The first symbol in the data element name identifies the data model. For example, "cmi" indicates the AICC CMI Data Model. This expands the functionality of the API by allowing the same API to be used with other data models;
- There are three reserved keywords. These are all lower case and proceeded by an underscore.
  - _version: keyword used to determine the version of the data model supported by the LMS.
  - _children: keyword used to determine which data model elements are supported by the LMS.
  - _count: keyword used to determine the number of elements currently in a list;
- All arrays are 0 based arrays. Items should be placed in the arrays in a sequential manner;
- The data model names are case sensitive; and
- The data model is implemented on a SCO by SCO basis. One SCO cannot access another SCO's data elements.

3.4.2. Data Model Elements

The data model elements are broken up into two categories: mandatory and optional. The AICC CMI001 Guidelines for Interoperability[4] document specifies which elements require mandatory implementation by an LMS, and which are optional.

All mandatory data model elements must be supported by the LMS. LMS environments may implement support for all or some of the optional data model elements**.**

All data elements are optional for use by SCOs. SCOs are required only to use the API functions LMSInitialize("") and LMSFinish(""); they are not required to use LMSSetValue() or LMSGetValue(). SCOs may be very, very small and not be designed to be tracked in detail. But if they are to be tracked, they must conform to a common data model for reusability across multiple LMS environments.

3.4.3. Handling Lists

There are several data elements that appear in a list or an array. An example of this would be objectives. There may be more than one objective covered in the content, and a student may be allowed to experience an objective more than once.

To get or set values in a list, the index number is used. The only time an index number may be omitted is when there is only one member in a list. Index numbering starts at 0. If a value is to be appended to the list, the SCO must know the last index number used.

All new array elements shall be added sequentially. The SCO shall not skip array numbers or leave empty array elements when constructing a list of array values. The _count keyword can be used to determine the current number of records in the list. For instance, to determine the number of objective records currently stored, the following API call would be used:

```
   var numOfObjectives = LMSGetValue("cmi.objective._count");
```

If the SCO does not know the count of the objective records, it can begin the current student count with 0. This would overwrite any information about objectives currently stored in the first index position. Overwriting or appending is a decision that is made by the SCO author when he creates the SCO.

Elements in a list are referred to with a dot-number notation (represented by .n). For instance the value of the status element in the first objective in a SCO would be referred to as "cmi.objective.0.status". The status element in the fourth objective would be referred to as "cmi.objective.3.status".

3.4.4. The SCORM Run-Time Environment Data Model

cmi.core Information required to be furnished by all LMS systems. What all SCOs may depend upon at start up. Children of cmi.core: student_id, student_name, lesson_location, credit, lesson_status, entry, score, total_time, lesson_mode, exit, session_time	
cmi.core._children	
Supported API calls: LMSGetValue() **LMS Mandatory:** Yes **Data Type:** CMIString255 **SCO Accessibility:** Read Only	**Definition:** The _children keyword is used to determine all of the elements in the core category that are supported by the LMS. If an element has no children, but is supported, an empty string is returned. If an element is not supported, an empty string is returned. A subsequent request for last error can verify that the element is not supported. **Usage:** To determine which cmi.core data elements are supported by the LMS. **Format:** The return value is a comma separated list of all of the element names in the core category that are supported by the LMS. **LMS Behavior:** • **Initialization:** The set of supported children for this group. So that on an LMSGetValue() request, the appropriate list of supported children is returned. • **LMSGetValue():** LMS returns a comma separated list of supported elements o **Example API call:** LMSGetValue("cmi.core._children") o **Example Return Values:** "student_id,student_name,lesson_location,credit,lesson_status, entry, score,total_time, exit,session_time" o **Error Code:** ▪ **401 - Not implemented error**. If the cmi.core._children element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE: The cmi.core._children element <u>must be</u> supported by LMS since the element is mandatory.* • **LMSSetValue():** LMS should set an error code according to the following:

	o **Error Code:**
	▪ **402 - Invalid set value, element is a keyword**. If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory*
	
	SCO Usage Example:
	SCOs can use the cmi.core._children request to determine if a certain element is implemented by the LMS:
	
	var coreChildren = LMSGetValue("cmi.core._children");
	if (coreChildren.indexOf("student_name") != -1)
	{
	  studentName = LMSGetValue("cmi.core.student_name");
	}

cmi.core.student_id

Supported API calls: LMSGetValue() **LMS Mandatory:** Yes **Data Type:** CMIIdentifier **SCO Accessibility:** Read Only	**Definition:** Unique alpha-numeric code / identifier that refers to a single user of the LMS system. **Usage:** Used to uniquely identify a student. **Format:** Up to 255 alpha-numeric characters with no spaces. Dashes (or hyphen) and the underscore are legal. Periods are illegal. Case insensitive. **LMS Behavior:** • **Initialization:** LMS is responsible, based on student registration. • **LMSGetValue():** Returns the current value stored by the LMS for the student. o **Example Return Values:** "Joe_Student1" "JS-2000" "joe2000-3" o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.* • **LMSSetValue():** LMS should set an error code according to the following: o **Error Code:** ▪ **403 Element is read only.** If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403. ▪ **401 - Not implemented error** If this element is not supported an errorcode is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.* **SCO Usage Example:** The SCO might want to display the student ID when it is launched: var coreStudentID = LMSGetValue("cmi.core.student_id");

cmi.core.student_name

Supported API calls: LMSGetValue()	**Definition:** Normally, the official name used for the student on the course roster. A complete name, not just a first name.

LMS Mandatory: Yes **Data Type:** CMIString255 **SCO Accessibility:** Read Only	**Usage:** Used to represent the students official name. **Format:** Last name, first name and middle initial. Last name and first name are separated by a comma. Spaces in the name must by honored. **LMS Behavior:** • **Initialization:** LMS is responsible, based on student registration. • **LMSGetValue():** Returns the current value stored by the LMS o **Example Return Values:** "Student, Joseph A." "Student, Mike A. Jr." o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory* • **LMSSetValue():** LMS should set an error code according to the following: o **Error Code:** ▪ **403 - Element is read only.** If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403. ▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory* **SCO Usage Example:** The SCO might want to display the student name on launch of the SCO: var coreStudentName = LMSGetValue(“cmi.core.student_name”);
cmi.core.lesson_location	
Supported API calls: LMSGetValue() LMSSetValue() **LMS Mandatory:** Yes **Data Type:** CMIString255 **SCO Accessibility:** Read / Write	**Definition:** This corresponds to the SCO exit point passed to the LMS system the last time the student experienced the SCO. This provides one mechanism to let the student return to a SCO at the same place he left it earlier. In other words, this element can identify the student's exit point and that exit point can be used by the SCO as an entry point the next time the student runs the SCO. **Usage:** This element defines where the student last was inside the SCO. The element could be used by the SCO to store off a "bookmark" during the session. If the SCO is suspended, and then is re-entered later, the lesson_location could be used by the SCO to send the student back into the SCO where they left off. **Format:** Implementation dependent. The LMS system simply holds this data and then returns it to the SCO when the student is re-entering it, if the SCO asks for it. Whatever the SCO passes back to the LMS system is returned. The format matches whatever the SCO expects -- the format is created by the SCO. The first time a student enters the SCO, or if there is no preferred starting point, lesson_location may equal an empty string (""). **LMS Behavior:** • **Initialization:** LMS should set this to be an empty string. A SCO may optionally set this value and then retrieve the data on re-entry into the SCO. • **LMSGetValue():** Returns the current value stored by the LMS o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.* • **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element. o **Error Code:** ▪ **405 – Incorrect Data Type.** If the element is supported

(element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory*.
- **Example Return/Set Values:** SCO implementation dependent

SCO Usage Example:
The SCO can use the lesson_location as a "bookmark"
- On launch of the SCO, the SCO may position the student where the student left off in the SCO during the previous attempt at the SCO.
- On exit of the SCO, the SCO may set the position so that the next time the student enters the SCO the position can be retrieved.

```
// Example SCO that is written using JavaScript and HTML with anchors

// This function could exist in a function call onLoad() that gets invoked when the HTML
// page is loaded

var coreSCOLocation =  LMSGetValue("cmi.core.lesson_location");
if (LMSGetLastError() == "0")
{
   // coreSCOLocation contains an anchor name defined in the
   // HTML page

   // Start the SCO off where the student left off
   window.location.hash = coreSCOLocation;
}
else
{
   // Error processing
}
```

cmi.core.credit

Supported API calls:
LMSGetValue()

LMS Mandatory: Yes

Data Type: CMIVocabulary (Credit)
"credit"
"no-credit"

SCO Accessibility:
Read Only

Definition: Indicates whether the student is being credited by the LMS system based on performance (pass/fail and score) in this SCO.

Usage: Used by the LMS system to indicate whether or not the student is taking the SCO for credit.

cmi.core.credit is used in conjunction with lesson_mode. See cmi.core.lesson_mode for more detail. Cmi.core.credit is also used in the determination of lesson_status. See cmi.core.lesson_status for more detail.

Format: A set vocabulary phrase. Two possible vocabulary values:
- **"credit"**. This means that the student is taking the SCO for credit. The LMS system is telling the SCO that if the SCO sends data to the LMS system, the LMS system will credit it to the student.
- **"no-credit"**. This means that the student is taking the SCO for no-credit. His current credit, if any (for instance a score of 80 and status of passed) will not be changed by his performance in this SCO. The LMS system is telling the SCO that if the SCO sends data to the LMS system it will not change the student's accreditation.

LMS Behavior:
- **Initialization:** LMS is responsible for determining whether or not the student is taking the course for credit or no-credit. **NOTE:** Right now there is no SCORM defined way to signify that learning content can be taken for credit or no-credit. Implementation therefore will be LMS specific.
- **LMSGetValue():** Returns the current value stored by the LMS
    - **Example Return Values:**
      "no-credit"
      "credit"
    - **Error Code:**

- **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory.*
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **403 - Element is read only.** If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory.*

SCO Usage Example:
The SCO might use the value returned from the LMS to determine what is displayed in the browser. There may be different content in the SCO that is displayed if the course is being taken for credit versus no-credit.

```
var creditFlag = LMSGetValue("cmi.core.credit")
if (creditFlag == "credit")
{
    // Student is taking course for credit handle appropriately
}
else
{
   // Student is taking course for no-credit, handle appropriately
}
```

cmi.core.lesson_status

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: Yes

Data Type: CMIVocabulary (Status)
passed
completed
failed
incomplete
browsed
not attempted

SCO Accessibility:
Read / Write

Definition: This is the current student status as determined by the LMS system. Six status values are possible.

Usage:
Normally the SCO determines its own status and passes it to the LMS.

1) If cmi.core.credit is set to “credit” and there is a mastery score in the manifest (adlcp:masteryscore), the LMS can change the status to either passed or failed depending on the student's score compared to the mastery score.
2) If there is no mastery score in the manifest (adlcp:masteryscore), the LMS cannot override SCO determined status.
3) If the student is taking the SCO for no-credit, there is no change to the lesson_status, with one exception. If the lesson_mode is "browse", the lesson_status may change to "browsed" even if the cmi.core.credit is set to no-credit.

On re-entry into the SCO, the LMS may change the status to either passed, failed, or browsed. Passed or failed based on criteria (defined in the manifest) for mastery of the SCO. browsed, if the SCO was launched on its initial attempt with a lesson_mode of “browse”.

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

- **passed**: Necessary number of objectives in the SCO were mastered, or the necessary score was achieved. Student is considered to have completed the SCO and passed.
- **completed**: The SCO may or may not be passed, but all the elements in the SCO were experienced by the student. The student is considered to have completed the SCO. For instance, passing may depend on a certain score known to the LMS system. The SCO knows the raw score but not whether that raw score was high enough to pass.
- **failed**: The SCO was not passed. All the SCO elements may or may not have been completed by the student. The student is considered to have completed the SCO and failed.
- **incomplete**: The SCO was begun but not finished.
- **browsed**: The student launched the SCO with a LMS mode of Browse on the initial attempt.
- **not attempted**: Incomplete implies that the student made an attempt to perform the SCO, but for some reason was unable to finish it. Not attempted means that the student did not even begin the SCO. Maybe he just read the

table of contents, or SCO abstract and decided he was not ready. Any algorithm within the SCO may be used to determine when the SCO moves from "not attempted" to "incomplete".

LMS Behavior:

- **Initialization:** If it is the student's first attempt at the SCO the lesson_status is set to **not attempted.** The LMS is responsible for setting the initial value to "not attempted".
    - Additional Behavior Requirements: If a SCO sets the cmi.core.lesson_status then there is no problem. However, the SCORM does not force the SCO to set the cmi.core.lesson_status. There is some additional requirements that must be adhered to successfully handle these cases:
        - Upon initial launch the LMS should set the cmi.core.lesson_status to "not attempted".
        - Upon receiving the LMSFinish() call or the user navigates away, the LMS should set the cmi.core.lesson_status for the SCO to "completed".
        - After setting the cmi.core.lesson_status to "completed", the LMS should now check to see if a Mastery Score has been specified in the cmi.student_data.mastery_score, if supported, or the manifest that the SCO is a member of. If a Mastery Score is provided and the SCO did set the cmi.core.score.raw, the LMS shall compare the cmi.core.score.raw to the Mastery Score and set the cmi.core.lesson_status to either "passed" or "failed". If no Mastery Score is provided, the LMS will leave the cmi.core.lesson_status as "completed"
- **LMSGetValue():** Returns the value stored in the data model. The return must be one of the set vocabularies for the status.
    - **Error Code:**
        - **401 - Not Implemented Error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.*
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
    - **Error Code:**
        - **405 – Incorrect Data Type:** If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory.*
- **Example Return/Set Values:**
  "completed"
  "failed"
  "browsed"

SCO Usage:

- passed – used when SCO is taken for credit.
- failed – used when SCO is taken for credit.
- completed – used when SCO is taken for no-credit.
- incomplete – used when SCO is taken for no-credit or credit, used when SCO is exited prematurely (before a passed/failed/completed status could be determined).
- browsed – used when the lesson_mode is browse.
- not attempted – SCO should never set lesson_status to not attempted (This is initialized by the LMS when the student first attempts the SCO).

SCO Usage Example:

```
var lessonStatus =  LMSGetValue("cmi.core.lesson_status");
if (lessonStatus == "failed")
{
   // Student failed the SCO, handle appropriately
}
else
{
   // Student did not fail the SCO, handle appropriately
}
```

cmi.core.entry

Supported API calls
LMSGetValue()

LMS Mandatory: Yes

Data Type: CMIVocabulary (Entry)
"ab-initio"
"resume"
"" *- empty string*

SCO Accessibility:
Read Only

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

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

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

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

LMS Behavior:

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

SCO Usage Example:

```
var entryStatus = LMSGetValue("cmi.core.entry")
if (LMSGetLastError() == "0")
{
    if (entryStatus == "resume")
    {
       // Student is resuming SCO
    }
    else
    {
      // This is the first time the student has entered the SCO
    }
}
else
{
   // Error condition, handle appropriately
}
```

cmi.core.score

Indication of the performance of the student.

Children of cmi.core.score:

raw, min, max

cmi.core.score._children

Supported API calls
LMSGetValue()

LMS Mandatory: Yes

Data Type: CMIString255

SCO Accessibility:
Read Only

Definition: The _children keyword is used to determine all of the elements in the score category that are supported by the LMS. If an element has no children, but is supported, an empty string is returned. If an element is not supported, there is no return. A subsequent request for last error can verify that the element is not supported.

Usage: Used to determine what cmi.core.score children are supported by the LMS. Raw is the only mandatory element that must be supported.

Format: The return value is a comma separated list of all the element names in the score category that are supported by the LMS.

LMS Behavior:

- **Initialization:** The set of supported children for this group. So that on an LMSGetValue() request, the appropriate list of supported children is returned.
- **LMSGetValue():** Returns a comma separated list of supported elements.
  - **Example API call:** LMSGetValue("cmi.core.score._children")
  - **Example Return Values:**"raw" - *conformant LMS's must support at least this element*"raw,min,max""raw,min"
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory.*
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **402 - Invalid set value, element is keyword.** If a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory*

SCO Usage Example:

```
var scoreChildren =  LMSGetValue("cmi.core.score._children");
if (coreChildren.indexOf("min") != -1)
{
   LMSSetValue("cmi.core.score.min","10");
}
```

cmi.core.score.raw

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: Yes

Definition: Indication of the performance of the student during his last attempt on the SCO. This score may be determined and calculated in any manner that makes sense to the SCO designer. 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.

Data Type: CMIDecimal or CMIBlank

SCO Accessibility: Read / Write

The cmi.core.score.raw must be a normalized value between 0 and 100.

Usage: When the student is in their first attempt at a SCO the cmi.core.score.raw should be set to "" (empty string). For additional attempts the cmi.core.score.raw reflects what was recorded on the student's last previous attempt. If no cmi.core.score.raw was set in a SCO and a request for the cmi.core.score.raw was made by the SCO, then an empty string should be returned ("").

Format: Decimal number or blank.

LMS Behavior:

- **Initialization:** LMS should initialize this to an empty string ("") upon initial launch of a SCO. The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("")
- **LMSGetValue():** Returns the value stored in the data model. The value returned must be of type CMIDecimal or CMIBlank.
  - **Example API call:** LMSGetValue("cmi.core.score.raw")
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory.*
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Example API call:** LMSSetValue("cmi.core.score.raw","85.7")
  - **Error Code:**
    - **405 – Incorrect Data Type:** If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.*
- **Example Return/Set Values:**
  "90"
  "85.7"
  ""

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

LMSSetValue("cmi.core.score.raw","85");

cmi.core.score.max

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: No

Data Type: CMIDecimal or CMIBlank

SCO Accessibility: Read / Write

Definition: The maximum score or total number that the student could have achieved.

The cmi.core.score.max must be a normalized value between 0 and 100.

Usage: Indication of the largest score the student could have achieved.

Format: Decimal number or blank.

LMS Behavior:

- **Initialization:** LMS should initialize this to an empty string ("") upon initial launch of a SCO. The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("").
- **LMSGetValue():** Returns the value stored in the data model. The value returned must be of type CMIDecimal or CMIBlank.
  - **Example API call:** LMSGetValue("cmi.core.score.max")
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.

- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
    - o **Example API call:** LMSSetValue("cmi.core.score.max","100")
    - o **Error Code:**
        - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "100"
  "5"
  ""

SCO Usage Example:

```
var scoreChildren =  LMSGetValue("cmi.core.score._children");
if (coreChildren.indexOf("max") != -1)
{
   LMSSetValue("cmi.core.score.max","100");
}
```

cmi.core.score.min

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: No

Data Type: CMIDecimal or CMIBlank

SCO Accessibility:
Read / Write

Definition: The minimum score that the student could have achieved.

The cmi.core.score.min must be a normalized value between 0 and 100.

Usage: Used to indicate the lowest score the student could have achieved.

Format: Decimal number or blank.

LMS Behavior:

- **Initialization:** LMS should initialize this to an empty string ("") upon initial launch of a SCO. The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("")
- **LMSGetValue():** Returns the value stored in the data model. The value returned must be of type CMIDecimal or CMIBlank.
    - o **Example API call:** LMSGetValue("cmi.core.score.min")
    - o **Error Code:**
        - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
    - o **Example API call:** LMSSetValue("cmi.core.score.min","5")
    - o **Error Code:**
        - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "10"
  "45.5"
  ""

SCO Usage Example:

```
var scoreChildren =  LMSGetValue("cmi.core.score._children");
 if (coreChildren.indexOf("min") != -1)
 {
    LMSSetValue("cmi.core.score.min","10");
 }
```

cmi.core.total_time

Supported API calls:

Definition: Accumulated time of all the student's sessions in the SCO.

LMSGetValue()

LMS Mandatory: Yes

Data Type: CMITimespan

SCO Accessibility:
Read Only

Usage: Used to keep track of the total time spent in every session of the given SCO for the given student. LMS should initialize the cmi.core.total_time to a default value the first time SCO is launched and then use SCO reported values (session_time) to keep a running total.

Format: Hours, minutes and seconds separated by a colon. HHHH:MM:SS.SS
Hours has a minimum of 2 digits and a maximum of 4 digits. Minutes shall consist of exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits. (i.e. 34.45).

LMS Behavior:

- **Initialization:** LMS should initialize this to "0000:00:00.00" upon initial launch of a SCO.
    - **Additional Behavior:** A SCO is able, in a single running, to perform multiple sets of the cmi.core.session_time. When the SCO issues the LMSFinish() or the user navigates away, the LMS should take the last cmi.core.session_time that the SCO set (if there was a set) and accumulate this time to the cmi.core.total_time. Upon subsequent launch of the SCO, and a LMSGetValue() call for cmi.core.total_time, the LMS should return the accumulated time. LMS's should not accumulate multiple time sent to the LMS by the LMSSetValue() call for cmi.core.session_time. If multiple calls to LMSSetValue() for cmi.core.session_time are made the LMS should overwrite any existing value that it is persisting for cmi.core.session_time.
- **LMSGetValue():** Returns the value stored in the data model. The value returned must be of type CMITimespan.
    - **Example API call:** LMSGetValue("cmi.core.total_time")
    - **Example Return Values:**
      "00:29:00"
      "01:27:45.5"
    - **Error Code:**
        - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory.*
- **LMSSetValue():** LMS should set an error code according to the following:
    - **Error Code:**
        - **403 - Element is read only.** If a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element must be supported by LMS since the element is mandatory.*

SCO Usage Example:

```
 var totalTime =  LMSGetValue("cmi.core.total_time");
 if (LMSGetLastError() == "0" )
 {
    // Use cmi.core.total_time
 }
```

cmi.core.lesson_mode

Supported API calls:
LMSGetValue()

LMS Mandatory: No

Data Type: CMIVocabulary
(Mode)
"browse"
"normal"
"review"

SCO Accessibility:

Definition: Identifies the SCO behavior desired after launch. Many SCOs have a single "behavior". Some SCOs, however, can present different amounts of information, or present information in different sequences, or present information reflecting different training philosophies based on an instructor's or designer's decisions. Designers may enable SCOs to behave in a virtually unlimited number of ways. This standard supports the communication of three parameters that may result in different SCO behaviors.

Usage: Used to represent the different modes that a SCO can be launched in. Used in conjunction with lesson_status**.**

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

Read Only	• **"browse":** The student wants to preview the materials, but not necessarily challenge the SCO for a grade.
	• **"normal":** This indicates that the SCO should behave as designed for a student wanting to get credit for his learning.
	• **"review":** The student has already seen the material at least once and been graded.
	
	If an unrecognized or unanticipated lesson_mode is received, then **normal** is assumed by the SCO.
	
	LMS Behavior:
	▪ **Initialization:** LMS should determine the mode in which the SCO is being launched.
	NOTE: Right now there is no SCORM defined way to signify that learning content can be taken in different modes. Implementation will therefore be LMS specific.
	▪ **LMSGetValue():** Returns the value stored in the data model. The return must be one of the set vocabularies for the entry.
	o **Example API call:** LMSGetValue("cmi.core.lesson_mode")
	o **Example Return Values:**
	“browse"
	"normal"
	"review"
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.
	▪ **LMSSetValue():** LMS should set an error code according to the following:
	o **Error Code:**
	▪ **403 - Element is read only.** If a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	
	SCO Usage Example:
	var mode = LMSGetValue(“cmi.core.lesson_mode”);
	if (LMSGetLastError() == “0”)
	{
	  if (mode == "browse")
	  {
	    // Student is browsing the SCO
	  }
	  else if (mode == "review")
	  {
	    // Student has already seen an been graded
	    // Must be reviewing material
	  }
	  else
	  {
	    // Student is launching the SCO in the normal mode
	  }
	}

cmi.core.exit	
Supported API calls: LMSSetValue() **LMS Mandatory:** Yes **Data Type:** CMIVocabulary (Exit) **"time-out"** **"suspend"** **"logout"** **""** ***- empty string***	**Definition:** An indication of how or why the student left the SCO. **Usage:** Used to indicate the reason that the SCO was last exited. **Format:** A set vocabulary phrase. Three possible vocabulary values: • **"time-out":** This indicates the SCO ended because the SCO has determined an excessive amount of time has elapsed, or the max_time_allowed has been exceeded. The max_time_allowed can be found in the manifest (adlcp:maxtimeallowed for the item) • **"suspend":** This indicates the student leaves the SCO with the intent of returning to it later at the point where he/she left. • **"logout":** This indicates that the student logged out from within the SCO

SCO Accessibility:
Write Only

instead of returning to the LMS system to log out. This implies that the SCO passed control to the LMS system, and the LMS system automatically logged the student out of the course -- after updating the appropriate data model elements.

- **"" :** The empty string vocabulary should be used to represent a normal exit state.

LMS Behavior:

- **Initialization:** Element does not need initialized. There is never a LMSGetValue() done on this element. Element is controlled by the SCO.
  - **Additional behavior:** A SCO has the option of setting the cmi.core.exit to one of four values. Based on the value, the LMS behavior should be as follows:
    - If the SCO set the cmi.core.exit to "time-out" the LMS should set cmi.core.entry to "" (empty) upon the next launching of the SCO.
    - If the SCO set the cmi.core.exit to "suspend" the LMS should set the cmi.core.entry to "resume" upon the next launching of the SCO.
    - If the SCO set the cmi.core.exit to "logout" the LMS should set the cmi.core.entry to "" (empty) upon the next launching of the SCO. In addition, the LMS should log the student out of the course when the SCO that set the cmi.core.exit to "logout" has issued the LMSFinish() or the user navigates away.
    - If the SCO set the cmi.core.exit to "" (empty) the LMS should set the cmi.core.entry to "" (empty) upon the next launching of the SCO.
    - If the SCO did not set the cmi.core.exit to any value that LMS should set cmi.core.entry to "" (empty) upon the next launching of the SCO.
- **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.*
    - **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Example API call:** LMSSetValue("cmi.core.exit","logout")
  - **Example Set Values:**
    "time-out"
    "suspend"
    "logout"
  - **Error Code:**
    - **405 – Incorrect Data Type:** If the element is supported (element must be supported by LMS since the element is mandatory) and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.*NOTE element <u>must be</u> supported by LMS since the element is mandatory.*

SCO Usage Example:
LMSSetValue("cmi.core.exit","time-out")

cmi.core.session_time

Supported API calls:
LMSSetValue()

Definition: This is the amount of time in hours, minutes and seconds that the student 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.

LMS Mandatory: Yes **Data Type:** CMITimespan **SCO Accessibility:** Write Only	**Usage:** Used to keep track of the time spent in a SCO for a session. The LMS will use this time in determining the cmi.core.total_time. **Format:** A length of time in hours, minutes and seconds shown in the following numerical format: HHHH:MM::SS.SS Hours has a minimum of 2 digits and a maximum of 4 digits. Minutes shall consist of exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits. (i.e. 34.45). **LMS Behavior:** ▪ **Initialization:** Element does not need initialized by the LMS. There is never a LMSGetValue() call made on this element. The SCO controls this element. o **Additional Behavior:** A SCO is able, in a single running, to perform multiple sets of the cmi.core.session_time. When the SCO issues the LMSFinish() or the user navigates away, the LMS should take the last cmi.core.session_time that the SCO set (if there was a set) and accumulate this time to the cmi.core.total_time. Upon subsequent launch of the SCO, and a LMSGetValue() call for cmi.core.total_time, the LMS should return the accumulated time. LMS's should not accumulate multiple time sent to the LMS by the LMSSetValue() call for cmi.core.session_time. If multiple calls to LMSSetValue() for cmi.core.session_time are made the LMS should overwrite any existing value that it is persisting for cmi.core.session_time. ▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string (""). o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.* ▪ **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string (""). ▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element. o **Example API call:** LMSSetValue("cmi.core.session_time","0010:34:34.56") o **Example Set Values:** "0010:34:34.56" "05:15:00" o **Error Code:** ▪ **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type. ▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.* **SCO Usage Example:** LMSSetValue("cmi.core.session_time","0000:12:30")

cmi.suspend_data

Supported API calls: LMSGetValue() LMSSetValue() **LMS Mandatory:** Yes **Data Type:** CMIString4096 **SCO Accessibility:** Read / Write	**Definition:** Unique information generated by the SCO during previous uses that is needed for the current use. This unique information is applicable to a launching SCO. Normally this is the element used by the SCO for restart information. This is normally data that is created by the SCO and stored by the LMS to pass back to the SCO the next time the SCO is run. The LMS must set aside a space for this group for each SCO for each student. It stores this data and returns it to the SCO when it is run again. The LMS shall retain this data as long as the student is in the course. **Usage:** Only available on restart of a SCO. SCO could set this value if a student exits before SCO is completed. The SCO then could use this information on restart.

Format: SCO unique. The only limitations on this data are:

1. Data must be transferred in ASCII format. The SCO may then convert it to any form that it requires.
2. To avoid over-burdening the LMS this element should be limited by 4096 bytes of data.

LMS Behavior:

- **Initialization:** LMS should initialize this element to blank (empty string). The SCO is responsible for setting this element. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string (“”).
- **LMSGetValue():** Returns the value stored in the data model. The return must be of the correct data type.
    - **Error Code:**
        - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.*
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
    - **Error Code:**
        - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.*
- **Example Return/Set Values:**
  SCO implementation dependent

SCO Usage Example:

```
// Upon re-launch of a SCO by a student
// Get the Suspend Data
var suspendData = LMSGetValue("cmi.suspend_data")
if (LMSGetLastError() == “0”)
{
  // Use suspend data
}
```

cmi.launch_data

Supported API calls:
LMSGetValue()

LMS Mandatory: Yes

Data Type: CMIString4096

SCO Accessibility:
Read Only

Definition: Unique information generated at the SCO’s creation that is needed for every use. Without this information, a SCO may not execute.

Usage: The cmi.launch_data is available to the SCO to aid in launching the SCO. This will always be the same for a given SCO.

Format: Text field. This contains whatever system-unique information is necessary for the SCO to function well. This field is limited to 4096 characters.

LMS Behavior:

- **Initialization:** This value should be initialized by the LMS using the manifest. The LMS should use the manifest (adlcp:datafromlms) element. If no launch data is found in the manifest, then the launch data should be set to an empty string ("").
- **LMSGetValue():** Returns the value stored in the data model. The return must be of the correct data type.
    - **Error Code:**
        - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. *NOTE element <u>must be</u> supported by LMS since the element is mandatory.*
- **LMSSetValue():** LMS should set an error code according to the following:

	o **Error Code:**
	▪ **403 - Element is read only.** If a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	SCO Usage Example:
	var launchData = LMSGetValue("cmi.launch_data")
	if (LMSGetLastError() == "0")
	{
	// use launch_data
	}

cmi.comments

Mechanism for collecting and distributing comments for a SCO.

cmi.comments - represents the ability for the SCO to Get/Set comments
cmi.comments_from_lms - represents the ability to provide comments to the SCO (read only from a SCO perspective).

cmi.comments

Supported API calls:
LMSGetValue()
LMSSetValue()
 	
LMS Mandatory: No
 	
Data Type: CMIString4096
 	
SCO Accessibility:
Read / Write
	Definition: Freeform feedback from the SCO. For example, the student may have the option of leaving comments at any point in the SCO, or they may be asked for comments at the end of the SCO. The comment may also have an indication of where or when in the SCO it was created. A location may be tagged and embedded in the comment.
	
	Usage: Used to allow the SCO to send comments to the LMS about the SCO. Could be used to collect student entered comments.
	
	Format: Freeform alphanumeric text and special characters.
	
	LMS Behavior:
	▪ **Initialization:** None, should be initialized to an empty string (""). The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("").
	▪ **LSGetValue():** turns the value stored in the data model. The return must be of the correct data type. If no comments were provided the LMS should return an empty string ("").
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **LMSSetValue():** Sets the data model element to the supplied value. The LMSSetValue() request sets the comments. The comments should be concatenated together.
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	▪ **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
	▪ **Example Return/Set Values:**
	SCO implementation dependent
	
	SCO Usage Example:
	var comments = LMSGetValue("cmi.comments")
	if (LMSGetLastError() == "0")
	{
	// use launch_data
	}

cmi.comments_from_lms

Supported API calls:
LMSGetValue()
 	
LMS Mandatory: No
 	
Data Type: CMIString4096
 	
SCO Accessibility:
Read Only
	Definition: This element represents comments that would come from the LMS. An example of how this might be used is in the form of instructor comments. These types of comments are directed at the student that the SCO may present to the student when appropriate.
	
	Usage: Used to allow the SCO to see any comments related to the SCO that originated in the LMS.
	
	Format: no specific format
	
	LMS Behavior:
	▪ **Initialization:** These comments should be initialized by the LMS if provided.
	▪ **LMSGetValue():** Returns the value stored in the data model. The return must be of the correct data type. If no instructor comments were provided the LMS should return an empty string ("").
	o **Example API call:** LMSGetValue("cmi.comments_from_lms")
	o **Example Return Values:** Implementation dependent
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **LMSSetValue():** LMS should set an error code according to the following:
	o **Error Code:**
	▪ **403 - Element is read only.** If a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	
	SCO Usage Example:
	// The SCO may want to see if there are any comments from the LMS for the student.
	var commentsFromLMS = LMSGetValue("cmi.comments_from_lms");
	if (LMSGetLastError() == "0")
	{
	// use the comments from the LMS
	}

cmi.objectives

Identifies how the student has performed on individual objectives covered in the SCO.

Children of cmi.objectives:

id, score, status

cmi.objectives._children

Supported API calls: LMSGetValue() **LMS Mandatory:** No **Data Type:** CMIString255 **SCO Accessibility:** Read Only	**Definition:** The children keyword is used to determine all of the elements in the cmi.objectives category that are supported by the LMS. If an element has no children, but is supported, an empty string is returned. If an element is not supported, there is no return. A subsequent request for last error can verify that the element is not supported. **Usage:** Used to determine which elements are supported by the LMS. **Format:** The return value is a comma separated list of all the element names in the cmi.objectives category that are supported by the LMS. Identifies what detailed information can be collected, on the student's performance in complex scenarios, such as

simulations.

LMS Behavior:

- **Initialization:** The set of supported children for this group. So that on an LMSGetValue() request, the appropriate list of supported children is returned
- **LMSGetValue():** Returns a comma separated list of supported elements
  - **Example API call:** LMSGetValue("cmi.objectives._children")
  - **Example Return Values:**
    “id,score”
    "id,status"
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.

SCO Usage Example:
SCO could use this element to determine which elements are supported by the LMS

```
var objChildren = LMSGetValue("cmi.objectives._children")
if (objChildren.indexOf(“status”) != -1)
{
   // Set the Objectives Status
}
```

cmi.objectives._count

Supported API calls:
LMSGetValue()

LMS Mandatory: No

Data Type: CMIInteger

SCO Accessibility:
Read Only

Definition: The _count keyword is used to determine the current number of records in the cmi.objectives list. The total number of entries is returned. If the SCO does not know the count of the cmi.objectives records, it can begin the current student count with 0. This would overwrite any information about objectives currently stored in the first index position. Overwriting or appending is a decision that is made by the SCO author when he/she creates the SCO.

Usage: Used to determine the number of objectives stored by the LMS. SCOs could use this number to determine which objective record to retrieve. If "3'" is returned then the SCO knows that it can reference record 0-2.

Format: Returns the value as an integer that indicates the number of items currently in an element list or array.

LMS Behavior:

- **Initialization:** LMS is responsible for initializing this to 0 on initial launch of a SCO, no objective data has been inserted by the SCO.
- **LMSGetValue():** Returns the total number of objective entries stored by the LMS.
  - **Example API call:** LMSGetValue("cmi.objectives._count")
  - **Example Return Values:**
    “0”
    "4"
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to

indicate that the element is not supported.

SCO Usage Example:
SCO could use the _count element to determine which array index to use later in the process.

```
// get the count of objectives recorded by the LMS
var totalObj = LMSGetValue("cmi.objectives._count")

// The value return from the LMS is the total number record

// subtract one from the total (to compensate for the zero based array)
var request = "cmi.objectives." + totalObj - 1 + ".id"

// Call set on the new Objective ID
LMSSetValue(request, "Obj1_110")
```

cmi.objectives.n.id

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: No

Data Type: CMIIdentifier

SCO Accessibility:
Read / Write

Definition: An internally, developer defined, SCO specific identifier for an objective.

Usage: Way of identifying an objective.

Format: Alpha-numeric string. No internal spaces.

LMS Behavior:
- **Initialization:** Initialized to an empty string (""). The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string (“”)
- **LMSGetValue():** Returns the objectives id stored in the LMS. The returned value must match the associated data type.
  - **Example API call:** LMSGetValue("cmi.objectives.0.id")
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Error Code:**
    - **405 – Incorrect Data Type** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "A1317"
  "Obj123"

SCO Usage Examples:
SCO may want to display summary information about the objectives.

```
var objID = LMSGetValue("cmi.objectives.0.id")
// Display the objective ID for the SCO
```

SCO may want to set the objective ID for reporting reasons.
```
LMSSetValue("cmi.objectives.0.id","Obj1")
```

cmi.objectives.n.score

Each objective can contain an associated score

Children of cmi.objectives.n.score:

raw, min, max

cmi.objecitves.n.score._children

Supported API calls:
LMSGetValue()

LMS Mandatory: No

Data Type: CMIString255

SCO Accessibility:
Read Only

Definition: The children keyword is used to determine all of the elements in the cmi.objectives.n.score category that are supported by the LMS. If an element has no children, but is supported, an empty string is returned. If an element is not supported, there is no return. A subsequent request for last error can verify that the element is not supported.

Usage: Used to determine which elements are supported by the LMS.

Format: The return value is a comma separated list of all the element names in the cmi.objectives.n.score category that are supported by the LMS.

LMS Behavior:

- **Initialization:** The set of supported children for this group. So that on an LMSGetValue() request, the appropriate list of supported children is returned.
- **LMSGetValue():** Returns a comma separated list of supported elements.
  - **Example API call:** LMSGetValue("cmi.objectives.0.score._children")
  - **Example Return Values:**
    "raw,min,max"
    "raw"
    ""
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.

SCO Usage Example:
SCO could use this element to determine which elements are supported by the LMS

```
var objScoreChildren = LMSGetValue("cmi.objectives.0.score")
if (objScoreChildren.indexOf("raw") != -1)
{
   // Set the Objectives score
}
```

cmi.objectives.n.score.raw

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: No

Data Type: CMIDecimal or CMIBlank

SCO Accessibility:
Read / Write

Definition: Numerical representation of student performance after each attempt on the objective. May be unprocessed raw score.

The cmi.objectives.n.score.raw must be a normalized value between 0 and 100.

Usage: Raw score after each attempt for an objective.

Format: Decimal number or blank.

LMS Behavior:

- **Initialization:** Element should be initialized to an empty string (""). The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("").
- **LMSGetValue():** The LMS should return the objectives raw score identified by the request. The returned value must match the associated data type.
  - **Example API call:** LMSGetValue("cmi.objectives.0.score.raw")
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Example API call:** LMSSetValue("cmi.objectives.0.score.raw","5")
  - **Error Code:**

    - **405 – Incorrect Data Type** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "96.7"
  "5"
  ""

SCO Usage Example:
SCO could use this to set a raw score associated with an objective.

```
var objScoreChildren = LMSGetValue("cmi.objectives.0.score")
if (objScoreChildren.indexOf(“raw”) != -1)
{
   LMSSetValue("cmi.objectives.0.score.raw","85")
}
```

cmi.objectives.n.score.max

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: No

Data Type: CMIDecimal or CMIBlank

SCO Accessibility:
Read / Write

Definition: The maximum score or total number that the student could have achieved on the objective.

The cmi.objectives.n.score.max must be a normalized value between 0 and 100.

Usage: Max score after each attempt for an objective.

Format: Decimal number or blank.

LMS Behavior:
- **Initialization:** Element should be initialized to an empty string (""). The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string (“”).
- **LMSGetValue():** The LMS should return the objectives max score identified by the request. The returned value must match the associated data type.
  - **Example API call:** LMSGetValue("cmi.objectives.0.score.max")
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Example API call:** LMSSetValue("cmi.objectives.0.score.max","10")
  - **Error Code:**
    - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "100"
  "10"
  ""

SCO Usage Example:
SCO could use this to set a max score associated with an objective.

```
var objScoreChildren = LMSGetValue("cmi.objectives.0.score")
if (objScoreChildren.indexOf(“max”) != -1)
{
   LMSSetValue("cmi.objectives.0.score.max","5")
}
```

cmi.objectives.n.score.min

Supported API calls:
LMSGetValue()
LMSSetValue()

Definition: The minimum score that the student could have achieved on the objective.

The cmi.objectives.n.score.min must be a normalized value between 0 and 100.

LMS Mandatory: No

Data Type: CMIDecimal or CMIBlank

SCO Accessibility: Read / Write

Usage: Min score after each attempt for an objective.

Format: Decimal number or blank.

LMS Behavior:

- **Initialization:** Element should be initialized to an empty string ("").The SCO is responsible for setting this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string (“”)
- **LMSGetValue():** The LMS should return the objectives min score identified by the request. The returned value must match the associated data type.
    - **Example API call:** LMSGetValue("cmi.objectives.0.score.min")
    - **Error Code:**
        - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
    - **Example API call:** LMSSetValue("cmi.objectives.0.score.min","5")
    - **Error Code:**
        - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "0"
  "45.5"
  ""

SCO Usage Example:
SCO could use this to set a min score associated with an objective.

```
var objScoreChildren = LMSGetValue("cmi.objectives.0.score")
if (objScoreChildren.indexOf(“min”) != -1)
{
   LMSSetValue("cmi.objectives.0.score.min","2")
}
```

cmi.objectives.n.status

Supported API calls
LMSGetValue()
LMSSetValue()

LMS Mandatory: No

Data Type:
CMIVocabulary (Status)
"passed"
"completed"
"failed"
"incomplete"
"browsed"
"not attempted"

SCO Accessibility: Read / Write

Definition: The status of the SCO’s objective obtained by the student after each attempt to master the SCO’s objective. Only 6 possible vocabulary values: passed, completed, failed, incomplete, not attempted or browsed.

Usage: Used to keep track of the students statuses for a given objective.

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

- **"passed**: Necessary number of objectives in the SCO were mastered, or the necessary score was achieved. Student is considered to have completed the SCO and passed.
- **"completed"**: The objective may or may not be passed, but objective was experienced by the student. The student is considered to have completed the objective. For instance, passing may depend on a certain score known to the LMS system. The SCO knows the raw score but not whether that raw score was high enough to pass the objective.
- **"failed"**: The objective was not passed. The objective may or may not have been completed by the student. The student is considered to have experienced the material associated with the objective and failed.
- **"incomplete"**: The material associated with the objective was begun but not finished.
- **"browsed"**: The student launched the SCO containing the material associated with the objective with an LMS mode of Browse on the initial attempt.
- **"not attempted"**: Incomplete implies that the student made an attempt to perform the SCO, but for some reason was unable to finish it. Not attempted means that the student did not even begin the SCO containing the objective. Maybe he just read the table of contents, or SCO abstract and decided he was not ready. Any algorithm within the SCO may be used to determine when the objective moves from "not attempted" to "incomplete".

LMS Behavior:

- **Initialization:** Handled by the SCO.
- **LMSGetValue():** The LMS should return the objectives status identified by the request. The returned value must match the associated data type.
  - **Example API call:** LMSGetValue("cmi.objectives.0.status")
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Example API call:** LMSSetValue("cmi.objectives.0.status","failed")
  - **Error Code:**
    - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- Example Return/Set Values:
  "not attempted"
  "passed"
  "browsed"

SCO Usage Example:
SCO could use this to set a status associated with an objective

```
var objScoreChildren = LMSGetValue("cmi.objectives._children")
if (objScoreChildren.indexOf("status") != -1)
{
   LMSSetValue("cmi.objectives.0.status","failed")
}
```

cmi.student_data

Information to support customization of a SCO based on a student's performance.

Children of cmi.student_data:

mastery_score, max_time_allowed, time_limit_action

cmi.student_data._children

Supported API calls:
LMSGetValue()

LMS Mandatory: No

Data Type: CMIString255

SCO Accessibility:
Read Only

Definition: The _children keyword is used to determine all of the elements in the student_data category that are supported by the LMS. If an element has no children, but is supported, an empty string is returned. If an element is not supported, an empty string is returned. A subsequent request for last error can verify that the element is not supported.

Usage: To determine which cmi.student_data data elements are supported by the LMS.

Format: The return value is a comma-separated list of all of the element names in the student_data category that are supported by the LMS.

LMS Behavior:

- **Initialization:** The set of supported children for this group. So that on an LMSGetValue() request, the appropriate list of supported children is returned.
- **LMSGetValue():** LMS returns a comma separated list of supported elements.
  - **Example Return Values:**
    "mastery_score,time_limit_action,max_time_allowed"
    "max_time_allowed"
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**

- **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
- **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.

SCO Usage Example:

```
var studentDataChildren =  LMSGetValue("cmi.student_data._children");
if (studentDataChildren.indexOf("attempt_number") != -1)
{
   var maxTimeAllowed = LMSGetValue("cmi.student_data.max_time_allowed");
}
```

cmi.student_data.mastery_score

Supported API calls:
LMSGetValue()

LMS Mandatory: No

Data Type: CMIDecimal

SCO Accessibility:
Read Only

Definition: The passing score, as determined outside the SCO. When the SCO score is greater than or equal to the mastery score, the student is considered to have passed, or mastered the content. In some cases, the SCO does not know what this passing score is, because it is determined by the LMS system.

Usage: For an LMS system to support mastery_score, it must be able to change the lesson_status based on the score passed to if from the SCO. Just passing a mastery_score to a SCO does not constitute full support for this feature.

Format: Decimal number

LMS Behavior:

- **Initialization:** LMS is responsible – value obtained from manifest (adlcp:masteryscore) item elements.
- **LMSGetValue():** Returns the current value stored by the LMS.
  - **Example Return Values:**
    "75"
    "100"
    "5"
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **403 - Element is read only.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.

SCO Usage Example:

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

cmi.student_data.max_time_allowed

Supported API calls:
LMSGetValue()

LMS Mandatory: No

Data Type: CMITimespan

SCO Accessibility:
Read Only

Definition: The amount of time the student is allowed to have in the current attempt on the SCO. See time_limit_action for the SCO's expected response to exceeding the limit.

Usage: Used to present the SCO with the maximum amount of time that the student is allowed to be in the SCO.

Format: Hours, minutes and seconds separated by a colon. HHHH:MM:SS.SS
Hours has a minimum of 2 digits and a maximum of 4 digits. Minutes shall consist of exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits. (i.e. 34.45).

LMS Behavior:

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

cmi.student_data.time_limit_action	
Supported API calls: LMSGetValue() **LMS Mandatory:** No **Data Type:** CMIVocabulary (Time Limit Action) **"exit,message"** **"exit,no message"** **"continue,message"** **"continue,no message"** **SCO Accessibility:** Read Only	**Definition:** Tells the SCO what to do when the max_time_allowed is exceeded. There are two arguments for this element: What the SCO should do – exit or continue What the student should see – message or no message **Usage:** Used to indicate to the SCO what the action should be when the maximum time allowed in the SCO has been exceeded. **Format:** A set vocabulary phrase. Four possible vocabulary values: • **"exit,message"** • **"exit,no message"** • **"continue,message"** • **"continue,no message"** **LMS Behavior:** ▪ **Initialization:** LMS is responsible – value obtained from manifest (adlcp:timelimitaction) item elements. ▪ **LMSGetValue():** Returns the current value stored by the LMS. o **Example Return Values:** "exit,message""continue,no message" o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. ▪ **LMSSetValue():** LMS should set an error code according to the following: o **Error Code:** ▪ **403 - Element is read only.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 403. ▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. **SCO Usage Example:** var timeLimitActionValue = LMSGetValue("cmi.student_data.time_limit_action");

cmi.student_preference

Selected options that are appropriate for subsequent SCOs

Children of cmi.student_preference:
audio, language, speed, text

cmi.student_preference._children

Supported API calls:
LMSGetValue()
 	
LMS Mandatory: No
 	
Data Type: CMIString255
 	
SCO Accessibility:
Read Only
	Definition: The _children keyword is used to determine all of the elements in the student_preference category that are supported by the LMS. If an element has no children, but is supported, an empty string is returned. If an element is not supported, an empty string is returned. A subsequent request for last error can verify that the element is not supported.
	
	Usage: To determine which cmi.student_preference data elements are supported by the LMS.
	
	Format: The return value is a comma separated list of all of the element names in the student_preference category that are supported by the LMS.
	
	LMS Behavior:
	▪ **Initialization:** The set of supported children for this group. So that on an LMSGetValue() request, the appropriate list of supported children is returned
	▪ **LMSGetValue():** LMS returns a comma separated list of supported element.
	o **Example Return Values:** "audio, language, speed" "language"
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **LMSSetValue():** LMS should set an error code according to the following:
	o **Error Code:**
	▪ **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	
	SCO Usage Example:
	var studentPreferenceChildren = LMSGetValue("cmi.student_preference._children");
	if (studentPreferenceChildren.indexOf("audio") != -1)
	{
	LMSSetValue("cmi.student_preference.audio", "10");
	}

cmi.student_preference.audio

Supported API calls: LMSGetValue() LMSSetValue() **LMS Mandatory:** No **Data Type:** CMISInteger **SCO Accessibility:** Read Write	**Definition:** Audio may be turned off, or its volume controlled. The element indicates whether the audio is turned off, or on. **Usage:** Used by the SCO to both set and obtain from the LMS audio preferences of the student. **Format:** Digit from -1 to 100 -1: is off (any negative number is an off command) 0: is a no-change status (the SCO uses its defaults or the status of the audio remains the same as the last SCO taken) 1 - 100: is volume level (1 is soft, 100 is loudest as possible) **LMS Behavior:** ▪ **Initialization:** If supported, the LMS should initialize this value to "0". It is the responsibility of the SCO to set this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string (""). ▪ **LMSGetValue():** Returns the current value stored by the LMS. o **Error Code:** ▪ **401 - Not implemented error**. If element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.

	▪ **LMSSetValue():** Sets the LMS data item to the value passed in as a parameter. o **Error Code:** ▪ **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type. ▪ **401 - Not implemented error**. If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported. ▪ **Example Return/Set Values:** "-1" "0" "50" **SCO Usage Example:** LMSSetValue("cmi.student_preference.audio", "10"); var audioValue = LMSGetValue("cmi.student_preference.audio");
cmi.student_preference.language	
Supported API calls: LMSGetValue() LMSSetValue() **LMS Mandatory:** No **Data Type:** CMIString255 **SCO Accessibility:** Read Write	**Definition:** For SCOs with multi-lingual capability, this element should be used to identify in what language the information should be delivered. **Usage:** Used by the SCO to both set and obtain from the LMS language preferences for the student. **Format:** Alphabetic string, may include white space. **LMS Behavior:** ▪ **Initialization:** If supported the LMS should initialize this value to an empty string (""). It is the responsibility of the SCO to set this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("") ▪ **LMSGetValue():** Returns the current value stored by the LMS. o **Error Code:** ▪ **401 - Not implemented error**. If element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. ▪ **LMSSetValue():** Sets the LMS data item to the value passed in as a parameter. o **Error Code:** ▪ **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type. ▪ **401 - Not implemented error**. If this element is not supported an error code is set to 401by the LMS to indicate that the element is not supported. ▪ **Example Return/Set Values:** "English" "French" **SCO Usage Example:** LMSSetValue("cmi.student_preference.language", "English"); var languageValue = LMSGetValue("cmi.student_preference.language");
cmi.student_preference.speed	
Supported API calls: LMSGetValue() LMSSetValue() **LMS Mandatory:** No **Data Type:** CMIInteger **SCO Accessibility:**	**Definition:** SCOs may sometimes be difficult to understand because of the pace. This element controls the pace of the content delivery. **Usage:** Used by the SCO to both set and obtain from the LMS speed preferences of the student. **Format:** Digit from -100 to 100 -100 is the slowest pace available in the system 0 is no-change status (The SCO uses its defaults. SCO moves at its normal speed) 100 is maximum pace available in the system.

Read
Write

LMS Behavior:

- **Initialization:** If supported by the LMS, this element should be initialized to "0". It is the responsibility of the SCO to set this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("").
- **LMSGetValue():** Returns the current value stored by the LMS.
  - o **Error Code:**
    - **401 - Not implemented error**. If element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the LMS data item to the value passed in as a parameter
  - o **Error Code:**
    - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error**. If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "-100"
  "0"
  "3"

SCO Usage Example:

```
LMSSetValue("cmi.student_preference.speed", "-100");
var speedValue = LMSGetValue("cmi.student_preference.speed");
```

cmi.student_preference.text

Supported API calls:
LMSGetValue()
LMSSetValue()

LMS Mandatory: No

Data Type: CMISInteger

SCO Accessibility:
Read
Write

Definition: In a SCO designed for audio, it may be possible to turn off the audio, and view the audio content in a text window. Or it may be possible to leave the audio on, and request that the text be presented simultaneously with the audio. Or it may be possible to make the text disappear so that only the audio and the screen graphics are available. This element identifies whether the audio text appears in the SCO.

Usage: Used by the SCO to both set and obtain from the LMS text preferences of the student.

Format: One of three digits:
-1 : text is off, not shown
0 : no change in status. Use default
1 : text is on screen, shown to student

LMS Behavior:

- **Initialization:** If supported by the LMS, this element should be initialized to "0". It is the responsibility of the SCO to set this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string ("").
- **LMSGetValue():** Returns the current value stored by the LMS.
  - o **Error Code:**
    - **401 - Not implemented error**. If element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** Sets the LMS data item to the value passed in as a parameter.
  - o **Error Code:**
    - **405 – Incorrect Data Type:** If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
    - **401 - Not implemented error**. If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
- **Example Return/Set Values:**
  "0"
  "-1"
  "1"

SCO Usage Example:

```
LMSSetValue("cmi.student_preference.text", "-1");
var textValue = LMSGetValue("cmi.student_preference.text");
```

cmi.interactions

In this context, an interaction is a recognized and recordable input or group of inputs from the student to the computer. All of the items in this group are related to a recognized and recordable input from the student. The category collects detailed information on each interaction measured as the student takes a SCO.

Children of cmi.interactions:

id, objectives, time, type, correct_responses, weighting, student_response, result, latency

cmi.interactions._children

Supported API calls: LMSGetValue()

LMS Mandatory: No

Data Type: CMIString255

SCO Accessibility: Read Only

Definition: The children keyword is used to determine all of the elements in the cmi.interactions category that are supported by the LMS. If an element has no children, but is supported, an empty string is returned. If an element is not supported, there is no return. A subsequent request for last error can verify that the element is not supported.

Usage: Used to determine which elements are supported by the LMS.

Format: The return value is a comma separated list of all the element names in the cmi.interactions category that are supported by the LMS. Identifies what detailed information can be collected on each interaction measured as the student takes a SCO.

LMS Behavior:

- **Initialization:** The set of supported children for this group. So that on an LMSGetValue() request, the appropriate list of supported children is returned
- **LMSGetValue():** Returns a comma separated list of supported elements.
  - **Example API call:** LMSGetValue("cmi.interactions._children")
  - **Example Return Values:**
    "id,time,type"
    "id,time,type,correct_responses,student_response"
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
  - **Error Code:**
    - **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
    - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.

SCO Usage Example:
SCO could use this element to determine which elements are supported by the LMS

```
var intChildren = LMSGetValue("cmi.interactions._children")
if (intChildren.indexOf("id") != -1)
{
   // Set the Interaction id
}
```

cmi.interactions._count

Supported API calls: LMSGetValue()

LMS Mandatory: No

Data Type: CMIInteger

SCO Accessibility: Read Only

Definition: The _count keyword is used to determine the current number of records in the cmi.interactions list. The total number of entries is returned. If the SCO does not know the count of the cmi.interactions records, it can begin the current student count with 0. This would overwrite any information about interactions currently stored in the first index position. Overwriting or appending is a decision that is made by the SCO author when he/she creates the SCO.

Usage: Used to determine the number of interactions stored by the LMS. SCOs could use this number to determine which interaction record to set. If "3"' is returned then the

	SCO knows that records 0-2 are occupied and the next available index is “3”.
	
	Format: Returns the value as an integer that indicates the number of items currently in an element list or array.
	
	LMS Behavior:
	▪ **Initialization:** LMS is responsible for initializing this to 0 on initial launch of a SCO, no interaction data has been inserted by the SCO.
	▪ **LMSGetValue():** Returns the total number of interaction entries stored by the LMS.
	    o **Example API call:** LMSGetValue("cmi.interactions._count")
	    o **Example Return Values:** “0 "4"
	    o **Error Code:**
	        ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **LMSSetValue():** LMS should set an error code according to the following:
	    o **Error Code:**
	        ▪ **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
	        ▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	
	SCO Usage Example:
	SCO could use the _count element to determine which array index to use later in the process
	
	// get the count of interactions recorded by the LMS
	var totalInteractions = LMSGetValue("cmi.interactions._count")
	
	// The value return from the LMS is the total number records
	
	var request = "cmi.interactions." + totalInteractions + ".id"
	
	// Set the Interaction ID
	LMSSetValue(request, "Int_110")

cmi.interactions.n.id

Supported API calls: LMSSetValue() **LMS Mandatory:** No **Data Type:** CMIIdentifier **SCO Accessibility:** Write Only	**Definition:** Unique identifier for an interaction. **Usage:** Used to set a unique interactions id. SCO specific. **Format:** Alpha-numeric string. No internal spaces. **LMS Behavior:** ▪ **Initialization:** Value is controlled by the SCO. ▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string (""). o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. ▪ **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string (""). ▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element. o **Example API call:** LMSSetValue(“cmi.interactions.0.id”,"I_001") o **Example Set Values:** "I_001" "i1" o **Error Code:** ▪ **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.

        - **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.

SCO Usage Example:
// Set the first interactions id
LMSSetValue("cmi.interactions.0.id","I_001");

cmi.interactions.n.objectives
cmi.interactions.n.objectives._count

Supported API calls:
LMSGetValue()

LMS Mandatory: No

Data Type: CMIInteger

SCO Accessibility:
Read Only

Definition: The _count keyword is used to determine the current number of records in the cmi.interactions objective id list. The total number of entries is returned. If the SCO does not know the count of the cmi.interactions.n.objectives records, it can begin the current student count with 0. This would overwrite any information about objective ids currently stored in the first index position. Overwriting or appending is a decision that is made by the SCO author when he/she creates the SCO.

Usage: Used to determine the number of objective ids stored by the LMS for a given interaction. SCOs could use this number to determine which objective id record to set. If "3"' is returned then the SCO knows that records 0-2 are occupied and the next available index is “3”.

Format: Returns the value as an integer that indicates the number of items currently in an element list or array.

LMS Behavior:

- **Initialization:** LMS is responsible for initializing this to 0 on initial launch of a SCO, no objective id data has been inserted by the SCO.
- **LMSGetValue():** Returns the total number of objective id entries stored by the LMS for a given interaction.
    - **Example API call:**
      LMSGetValue("cmi.interactions.3.objectives._count")
    - **Example Return Values:**
      “0”
      "4"
    - **Error Code:**
        - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
- **LMSSetValue():** LMS should set an error code according to the following:
    - **Error Code:**
        - **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
        - **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.

SCO Usage Example:
SCO could use the _count element to determine which array index to use later in the process

// get the count of objective ids recorded by the LMS for a given interaction
var totalIDs = LMSGetValue("cmi.interactions.0.objectives_count")

// The value return from the LMS is the total number records

var request = "cmi.interactions.0.objectives." + totalID + ".id"

// Set the Interactions Objective ID
LMSSetValue(request, “ObjID-110")

cmi.interactions.n.objectives.n.id

Supported API calls:
LMSSetValue()

Definition: Developer created identifier for an objective.

LMS Mandatory: No
 	
Data Type: CMIIdentifier
 	
SCO Accessibility: Write Only
	Usage: Used to identify the objective that the interaction is for.
	
	Format: Alpha-numeric string. No internal spaces.
	
	LMS Behavior:
	▪ **Initialization:** Value is controlled by the SCO.
	▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
	▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
	o **Example API call:** LMSSetValue(“cmi.interactions.0.objectives.0.id","A1333")
	o **Example Set Values:**
	"A1333”
	"Obj123"
	o **Error Code:**
	▪ **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.
	
	SCO Usage Example:
	// set the objective ID for the given interaction
	LMSSetValue("cmi.interactions.0.objectives.0.id","A1333")

cmi.interactions.n.time

Supported API calls: LMSSetValue() **LMS Mandatory:** No **Data Type:** CMITime **SCO Accessibility:** Write Only	**Definition:** Identification of when the student interaction was completed. **Usage:** Used as a timestamp for the interaction. **Format:** A chronological point in a 24 hour clock. Identified in hours, minutes and seconds in the format: HH:MM:SS.S Hours and minutes shall contain exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits (i.e. 34.45) **LMS Behavior:** ▪ **Initialization:** Value is controlled by the SCO. ▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string (""). o **Error Code:** ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported. ▪ **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string (""). ▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element. o **Example API call:** LMSSetValue(“cmi.interactions.0.time","12:33:35.5") o **Example Set Values:** "12:33:35.5" "22:30:40" o **Error Code:** ▪ **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type. ▪ **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element

is not supported.

SCO Usage Example:
// Set the Interactions time
LMSSetValue("cmi.interactions.0.time","12:33:35.5")

cmi.interactions.n.type

Supported API calls:
LMSSetValue()

LMS Mandatory: No

Data Type:
CMIVocabulary (Interaction)
"true-false"
"choice"
"fill-in"
"matching"
"performance"
"sequencing"
"likert"
"numeric"

SCO Accessibility:
Write Only

Definition: Indication of which category of interaction is recorded. The type of interaction determines how the interaction response should be interpreted. Seven possible question types are defined below. They are not meant to be limiting. There are other types of questions. However, if one of these seven types is used, these are the identifiers that match those types.

Usage: To indicate the type of interaction that is taking place.

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

- **"true-false":** A question with only two possible responses.
- **"choice":** A question with a limited number of predefined responses from which the student may select. Each response is numbered or lettered. One or more responses may be correct.
- **"fill-in":** A question with a simple one or few-word answer. The answer/response is not predefined, but must be created by the student (as opposed to selected).
- **"matching":** A question with one or two sets of items. Two or more of the members of these sets are related. Answering the question requires finding and matching related members.
- **"performance":** A performance question is in some ways similar to a multiple choice question. However, instead of selecting a written answer, the student must perform a task or action.
- **"sequencing":** In a sequencing question, the student is required to identify a logical order for the members of a list.
- **"likert":** A likert question offers the student a group of alternatives on a continuum. The response is generally based on the student's opinion or attitude.
- **"numeric":** Simple number with or without a decimal point required answering the question. Correct answer may be a single number within a range of numbers.

LMS Behavior:

- **Initialization:** Value is controlled by the SCO.
- **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
    - **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
- **LMSSetValue():**Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Example API call:** LMSSetValue("cmi.interactions.0.type","likert")
  - **Example Set Values:**
    "likert"
    "true-false"
    "performance"
  - **Error Code:**
    - **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.

SCO Usage Example:
// set the interactions type
LMSSetValue("cmi.interactions.0.type","choice");

cmi.interactions.n.correct_responses
cmi.interactions.n.correct_responses._count

Supported API calls: LMSGetValue()
 	
LMS Mandatory: No
 	
Data Type: CMIInteger
 	
SCO Accessibility: Read Only
	Definition: The _count keyword is used to determine the current number of records in the cmi.interactions correct responses list. The total number of entries is returned. If the SCO does not know the count of the cmi.interactions.n.correct_responses records, it can begin the current count with 0. This would overwrite any information about correct responses currently stored in the first index position. Overwriting or appending is a decision that is made by the SCO author when he/she creates the SCO.
	
	Usage: Used to determine the number of correct_responses stored by the LMS for a given interaction. SCOs could use this number to determine which correct_responses pattern record to set. If "3'" is returned then the SCO knows that records 0-2 are occupied and the next available index is "3".
	
	Format: Returns the value as an integer that indicates the number of items currently in an element list or array.
	
	LMS Behavior:
	▪ **Initialization:** LMS is responsible for initializing this to 0 on initial launch of a SCO, no correct_responses pattern data has been inserted by the SCO.
	▪ **LMSGetValue():** Returns the total number of correct_responses pattern entries stored by the LMS for a given interaction.
	o **Example API call:** LMSGetValue("cmi.interactions.0.correct_responses._count")
	o **Example Return Values:** "0" "4"
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **LMSSetValue():** LMS should set an error code according to the following:
	o **Error Code:**
	▪ **402 - Invalid set value, element is a keyword.** If the element is supported by the LMS and a request attempts to invoke an LMSSetValue() on this element, then the LMS should set the error code to 402.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to 401 by the LMS to indicate that the element is not supported.
	
	SCO Usage Example:
	SCO could use the _count element to determine which array index to use later in the process
	
	// get the count of objective ids recorded by the LMS for a given interaction
	var totalPattern = LMSGetValue("cmi.interactions.0.correct_responses._count")
	
	// The value return from the LMS is the total number records
	
	var request = "cmi.interactions.0.correct_responses." + totalPattern + ".pattern"
	
	// Set the Interactions correct_responses pattern
	LMSSetValue(request, "t")

cmi.interactions.n.correct_responses.n.pattern

Supported API calls: LMSSetValue() **LMS Mandatory:** No **Data Type:** CMIFeedback **SCO Accessibility:** Write Only	**Definition:** Description of possible student responses to the interaction. There may be more than one correct response, and some responses may be more correct than others. **Usage:** This is the correct response to the interaction provided by the SCO. **Format:** If the cmi.interactions.n.type is: **true-false:** Then the pattern is a single character or numeral. Legal characters are 0,1,t, and f. 0 corresponds to false. If the response is a complete word (i.e. "true") only the first letter is significant.

	choice: Then the pattern is one or more characters separated by a comma. Integers (0-9), letters (a - z) or both may be used. Each possible response is limited to a single character. If there are more than 26 possibilities, then a performance type response must be used.
	fill-in: Then the pattern is an alphanumeric string. Spaces are significant, after the first printable character.
	numeric: Then the pattern is a single number. The number may or may not have a decimal.
	likert: There is no incorrect response for a likert question. Field may be left blank.
	matching: Then the pattern is pairs of identifiers separated by a period. Each matching possibility consists of a source and a target.
	performance: Then the pattern is an alpha-numeric field limited to 255 characters.
	sequencing: Then the pattern is elements identified in any order. The final positioning of the elements is used to determine correctness, not the order in which they were sequenced.
	
	LMS Behavior:
	▪ **Initialization:** Value is controlled by the SCO.
	▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").
	  o **Error Code:**
	    ▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	    ▪ **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
	▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
	  o **Example API call:** If the cmi.interactions.n.type is matching then LMSSetValue("cmi.interactions.0.correct_responses.0.pattern","1.c,2.b,3.a,4.d")
	  o **Example Set Values:**
	  "1.c,2.b,3.a,4.d"
	  "t"
	  o **Error Code:**
	    ▪ **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.
	    ▪ **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.
	
	SCO Usage Example:
	// set the response pattern
	// Question type is a true-false question
	LMSSetValue("cmi.interactions.0.correct_responses.0.pattern","t");

cmi.interactions.n.weighting	
Supported API calls: LMSSetValue() **LMS Mandatory:** No **Data Type:** CMIDecimal **SCO Accessibility:** Write Only	**Definition:** Interactions vary in importance. The weighting is a factor which is used to identify the relative importance of one interaction compared to another. For instance, if the first interaction has a weight of 15 and the second interaction has a weight of 25, then any combined score that reflects weighting would be more influenced by the second interaction. If all interactions are equal in importance, then each interaction has the same weight. A weight of 0 indicates that the interaction should not be counted in the weighted final score. **Usage:** An interaction may have a weight, and similarly, individual actions or responses inside a complex interaction may have a weight. **Format:** A single floating point number. The decimal point is optional and does not have to appear in every weighting. **LMS Behavior:** ▪ **Initialization:** Value is controlled by the SCO. ▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").

- o **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
    - **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - o **Example API call:** LMSSetValue("cmi.interactions.0.weighting","0.66")
  - o **Example Set Values:**
    "0.66"
    "0"
  - o **Error Code:**
    - **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.

SCO Usage Example:
// Set the weighting for the interaction
LMSSetValue("cmi.interactions.0.weighting","0");

cmi.interactions.n.student_response

Supported API calls
LMSSetValue()

LMS Mandatory: No

Data Type:
CMIFeedback

SCO Accessibility:
Write Only

Definition: Description of possible responses to the interaction. There may be more than one correct response, and some responses may be more correct than others.

Usage: This is the actual student response to the interaction. This value then can be compared with the cmi.interactions.n.correct_responses.n.pattern.

Format:
If the cmi.interactions.n.type is:
true-false: Then a single character or numeral. Legal characters are 0,1,t and f. 0 corresponds to false. If the response is a complete word (i.e. "true") only the first letter is significant.
choice: Then one or more characters separated by a comma. Integers (0-9), letters (a - z) or both may be used. Each possible response is limited to a single character. If there are more than 26 possibilities, then a performance type response must be used.
fill-in: Then an alphanumeric string. Spaces are significant, after the first printable character.
numeric: Then a single number. The number may or may not have a decimal.
likert: Then there is no incorrect response for a likert question. Field may be left blank.
matching: Then pairs of identifiers separated by a period. Each matching possibility consists of a source and a target.
performance: Then alpha-numeric field limited to 255 characters.
sequencing: Then the elements may be identified in any order. The final positioning of the elements is used to determine correctness, not the order in which they were sequenced.

LMS Behavior:
- **Initialization:** Value is controlled by the SCO.
- **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").
  - o **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
    - **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - o **Example API call:** If the cmi.interactions.n.type is matching then LMSSetValue("cmi.interactions.0.student_response","1.c,2.b,3.a,4.d")

- **Example Set Values:**
  "t"
  "1.c,2.b,3.c,4.d"
- **Error Code:**
  - **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.
  - **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.

SCO Usage Example:
// set the students response
// Student response was to a true-false
LMSSetValue("cmi.interactions.0.student_response","f");

cmi.interactions.n.result

Supported API calls:
LMSSetValue()

LMS Mandatory: No

Data Type:
CMIVocabulary (Result)
"correct"
"wrong"
"unanticipated"
"neutral"
"x.x" (CMIDecimal)

SCO Accessibility:
Write Only

Definition: How the system judges the described response.

Usage: This is the actual result from the student_response.

Format: A set vocabulary phrase. Five possible vocabulary values:
- **"correct"**
- **"wrong"**
- **"unanticipated"**
- **"neutral"**
- **"x.x" (CMIDecimal)**

LMS Behavior:
- **Initialization:** Value is controlled by the SCO.
- **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").
  - **Error Code:**
    - **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
    - **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
- **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
  - **Example API call:**
    LMSSetValue("cmi.interactions.0.result","correct")
  - **Example Set Values:**
    "correct"
    "95.5"
    "unanticipated"
  - **Error Code:**
    - **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.
    - **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.

SCO Usage Example:
// set the result for the question
LMSSetValue("cmi.interactions.0.result","correct");

cmi.interactions.n.latency

Supported API calls:
LMSSetValue()

LMS Mandatory: No

Data Type:
CMITimespan

Definition: The time from the presentation of the stimulus to the completion of the measurable response.

Usage: If latency is recorded, there can be a latency figure for each response.

Format: Hours, minutes and seconds separated by a colon. HHHH:MM:SS.SS
Hours has a minimum of 2 digits and a maximum of 4 digits. Minutes shall consist of

SCO Accessibility: Write Only	exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits. (i.e. 34.45).
	
	LMS Behavior:
	▪ **Initialization:** Value is controlled by the SCO.
	▪ **LMSGetValue():** LMS should set an error code according to the following and return an empty string ("").
	o **Error Code:**
	▪ **401 - Not implemented error.** If this element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	▪ **404 - Element is write only.** If a SCO tries to call LMSGetValue() on this element, the LMS should set the error code to 404 and return an empty string ("").
	▪ **LMSSetValue():** Sets the data model element to the supplied value. Value must match the data type for this element.
	o **Example API call:** LMSSetValue("cmi.interactions.0.latency","00:29:00")
	o **Example Set Values:** "00:29:00" "1234:44:30"
	o **Error Code:**
	▪ **205 – Incorrect Data Type:** If an LMSSetValue() is invoked and the value to be used to set the element to is not of the correct Data Type.
	▪ **401 - Not implemented error.** If this element is not supported an error code is set to indicate that the element is not supported.
	
	SCO Usage Example:
	// set the latency for the interaction
	LMSSetValue("cmi.interactions.0.latency","00:20:00");

3.4.5. Data Types and Controlled Vocabulary

There exists a data type definition for each element in the AICC CMI Data Model[4]. The following sections define the specifics of each data type specified for the data model elements. These definitions further define how the API and data model must be implemented.

The following definitions are for the data types used to describe the format of each data element. All of the data types have the first three characters of "CMI" to clearly indicate that they are data types that may be unique to the AICC CMI Data Model[4].

Data Type	Description
CMIBlank	An empty string ("").
CMIBoolean	A vocabulary of two words. ("true" or "false").
CMIDecimal	A number that may have a decimal point. If not preceded by a minus sign, the number is presumed to be positive. Examples are "2","2.2" and "-2.2).

CMIFeedback	A structured description of a student response in an interaction. The structure and contents of the feedback depends upon the type of interaction. The currently defined interactions are: **true-false:** Feedback is one of the following single characters: “0”,”1”,”t”, or “f”. **choice:** Feedback is one or more single characters separated by a comma. Legal characters are “0” to “9” and “a” to “z”. If all the characters must be chosen to assume the feedback is correct, then the comma-separated list must be surrounded by curly brackets: { } **fill-in:** Any alpha-numeric string up to 255 characters in length. After the first letter spaces are significant. **numeric**: CMIDecimal **likert:** Single character. Legal characters are 0 to 9 and a to z. **matching**: One or more pairs of identifiers. Each identifier is a single letter or number (0 to 9 and a to z). The identifiers in a pair are separated by a period. Commas separate the pairs. If all pairs must be matched correctly to consider the interaction correct, then the comma separated list of pairs are surrounded by curly brackets: { } **performance**: This is a very flexible format. Essentially an alphanumeric string of 255 characters or less. **sequencing**: A series of single characters separated by commas. Legal characters are 0 to 9 and a to z. The order of the characters determines the correctness of the feedback.
CMIIdentifier	An alphanumeric group of characters with no white space or unprintable characters in it. Maximum of 255 characters.
CMIInteger	An integer number from 0 to 65536.
CMISInteger	A signed integer number from –32768 to +32768.
CMIString255	A set of ASCII characters with a maximum length of 255 characters.
CMIString4096	A set of ASCII characters with a maximum length of 4096 characters.
CMITime	A chronological point in a 24 hour clock. Identified in hours, minutes and seconds in the format: HH:MM:SS.SS Hours and minutes shall contain exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits (i.e. 34.45).
CMITimespan	A length of time in hours, minutes and seconds shown in the following numerical format: HHHH:MM:SS.SS Hours have a minimum of 2 digits and a maximum of 4 digits. Minutes shall consist of exactly 2 digits. Seconds shall contain 2 digits, with an optional decimal point and 1 or 2 additional digits. (i.e. 34.45).

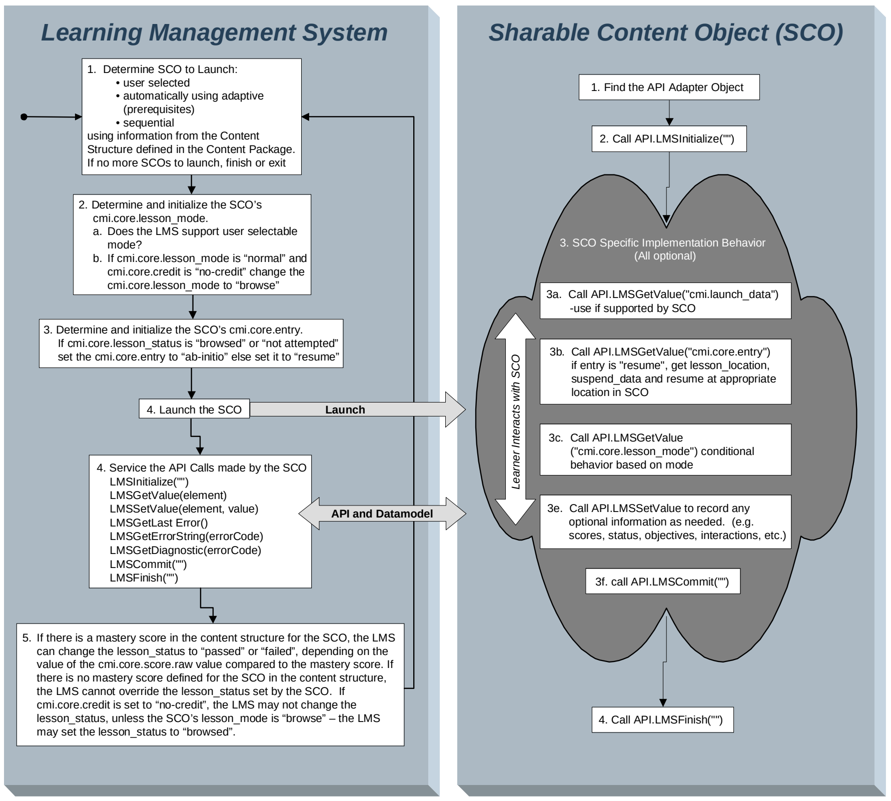
CMIVocabulary	Used to attach specific vocabularies within contexts in a schema. Vocabulary words must be complete and exact matches to those below. Single letters and abbreviations may not be used in API communication. See the table below for the valid list of vocabularies.

In addition to data types, some data model elements are defined with bounded vocabularies of possible values. The table below summarizes the vocabulary type and values for these elements (from AICC CMI001 Guidelines for Interoperability[4]).

Vocabulary Type	Members of Vocabulary
Mode cmi.core.lesson_mode	"normal" "review" "browse"
Status cmi.core.lesson_status cmi.objectives.n.status	"passed" "completed" "failed" "incomplete" "browsed" "not attempted"
Exit cmi.core.exit	"time-out" "suspend" "logout" "" - *(empty string)*
Credit cmi.core.credit	"credit" "no-credit"
Entry cmi.core.entry	"ab-initio" "resume" "" - *(empty string)*
Interaction cmi.interactions.n.type	"true-false" "choice" "fill-in" "matching" "performance" "likert" "sequencing" "numeric"
Result cmi.interactions.n.result	"correct" "wrong" "unanticipated" "neutral" X.X (CMIDecimal)
Time Limit Action cmi.student_data.time_limit_action	"exit,message" "exit,no message" "continue,message" "continue,no message"

3.5. Run-Time Environment Behavior

The following illustration describes one *possible* run-time communication scenario between an LMS and a Sharable Content Object. The following diagram illustrates common run-time behaviors that an LMS and SCO may portray. Note that this is only an example and not a required implementation.


Learning Management System
1. Determine SCO to Launch:
• user selected
• automatically using adaptive (prerequisites)
• sequential
using information from the Content Structure defined in the Content Package.
If no more SCOs to launch, finish or exit
2. Determine and initialize the SCO's cmi.core.lesson_mode.
a. Does the LMS support user selectable mode?
b. If cmi.core.lesson_mode is "normal" and cmi.core.credit is "no-credit" change the cmi.core.lesson_mode to "browse"
3. Determine and initialize the SCO's cmi.core.entry. If cmi.core.lesson_status is "browsed" or "not attempted" set the cmi.core.entry to "ab-initio" else set it to "resume"
4. Launch the SCO
Launch
4. Service the API Calls made by the SCO
LMSInitialize("")
LMSGetValue(element)
LMSSetValue(element, value)
LMSGetLast Error()
LMSGetErrorString(errorCode)
LMSGetDiagnostic(errorCode)
LMSCommit("")
LMSFinish("")
API and Datamodel
5. If there is a mastery score in the content structure for the SCO, the LMS can change the lesson_status to "passed" or "failed", depending on the value of the cmi.core.score.raw value compared to the mastery score. If there is no mastery score defined for the SCO in the content structure, the LMS cannot override the lesson_status set by the SCO. If cmi.core.credit is set to "no-credit", the LMS may not change the lesson_status, unless the SCO's lesson_mode is "browse" – the LMS may set the lesson_status to "browsed".
Sharable Content Object (SCO)
1. Find the API Adapter Object
2. Call API.LMSInitialize("")
3. SCO Specific Implementation Behavior (All optional)
3a. Call API.LMSGetValue("cmi.launch_data") -use if supported by SCO
3b. Call API.LMSGetValue("cmi.core.entry") if entry is "resume", get lesson_location, suspend_data and resume at appropriate location in SCO
3c. Call API.LMSGetValue ("cmi.core.lesson_mode") conditional behavior based on mode
3e. Call API.LMSSetValue to record any optional information as needed. (e.g. scores, status, objectives, interactions, etc.)
3f. call API.LMSCommit("")
Learner Interacts with SCO
4. Call API.LMSFinish("")

This page intentionally left blank.

APPENDIX A
ACRONYM LIST

This page intentionally left blank.

Acronym Listing

ADL	Advanced Distributed Learning
AICC	Aviation Industry CBT Committee
API	Application Program Interface
ARIADNE	Alliance of Remote Instructional Authoring & Distribution Networks for Europe
ASCII	American Standard Code for Information Interchange
AU	Assignable Unit
AWT	Abstract Window Toolkit
CBI	Computer-Based Instruction
CBT	Computer-Based Training
CDATA	Character Data
CMI	Computer Managed Instructions
COTS	Commercial Off-The-Shelf
CSF	Content Structure Format
DC	Dublin Core
DoD	Department of Defense
DOL	Department of Labor
DTD	Document Type Definition
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDA	Institute for Defense Analyses
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
ITS	Intelligent Tutoring Systems
LMS	Learning Management System
LOM	Learning Objects Metadata
LTSC	Learning Technology Standards Committee
MIME	Multipurpose Internet Mail Extensions
NGB	National Guard Bureau
OSTP	Office of Science and Technology Policy
PCDATA	Parsable Character Data
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	eXtensible Markup Language

This page intentionally left blank.

APPENDIX B
REFERENCES

This page intentionally left blank.

References

1. Aviation Industry CBT (Computer-Based Training) Committee. (http://www.aicc.org/)

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

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

4. AICC/CMI CMI001 Guidelines for Interoperability Version 3.4. October 23, 2000. Includes: AICC Course Structure Format, AICC CMI Data Model Available at: http://www.aicc.org/.

5. ADL Co-Laboratories. (http://www.adlnet.org/)

6. Institute for Defense Analyses (IDA). (http://www.ida.org/)

7. Executive Order 13111: Using Technology To Improve Training Opportunities for Federal Government Employees.

8. Gettinger, M. (1984) Individual differences in time needed for learning: A review of the literature. Educational Psychologist, 19,15-29.

9. Graesser, A. C., & Person, N. K. (1994). Question asking during tutoring. American Educational Research Journal, 31, 104-137.

10. Bloom, B.S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. Educational Researcher, 13, 4-16.

11. Fletcher, J. D. (2001) Evidence for Learning from Technology-Assisted Instruction. In H. F. O'Neil Jr. and R. Perez (Eds.) Technology Applications in Education: A Learning View. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

13. Gibbons, A.S. & Fairweather, P.G. Computer-based Instruction. (2000) In, S. Tobias and J.D. Fletcher (Eds.), Training and Retraining: A Handbook for Business, Industry, Government, and the Military. New York: Macmillan Gale Group.

14. Suppes, P. (1964) Modern learning theory and the elementary-school curriculum. American Educational Research Journal, 1, 79-93.

15. Carbonell, J. R., "AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction," IEEE Transactions on Man-Machine Systems, Vol. 11, 1970, pp. 190-202.

16. Sleeman, D, & Brown, J. S. (Eds.) (1982) Intelligent Tutoring Systems. New York, NY: Academic Press, 1982.

17. Woolf, B.P., & Regian, J.W. (2000). Knowledge-based training systems and the engineering of instruction. In S. Tobias and J. D. Fletcher (Eds.), Training and retraining: A handbook for business, industry, government, and the military (339-356). New York: Macmillan Reference.

18. Gibbons, A.S. & Fairweather, P.G. (1998) Computer-based Instruction: Design and Development. Englewood-Cliffs, NJ: Educational Technology Publications.

19. Gibbons, A.S. & Fairweather, P.G. (2000) op. cit.

20. IMS Content Packaging Specification Version 1.1.2.
Available at: http://www.imsglobal.org/.

21. IEEE Information Technology - Learning Technology - Learning Objects Metadata LOM: Working Draft 6.1 (2001-04-18).
As referenced by the IMS Learning Resource Meta-data Specification Version 1.2.
Available at: http://ltsc.ieee.org/.

22. IMS Learning Resource Meta-data Specification Version 1.2.
Includes: IMS Learning Resource Meta-data Information Model, IMS Learning Resource Meta-data XML Binding Specification, and IMS Learning Resource Meta-data Best Practice and Implementation Guide
Available at: http://www.imsglobal.org/.

23. ISO 639: This is an international standard for the representation of languages. Version 1 uses two-letter language codes, e.g. 'en' for English, 'fr' for French, 'nl' for Dutch, etc. These language codes are a basis for the IETF registry of language tags, as specified in RFC 1766: Tags for the identification of languages.
Available at: http://www.iso.ch/.

24. ISO 3166: This is an international standard for the representation of country names, e.g. 'BE' for Belgium, 'CA' for Canada, 'FR' for France, 'GB' for United Kingdom, 'US' for United States, etc.
Available at: http://www.iso.ch/.

25. vCard: This standard defines how contact details for people and organizations can be represented.
Available at: http://www.imc.org/pdi/.

26. ISO 8601: This is an international standard that specifies numeric representations of date and time.
Available at: http://www.iso.ch/.

27. World Wide Web Consortium (W3C). http://www.w3c.org/
Includes: Universal Resource Locator, Universal Resource Identifier, Extensible Markup Language Version 1.0, Document Object Model (DOM) Specification.

28. Dublin Core Metadata Initiative. http://www.dublincore.org/.

APPENDIX C
REVISION HISTORY

This page intentionally left blank.

Revision History

SCORM Version	Release Date	Description of Change
1.1	16-Jan-2001	Added more detail to each sub-section of the Run-Time Environment section • Launch • API • Data Model Run-Time Environment Launch changes • Added more detail in describing the launching of SCOs Run-Time Environment API changes • LMSInitialize(“”), LMSFinish(“”) and LMSCommit(“”) now must all take in an empty string as a parameter. • LMSInitialize(“”), LMSFinish(“”), LMSCommit(“”) and LMSSetValue() must now all return a string that can be converted to a CMIBoolean (“true” or “false”) • More detailed information on the descriptions of each of the API calls • Added API Error Code usage section o Defines the scenarios on when to use the different error codes o Added new error codes • Changed findAPI() algorithm to no longer search the sibling frames Run-Time Environment Data Model changes • Added new table to describe in detail all data model elements. • For Run-Time Data Model element changes see Table C-1a • For Run-Time Data Model data type changes see Table C-1b • For Run-Time Data Model Vocabulary changes see Table C-1c

1.2	01-Oct-2001	Changed Figure 3.1a to indicate that SCOs and Assets can be launched.
		Added more detail in the Launch Section to describe the launching of more than just SCOs.
		Added SCO To LMS Communications API State transition diagram and explaination.
		Added more specifics to the API Error code usage table to indicate when certain error codes should be used.
		Added more clarifcation to the following data model elements: • cmi.core.lesson_status • cmi.core.entry • cmi.core.exit • cmi.core.session_time Added more specific definition to those elements that represent scores (cmi.core.score.raw, cmi.core.score.min, cmi.core.score.max, cmi.objectives.n.score.raw, cmi.objectives.n.score.min, cmi.objectives.n.score.max). Added a statement that says: "must be a normalized value between 0 and 100" General Table formatting cleanup
		Added section 3.5 Run-Time Environment Behavior section
		General grammar clean up.

Table C-1a: Run-Time Environment Data Model Element Changes

Data Model Element	Change from SCORM 1.0
cmi.core._children	Changed the data type CMIString256 to CMIString255
cmi.core.student_id	None
cmi.core.student_name	Changed the data type CMIString256 to CMIString255
cmi.core.lesson_location	Changed the data type CMIString256 to CMIString255
cmi.core.credit	None
cmi.core.lesson_status	None
cmi.core.entry	None
cmi.core.score._children	Changed the data type CMIString256 to CMIString255
cmi.core.score.raw	None
cmi.core.score.min	None
cmi.core.score.max	None
cmi.core.total_time	None
cmi.core.lesson_mode	None

cmi.core.exit	None
cmi.core.session_time	None
cmi.suspend_data	None
cmi.launch_data	None
cmi.comments	This element is now used to hold comments from the SCO/Student. A SCO can now invoke LMSGetValue() and LMSSetValue() on this element.
cmi.comments_from_lms	This new element was added to be a place where comments from the LMS and/or instructor can be placed and used by the SCO.
cmi.objectives._count	None
cmi.objectives._children	Changed the data type CMIString256 to CMIString255
cmi.objectives.n.id	None
cmi.objectives.n.score._children	Changed scores to score. Changed the data type CMIString256 to CMIString255.
~~cmi.objectives.n.score._count~~	Removed the support for list of scores for objectives
cmi.objectives.n.score.raw	Removed the list support for score. Each objective can now only have 1 score. Name change from scores to score.
cmi.objectives.n.score.min	Removed the list support for score. Each objective can now only have 1 score. Name change from scores to score.
cmi.objectives.n.score.max	Removed the list support for score. Each objective can now only have 1 score. Name change from scores to score.
cmi.objectives.n.status	Removed the list support for score. Each objective can now only have 1 score. Name change from scores to score.
~~cmi.objectives.n.mastery_time~~	Removed
~~cmi.evaluation._children~~	Removed
~~cmi.evaluation.course_id~~	Removed
~~cmi.evaluation.comments~~	Removed
~~cmi.evaluation.comments.n.time~~	Removed
~~cmi.evaluation.comments.n.location~~	Removed
~~cmi.evaluation.comments.n.content~~	Removed
~~cmi.evaluation.interactions._children~~	Removed
~~cmi.evaluation.objectives_status._children~~	Removed
~~cmi.evaluation.paths._children~~	Removed
~~cmi.evaluation.performance._children~~	Removed
~~cmi.evaluation.lesson_id~~	Removed
~~cmi.evaluation.date~~	Removed
cmi.interactions._children	Added
cmi.interactions._count	Added

cmi.interactions.n.id	Changed the data type CMIString256 to CMIString255
cmi.interactions.n.objectives._count	Added
cmi.interactions.n.objectives.n.id	Changed the data type CMIString256 to CMIString255
cmi.interactions.n.time	None
cmi.interactions.n.type	None
cmi.interactions.n.correct_responses._count	Added
cmi.interactions.n.correct_responses.n.pattern	None
cmi.interactions.n.weighting	None
cmi.interactions.n.student_response	None
cmi.interactions.n.result	None
cmi.interactions.n.latency	None
cmi.student_data._children	Changed the data type CMIString256 to CMIString255
~~cmi.student_data.attempt_number~~	Removed
cmi.student_data.mastery_score	None
cmi.student_data.max_time_allowed	None
cmi.student_data.time_limit_action	None
~~cmi.student_data.attempt_records._count~~	Removed
~~cmi.student_data.attempt_records._children~~	Removed
~~cmi.student_data.attempt_records.n.lesson_score._children~~	Removed
~~cmi.student_data.attempt_records.n.lesson_score.raw~~	Removed
~~cmi.student_data.attempt_records.n.lesson_score.min~~	Removed
~~cmi.student_data.attempt_records.n.lesson_score.max~~	Removed
~~cmi.student_data.attempt_records.n.lesson_status~~	Removed
~~cmi.student_data.tries_during_lesson~~	Removed
~~cmi.student_data.tries.n.score.raw~~	Removed
~~cmi.student_data.tries.n.score.max~~	Removed
~~cmi.student_data.tries.n.score.min~~	Removed
~~cmi.student_data.tries.n.status~~	Removed
~~cmi.student_data.tries.n.time~~	Removed
~~cmi.student_demographics._children~~	Removed
~~cmi.student_demographics.city~~	Removed
~~cmi.student_demographics.class~~	Removed
~~cmi.student_demographics.company~~	Removed
~~cmi.student_demographics.country~~	Removed
~~cmi.student_demographics.experience~~	Removed
~~cmi.student_demographics.familiar_name~~	Removed
~~cmi.student_demographics.instructor_name~~	Removed
~~cmi.student_demographics.title~~	Removed
~~cmi.student_demographics.native_language~~	Removed
~~cmi.student_demographics.state~~	Removed
~~cmi.student_demographics.street_address~~	Removed
~~cmi.student_demographics.telephone~~	Removed
~~cmi.student_demographics.years_experience~~	Removed
cmi.student_preference._children	Changed the data type CMIString256 to CMIString255
cmi.student_preference.audio	None
cmi.student_preference.language	Changed the data type CMIString256 to CMIString255

~~cmi.student_preference.lesson_type~~	Removed
cmi.student_preference.speed	None
cmi.student_preference.text	None
~~cmi.student_preference.text_color~~	Removed
~~cmi.student_preference.text_location~~	Removed
~~cmi.student_preference.text_size~~	Removed
~~cmi.student_preference.video~~	Removed
~~cmi.student_preference.windows.n~~	Removed
~~cmi.paths.n.location_id~~	Removed
~~cmi.paths.n.time~~	Removed
~~cmi.paths.n.status~~	Removed
~~cmi.paths.n.why_left~~	Removed
~~cmi.paths.n.time_in_element~~	Removed

Table C-1b: Run-Time Environment Data Type Changes

~~CMIDate~~	Removed
CMIBoolean	Changed the types to all lower case (“true”, “false”)
CMIFeedback	Added more information, to define the usage of the CMIFeedback data type
~~CMILocale~~	Removed
~~CMISIdentifier~~	Removed
CMIString255	Changed CMIString255 – maximum length of 255 characters
CMITime	Now can have at most 2 digits following the optional decimal (i.e. 34.43)
CMITimespan	Now can have at most 2 digits following the optional decimal (i.e. 34.43)

Table C-1c: Run-Time Environment CMIVocabulary Changes

Exit	Added an additional vocabulary “” – empty string
~~Why Left~~	Removed
Credit	Changed “no credit” to “no-credit”
Entry	Added an additional vocabulary “” – empty string
Time Limit Action	Changed all vocabulary choices
Interaction	Changed “multiple choice” to “choice” Changed “fill in the blank” to “fill-in” Changed “simple performance” to “performance”