Welcome to the Developing SCORM Compliant Content, Technical Issues and Solutions session, presented by John Toews of the Academic ADL Co-Lab at the New Media Center's Online Conference on Learning Objects. An audio script is available.
Need SCORM?

Don’t wait till the end of the development process!

Simply implementing LMSInitialize and LMSFinish doesn’t really take advantage of SCORM!

» Tracking
» Moving
» Locating
» Sequencing

There are two points in particular that I would like to emphasize during this session. While the technological issues can be easily taught and shared, changing the methods people do work by takes more attention.

Too often content development processes ignore SCORM, along with its benefits and constraints, until the end or near end of those processes.

In addition, many times only the SCORM minimum functions LMSInitialize and LMSFinish are implemented. These give content none of the advantages and opportunities that SCORM offers.

Think about SCORM when planning a course (or learning object, or other content) and continue to keep it in mind during the development process. Involve your developers and instructional designers – ensure that they understand the possibilities and implications!
On the previous slide I talked about the advantages and opportunities that SCORM offers.

One thing to remember about SCORM is that it allows you to implement features in a standardized way, that will be supported by numerous LMSs and other tools. For example, you can move your SCORM compliant content from LMS to LMS, usually with a short import process. And while you may have been able to allow users to bookmark content before SCORM, with it you can have consistent bookmarking that can work across LMSs.

As more and more LMSs begin to support SCORM, they also are beginning to integrate its capabilities, such as scoring, into their native functionality, creating a more seamless environment.
I want to point out that I will be getting fairly technical during this session. In order to understand everything presented, I suggest that you are comfortable editing HTML and JavaScript, as well as have a familiarity with SCORM itself.

I’ve listed some resources if you aren’t up to speed. The sample SCORM package is a small file with commented code to explain what is happening. The Academic ADL Co-Lab’s SCOurse we have made available is probably not technical enough for this session, but it is a good introduction to the concepts of SCORM.
When beginning to plan your SCOs, whether there is one in a Learning Object or many in a course, an important design decision is what size a SCO should be. The SCORM does not prescribe how large or small a SCO should be, either by file size, pages or time on task – it only says that a SCO is the “smallest logical size of content that one might desire to have tracked by an LMS at run-time.”

In practice you should simply make a choice based on what makes sense in your project – taking into consideration the potential reusability of the content, the features you have chosen to be implemented, and pedagogical concerns.

Note that it might take some practice using SCORM to be able to make these decisions without having to revisit them later. The Academic ADL Co-Lab twice reworked our SCOurse project before finding a format that worked to our liking.
In the process of several content development projects, the Academic ADL Co-Lab has made certain design and development decisions. One was the directory structure pictured on this slide, in which a root directory manages the entire course, while subdirectories contain individual SCOs.

This setup allows us to package any number of individual SCOs, along with global resources that require only one instance within the entire development site.
Another useful decision we made was to create scripts to manage navigation within the SCO. While SCORM handles intra-SCO navigation, and in fact requires that you don’t link between SCOs, the content developer must still ensure that inter-SCO navigation is in place. Of course, this only applies to multi-page SCOs.

In the Academic Co-Lab’s SCOurse, scripting places the back and next buttons on the screen, disabling them if necessary, as well as inserting a bar of rectangles that have a link to each page in the SCO. The scripting also changes the rectangle color to red if that page has been viewed, giving a visual sense of completion to the learner.
By combining our frames based templating system (which is described later), along with the directory structure and automatic navigation, we are able to display content to the user in different ways, with different functionality.

In this slide you see the same content, albeit different revisions, in two separate versions.

In the top picture, you see our SCOurse content in a custom viewer created for demo and non-SCORM applications.

In the lower picture, you see our SCOurse content in a LMS that supported importing of SCORM content.
The SCORM requires that SCOs use an API to communicate with the host system, which is usually an LMS, to initiate communications (usually on page load), and finish communications (usually on page unload).

For single page SCOs there is no issue with unloading the page since it always means the communications are finished and therefore LMSFinish should be run. However, problems arise with multi-page SCOs as the user moves from page to page, as the content should not call LMSFinish unless the user is leaving the content as a whole, and not simply the current page. In addition, LMSInitialize should only be run once per SCO, and in this model it is run at the start of every page.

As you can see in the graphic, LMSFinish is run when the user unloads page one by proceeding to page two of the content, when it really should only be run when the user leaves the content as a whole. Also, LMSInitialize is run when page two is loaded, making the SCO non-conformant.

Browsers don’t support this type of functionality with standard JavaScript functions, so a workaround is necessary.
When setting out to find an answer to the multi-page SCO problem, the Academic ADL Co-Lab identified a few workarounds.

One was to set a JavaScript variable before each link on the page – the variable would indicate to the onUnload function that the user is continuing instruction and not leaving the content, so LMSFinish would know not to run. Unfortunately this would involve editing every link in the content, and therefore could have reliability problems.

A second idea proposed was a session cookie that would track the status of LMSInitialize. This solves the issue of LMSInitialize running multiple times, however LMSFinish is still a problem on the unload of pages.

Another method discussed was to launch the content itself in a new window, with the SCORM API functionality trapped in the original window (also known as the parent). We eventually decided against this because of the likelihood the user would see a blank page in their LMS window and lose track of the actual content.

The solution we currently favor is a frameset, where a “hidden” frame manages the SCORM API functions and the content would be presented in a larger frame. In this method, the content needs no editing of actual links but instead requires implementation of SCORM functions LMSGetValue and LMSSetValue by passing them through the hidden frame.
In this graphic you see our frameset solution visualized.

The top, persistent HTML frame is loaded only once, when the content starts, and is unloaded only once, when the content is closed. Therefore the standard JavaScript functions `onLoad` and `onUnload` can be used without any issues. In the bottom frame, the actual content is represented by three white pages, and the user is free to move between these pages without running `LMSInitialize` multiple times and `LMSFinish` before it is needed.
SCORM requires its communications functions to be implemented in JavaScript, and LMSs typically implement the SCORM API in a Java applet. The technology that allows JavaScript to talk to Java applets is called LiveConnect, which is Netscape’s term. Not all browsers support the LiveConnect technology, which means that SCORM communications cannot occur – this is specifically a problem on the Macintosh OS. Neither Internet Explorer for the Mac or Apple’s newest browser, Safari, implement LiveConnect. Note that ActiveX controls typically suffer from the same restrictions as Java applets.
Flash Issue Defined

- Similar to SCORM API issue

- For Flash to be 100% SCORM (bi-directional communication) requires both:
  - LiveConnect
  - Flash player implementing “fscommand”

- Minimal SCORM functionality (outgoing communications only) more widely available – should developers use?

Similar to the SCORM API requiring LiveConnect, in order to use all of SCORM’s features with Flash, a browser must implement LiveConnect and the Flash plugin must support the correct technology, called fscommand.

A Flash movie can support outgoing information, in other words LMSSetValue, on nearly all browser and OS combinations via the geturl ActionScript function. However, to be fully functional and support incoming information, in other words LMSGetValue, the browser and Flash player must implement the correct technology.
The SCORM Data Model has some issues as well, and I put this in quotes because it was a foreseeable problem. In the SCORM 1.2, LMSs are not required to implement the full data model, which sometimes leaves very useful optional elements out, such as those that track both objectives and interactions. This leads to developers implementing custom functions that essentially use mandatory elements for purposes they weren’t designed for.

Thankfully, the next version of SCORM, due out soon, will solve these issues by requiring the full implementation of the data model.
 Threaded discussions are available throughout the New Media Center's Online Conference on Learning Objects and, I will be monitoring them.

In addition, several resources are available through iCohere.

The Academic ADL Co-Lab has signed partner agreements among many institutions and consortia of higher education. We maintain a private partners-only website among other benefits – and you may already be a partner.

You can also contact me via email.