Building Standards-Conformant, Accessible Learning Objects with Macromedia Flash™ MX

by Jay Heins and Bob Regan

July 2002
SCORM-conformant content is context independent. This means that it will work in a variety of situations, whether as part of a stand-alone course, a learning management system, a fully online course, or a hybrid setting.

You may learn more about ADL and download the current SCORM specifications from: http://www.adlnet.org/.

Creating Accessible Macromedia Flash Learning Objects

Creating accessible Learning Objects with Macromedia Flash MX is easier than ever before. With new features built into the Macromedia Flash Player and the Macromedia Flash MX authoring tool, designers are able to present content to users with disabilities in ways not previously possible.

One of the challenges of creating accessible Macromedia Flash content is understanding what exactly is meant by ‘accessible.’ As was discussed earlier, accessibility standards generally do not address the specific techniques and concerns of Macromedia Flash design. Instead, it is up to the individual designer to evaluate how well the content works with assistive technology and how usable it is for people with disabilities.

This section provides a basic set of guidelines for creating accessible Macromedia Flash content. These guidelines are not intended to be comprehensive. Instead, they are intended to provide designers with a clear starting point for accessible Macromedia Flash design. This section will present and explain the following guidelines:

- Provide text equivalents
- Reveal structure
- Ensure a coherent reading order
- Design for keyboard access
- Provide captions

Provide Text Equivalents

By default, Macromedia Flash Player 6 exposes all text elements to screen readers. The screen reader can read all static text and dynamic text symbols on the stage. Designers and developers do not need to make any modifications. For example, the simple Macromedia Flash banner below was created by adding text to the stage.
Movie clips and buttons are also exposed to the screen reader; however, screen readers are not able to view the contents of a picture and describe them for the user. Instead, the designer must assign a text equivalent for graphic elements.

In the example below, the Macromedia Flash MX logo is displayed. The letter ‘f’ in the logo is not actually text, but a graphic that is part of the design as a whole.

Because the letter ‘f’ is not text, a screen reader will not read it. Moreover, the letter ‘f’ alone does not provide enough information about the banner to be helpful for screen reader users. In this case a text equivalent should be provided. To add a text equivalent, use the Accessibility panel. First, make sure the logo is selected. Before adding a text equivalent to this item, the object must be saved as a symbol in the library. Since text equivalents are not supported for graphic symbols, save the object as a movie symbol or a button.

To bring the Accessibility panel to the front, press F9.

Notice the two fields titled “Name” and “Description.” In general terms, the Name field is used for shorter text equivalents and the Description field is used for longer ones. (This parallels the use of the ‘alt’ and ‘longdesc’ attributes in HTML.) In more practical terms, the difference is less important as the Window-Eyes screen reader reads both by default.
Deciding on the proper text equivalent is not always easy. In most cases, it is wise to consider the purpose of the image rather than giving it a merely literal description. In the example above, a description may read, “Dark rectangle with letter f in red.” This may not be particularly helpful information to a screen reader user. By contrast, the phrase “Macromedia Flash MX” might be a more meaningful equivalent for the logo.

In this case a description would not likely be necessary. Descriptions are best used in cases where the text equivalent needs to be longer than about 50 characters. One thing to keep in mind is that the description is generally read following the name. If the content does not flow when the two are read together, the designer or developer may choose not to use a name, only a description of the object.

**Text Equivalents for Animation**

When a screen reader encounters a piece of Macromedia Flash content, the screen reader loads the current state of the movie and notifies the user. With the Window-Eyes screen reader, the user hears, “Loading...load done.” Once a piece of content has been read, the screen reader moves on to read other parts of the Macromedia Flash content and the rest of the page.

A basic feature of Macromedia Flash content is that it may change over time. As the content changes, Macromedia Flash Player 6 sends a signal to the screen reader notifying it that there has been a change. When the screen reader receives this notification, it automatically returns to the top of the page and begins reading it again.

The following example illustrates the serious implications of Macromedia Flash content created without consideration for users of screen readers. A poorly designed logo with spinning letters placed at the top of the page might loop constantly through a few frames. When Macromedia Flash Player encounters this banner, it will send repeated notifications to the screen reader of changes in the content, and the screen reader will continually return to the top of the page. This problem can seriously erode the experience for screen reader users.

To address this issue, the design may hide the animation from assistive technologies. Macromedia Flash MX allows designers and developers to assign a text equivalent for an entire movie or for a collection of objects within a movie. Designers and developers might choose to provide a text equivalent for Macromedia Flash content for one of two reasons.

First, animations are often used to illustrate visual relationships among elements on the screen. Adding text equivalents to the individual elements may not provide a sufficient description of the relationships among the elements. For example, in an animation of the solar system, the designer or developer might add text equivalents to the planets; however, these text equivalents would not convey information about how the planets move in relation to one another. A text equivalent for the entire movie could provide a better description of this relationship.
Second, text equivalents may be used to hide animated content from the screen reader. In the example of a rotating banner ad, a single text equivalent for the entire ad would convey the content of the advertisement to the user and prevent the screen reader from constantly returning to the top of the page.

**Reveal Structure**

When a sighted user first encounters a Macromedia Flash screen, her eyes will normally scan immediately through the visual elements. She’ll read the text, process the images, and identify the buttons. Experienced designers know how this process works for most users and take advantage of it. Important elements are organized into rows and columns, usually starting at the top left. Color is used to group elements and also to distinguish them. Visual cues are used to identify the relationships among items and, ideally, what can be done with them.

For the blind user, none of this visual design is apparent. She is limited only to the text that is read to her. That doesn’t mean there aren’t ways to convey the same information as the visual design, but they are necessarily different. While the designer relies on the sighted user to scan the screen and quickly assess its meaning and how she can interact with it, it is imperative to use accessible text to orient the blind user to both structure and function.

Macromedia Flash MX provides a simple way to do this. It is possible to attach a text equivalent to an entire Macromedia Flash movie that briefly describes the design of the movie and its major elements. The description should introduce and explain both the purpose of the screen and its layout.

In the example below, a simple registration form has been created in Macromedia Flash MX. This form includes seven form fields and a Submit button.

![Mailing Request Form](image)

It is possible to add or modify the accessible name and description for the entire movie by going to the _root timeline, and deselecting all options. The Accessibility window should change to indicate that only the name and the description can be modified.
In this case, the designer might enter the following as a description for the form: “This form is used to request information via mail.”

An alternate method is to attach a description to a movie clip in the upper left corner of the screen. This clip can be small, but it must contain some art; empty movie clips, even those with accessibility properties, are not accessible using screen readers. If the clip is placed in the far upper-left-hand corner, its description will be read first whenever the screen is loaded. This allows more control over when the text might be read. For instance, a different announcement could be read every time there is a significant change to the Macromedia Flash screen. This method is also more reliable since Window-Eyes tends not to read the description field the first time through.

Ensure a Coherent Reading Order

Designers and developers need to consider how screen readers will move through the contents of a Macromedia Flash movie. The default reading order of Macromedia Flash content, instead of moving in an orderly fashion from left to right and top to bottom, might be unpredictable.

This problem may be addressed in one of two ways. First, using simple trial and error, a designer may use a screen reader to listen to the contents and adjust the placement on the stage until the content is read in a sensible order. This method works well with movies that use a small stage size. The larger the stage, the more complex the reading order becomes.

A more precise method for controlling the reading order uses ActionScript. In the example below, two sets of buttons are presented visually as a row and a column. In cases where the designer would like the column of buttons to be read before the row, a tab order would need to be specified. A custom tab order for content within a Macromedia Flash movie may be specified using ActionScript.
Tab order can be assigned to dynamic text objects, buttons, movie clips, and input text fields. To assign tab order to a static text object, you must first convert it to a dynamic text object. Next, each instance of the item on the stage must be given a name. This is different from the symbol name. The instance name may be assigned using the Property inspector. See the highlighted area of the Property inspector below. It is important that instance names be assigned for each button as well as for the dynamic text within that button.

Once you have assigned the instance names, you need simply to assign a number to the tabIndex property. At the root level of the movie, the ActionScript code below is entered. In the first line, note that myOption1 refers to the instance name of the button and myHomeBut refers to the instance name for the dynamic text within the button. It is acceptable for all of the dynamic text elements to use the same instance name because they exist within separate buttons.

```actionscript
_root.myOption1.tabIndex = 1;
_root.myOption2.tabIndex = 2;
_root.myOption3.tabIndex = 3;
_root.myOption4.tabIndex = 4;
_root.myOption5.tabIndex = 5;
_root.myOption6.tabIndex = 6;
```

Please make a note of the following exception: if you provide a custom tab order for a given frame in your movie and you do not specify a tab position for one or more of the accessible objects in that frame, the Macromedia Flash Player will disregard your custom tab order when users are using a screen reader. It is important to keep in mind that providing a complete tab order for all accessible objects requires assigning tab indexes to dynamic text objects and movie clips, even when these objects are not tab stops (i.e., they never receive the input focus).
Design for Keyboard Access

Content designed in Macromedia Flash MX should work as well for users who rely on the keyboard as for those who use the mouse. Many users with disabilities who are not able to use a mouse will rely on the keyboard or a modified keyboard to interact with controls. In addition, keyboard access to controls is also important for users who are blind. If one cannot see the cursor on the screen, using the mouse often proves impossible.

Buttons and Forms in Macromedia Flash Player 6

By default, the Macromedia Flash Player makes buttons available to assistive technology. With a screen reader, the sample button below would be read as follows. First, the word ‘button’ is read to provide a cue to the user that there is an action associated with this element. Next, the text equivalent is read to the user—in this case, the word “home.” Finally, the user may activate this button by pressing Enter on the keyboard.

As with buttons, Window-Eyes users are also notified of form fields. In the example below, the screen reader might read, “Electronic Registration. Edit Box: Name. Edit Box: Address. Button: Send.”

Auto-Labeling in Macromedia Flash MX

In both of the examples above, the name and description do not have to be specified in the Accessibility panel. Macromedia Flash MX uses its auto-labeling feature to generate the labels automatically. It looks for a text label placed on top of, inside, or near a button or text field. When the Macromedia Flash Player discovers an arrangement like this, it assumes that the text object is a label for the button or text field.
In some cases it may be desirable to specify a text equivalent that is different from the text label. This may be appropriate when using a short text label, or when the label is split up or placed away from the button or form object.

In the example below, notice that the name fields are grouped visually. Putting the word *name* at the top implies that the labels *first* and *last* signify “first name” and “last name.” Using assistive technology, this relationship may not be obvious. To clarify these labels, the text equivalents “first name” and “last name” may be specified using the Accessibility panel.

In this case, the text labels should be hidden from the assistive technology. Assistive technologies might otherwise read the text label and the text equivalent as, “Name. First. Edit Box: First Name.” If the designer or developer converts the text objects to a movie symbol and then deselects the Make Object Accessible option, the assistive technology will read only the text equivalent. It would then read, “Edit Box: First Name.”
Using the Down State in Buttons

In Macromedia Flash MX, designers and developers are able to create buttons with animation and scripting to build complex interactions; however, screen readers and other assistive technologies generally work best with buttons that are relatively simple.

One common technique is to place text and graphics in the down state of a button in Macromedia Flash MX. When the button is pressed, the text and graphics are revealed. The Macromedia Flash Player does not expose the contents of the down state, except for a single text element. Navigational cues, menu bars, and other content are not made available using this technique.

Another common technique among Macromedia Flash designers and developers is the use of invisible buttons placed over a background with text. This practice is discouraged for accessibility purposes. The Macromedia Flash Player notices only buttons that have content; thus, invisible buttons are not made available to assistive technologies.

Notes on Keyboard Shortcuts

The keyboard shortcut field on the Accessibility panel allows designers and developers to add keystrokes to individual buttons and form elements. To provide a keyboard shortcut, ActionScript must be used to detect keypresses by the user during movie playback. (For more information, see “Capturing keypresses” in Macromedia Flash Help.) Keyboard shortcut functionality also depends on the assistive technology and software used.

For keyboard shortcuts, use the following conventions:

- Spell out key names, such as Ctrl or Alt.
- Use capital letters for alphabetic characters.
- Use a plus sign (+) between key names, with no spaces; for example, Ctrl+A.

Creating Accessible Forms

Macromedia Flash MX allows designers and developers to create accessible forms to build dynamic web applications. The simple form components, such as input text, radio buttons, check boxes, and component buttons, work with assistive technologies; however, more complex objects like the scroll bar, the combo box, and the list box do not work with assistive technologies. Although Macromedia plans to distribute accessible versions of these objects in the future, they are not available as of the launch of Macromedia Flash MX.
SCORM-Conformant Learning Objects

Now that we have discussed how to create an accessible Learning Object in Macromedia Flash MX, we need to describe the steps necessary to make it SCORM-compliant. While making Macromedia Flash content accessible is something that designers and developers generally do as they go along, adding information to support the SCORM specifications is most often done at the end of the process.

This section begins with a more detailed explanation of the SCORM standard. We will examine the principal components and their functions. We also include a description of the elements that make up the SCORM specifications: the Content Aggregation, the Run-time Environment, and Meta-Data.

Later, we will present more specific information and examples of code and functions involved in creating SCORM-conformant Macromedia Flash MX Sharable Content Object (SCOs).

Details of the SCORM Specifications

In order to describe the process of building and deploying SCORM-conformant Learning Objects accurately, it is necessary to have a clear understanding of the technology, component parts, and conformance requirements of the SCORM standard.

Component Parts

The two technology component parts that make up SCORM are the SCORM-conformant LMS/LCMS and the learning content that is packaged as SCOs.

In short, the LMS (which manages learner information) and LCMS (which manages content-related information) launch and communicate with SCOs and can interpret instructions that tell them which SCO comes next. It is the responsibility of the management system to control the navigation between the SCOs.

A SCO is a standardized form of reusable Learning Object. Each SCO is a self-contained, modular unit of learning content which contains one or more assets of various media like text, sound, web pages, and assessment objects. The size and scope of a SCO is functionally defined by its ability to initialize and terminate communication with an LMS/LCMS. In practice, shorter and more focused SCOs tend to be more flexible and re-usable for different audiences or courses.

A SCO is required to adhere to the rules of the SCORM Run-Time Environment. Each SCO communicates various instructions and requests to the management system. At minimum, the SCO must signal back to the LMS/LCMS when it begins and ends. The SCO may also transfer information back and forth to track learner data like name, ID, test scores, language, and such. It is the responsibility of the SCO to handle the navigation internal to the presentation of content.
What is Content Aggregation?

SCOs can be used as building blocks to create larger ‘packages’ of SCOs. Three things must be done to create a larger unit of learning from SCOs.

- The SCOs must be identified and organized into a structure.
- The LMS must be given instructions regarding the ordering of the SCOs.
- The SCOs and instructions must be bundled into a portable package.

This process is called content aggregation.

Content aggregation is the process by which the assembled learning resources are identified, consolidated and mapped into a coherent, structured learning unit. The specifications dictate the format and structure that all SCORM-conformant content must take. Again, this structure helps to facilitate opportunities for reuse. In this way, any SCORM-compatible LMS/LCMS is interoperable and can load, deliver, and track data that has been developed for another learning management system.

A manifest file is used to describe the collection of resources used and the organization of the course.

For more information about content aggregation, please see SCORM 1.2 Content Aggregation Model, available at the ADL website at http://www.adlnet.org/.

What is the Run-Time Environment?

The SCORM Run-Time Environment (RTE) deals with the launching, communicating, and tracking of content between the course and the LMS/LCMS. Common methods for starting learning experiences and communicating instructions and data provide the means by which learning resources can be reusable and interoperable across multiple LMS/LCMS systems.

HTML pages containing learning content communicate with the SCORM Application Protocol Interface (API) contained in the LMS/LCMS via JavaScript embedded in a web page.

The RTE consists of three categories of Run-Time commands: execution state, data transfer, and SCO state. The range of these commands is used for starting and ending communication, tracking data, and diagnosing problems with SCORM-conformant courses. Execution state consists very simply of two commands: LMSInitialize and LMSFinish to open and close communications with the management system. Data transfer states include LMSGetValue, LMSSetValue, and LMSCommit. These command states allow a SCO to read, write, and permanently write run-time variables to the LMS database. The third set of commands, state management, helps with diagnosing any errors that may have been returned from the LMS. These states include LMSGetLastError, LMSGetErrorMessage, and LMSGetDiagnostic.

Table 1: SCORM API communications functions. Source: E.R. Jones, ADL’s SCORM Course at http://www.scorm.tamucc.edu

<table>
<thead>
<tr>
<th>Command Category</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution State</td>
<td>Function Call</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LMSInitialize(&quot;&quot;)</td>
<td>Opens communications between the SCO and LMS</td>
<td></td>
</tr>
<tr>
<td>LMSFinish(&quot;&quot;)</td>
<td>Closes communications between the SCO and LMS</td>
<td></td>
</tr>
<tr>
<td>LMSGetValue(element)</td>
<td>Allows a SCO to read a student meta-data value from the LMS database</td>
<td></td>
</tr>
<tr>
<td>LMSSetValue(element,value)</td>
<td>Allows a SCO to write a student meta-data value to the LMS database</td>
<td></td>
</tr>
<tr>
<td>LMSCommit(&quot;&quot;)</td>
<td>Requires the LMS to write any cached student meta-data values to its database</td>
<td></td>
</tr>
<tr>
<td>LMSGetLastError(&quot;&quot;)</td>
<td>Returns the number associated with the last error condition from the LMS</td>
<td></td>
</tr>
<tr>
<td>LMSGetErrorString(ErrNum)</td>
<td>Returns a string describing the last error from the LMS associated with the error number &quot;ErrNum&quot;.</td>
<td></td>
</tr>
<tr>
<td>LMSGetDiagnostic(parm)</td>
<td>Returns a LMS-specific diagnostic error string for the error condition associated with &quot;parm&quot;</td>
<td></td>
</tr>
</tbody>
</table>


**What is Meta-Data?**

Meta-data is *data about data*. It is the common method used to describe resources in a consistent manner. In SCORM, there are two kinds of meta-data used: *course-, SCO-, and asset-level meta-data*, and the *data-model*.

Several layers of meta-data exist to help with the indexing and searchability of learning content and SCOs. Course-, SCO-, and asset-level meta-data is used to help with the search, discovery, and reuse of the various levels of granularity of learning content. When used within and across online repository systems, well-crafted meta-data facilitates the finding and extraction of learning content for reuse.

Another kind of meta-data exists in SCORM that tracks a set of variables and stores the values used by the run-time environment. The data-model, or learner meta-data, tracks learner progress, scores, status, and a range of other variables.
More information about course-, SCO-, and asset-level meta-data can be found in SCORM 1.2 Content Aggregation Model. Find details about the data-model in SCORM 1.2 Run-Time Environment. Both are available from the ADL website at http://www.adlnet.org/.

Conformance Requirements

In order to meet minimum SCORM specification requirements, a SCO must be able to locate the API object provided by an LMS/LCMS, issue the LMSInitialize command, and issue the LMSFinish command.

A second, higher level of conformance is recognized when a SCO is able to read and write information to the LMS/LCMS and its database. These elements, LMSGetValue and LMSSetValue, are used to “get” and “set” the information that is used to track data and enhance the learning experience.

It is important to note that only one SCO can be loaded from the LMS/LCMS at one time for one user. This rule prevents either the LMSInitialize or LMSFinish commands from being issued twice in a row for that user. If this were to happen, an error state would result.

In order to maintain the stand-alone capacity of a SCO, two rules must be observed: one SCO cannot directly link to another SCO, and one SCO cannot access data from another SCO.

Description of the Run-Time Process

When we examine the component parts and the interactions of the SCORM, taken together, a dynamic illustration of the run-time environment emerges:

1. The LMS/LCMS launches a SCO.
2. The SCO finds the API and opens communications with management system using LMSInitialize method.
3. The SCO and management system communicate information and variables back and forth, as required or requested by the SCO. Data transfer states include communicating student meta-data like scores, completion, and time taken. State management reports errors, if any are encountered.
4. When the lesson is complete, the SCO ends communications with management system by invoking the LMSFinish command.
5. The next SCO is launched and the process repeated, or the session is ended.
How to Build SCORM-Conformant Learning Objects

This section demonstrates how to build SCORM-conformant Macromedia Flash MX SCOs. We will be employing several Macromedia tools to help with the process. In addition to Macromedia Flash MX, we will also use Macromedia Dreamweaver® MX and the Extensions Manager.

Macromedia Extensions are developed by partners and customers; they allow developers to add new features easily to certain Macromedia products. The Extension Manager controls the download, installation, and tracking of extensions. We recommend that users download and install Extension Manager 1.5. This version fully supports the new features of the MX suite of tools. If you need help installing the Macromedia Extension Manager, see: [http://www.macromedia.com/exchange/](http://www.macromedia.com/exchange/).

This section will use two extensions. The first, FS SCORM, is available from the Macromedia Flash Exchange. This extension uses FS commands to pass SCORM data back and forth between the Macromedia Flash object and the LMS/LCMS. The second, Manifest Maker for ADL SCORM, is available from the Dreamweaver exchange. This extension helps the designer to compile the SCORM manifest file.
Launching, Communicating with, and Tracking Content in a Web-based Environment

As we begin discussing the process of launching, communicating, and tracking content with Macromedia Flash MX Learning Objects, it is important to distinguish between the two components that facilitate the process. Note that one component is internal to the Macromedia Flash MX Learning Object, while the other is external to the file and resides in the HTML, or wrapper, page.

The first component is the code internal to the Macromedia Flash Learning Object, written in ActionScript, that makes the necessary API calls. The ActionScript function `fscommand` is used to run the JavaScript function in the host HTML page. It is the developer’s responsibility to add the code to enable communications to the host page.

The external JavaScript code in the HTML page in turn connects to the LMS/LCMS. Publishing a Learning Object with the appropriate JavaScript and HTML code is accomplished using the FS SCORM extension. This extension provides all the needed code to connect and communicate with the LMS/LCMS.

Macromedia Flash MX Learning Object Communication using fscommand

In order to enable communications between the LMS/LCMS and the SCO, the API functions must be installed in the Macromedia Flash MX Learning Object. The first step is to include in the LO the required `LMSInitialize` and `LMSFinish` API calls. `LMSGetValue` and `LMSSetValue` are optional functions that support the communications of the data-model.

**LMSInitialize and LMSFinish**

Recall that in order to be SCORM-conformant, a SCO must, at minimum, be able to open and close communications with the LMS/LCMS.

In the case of a Macromedia Flash MX SCO, the Initialize and Finish commands must be communicated to the SCORM API. This is done with the following ActionScript code:

```actionscript
function initializeLMS() {
    fscommand ("LMSinitialize",""};
}
```

and:

```actionscript
function finishLMS() {
    fscommand ("LMSFinish",""};
}
```
In the sample asset provided for this whitepaper, the Initialize command is set to be activated automatically upon loading of the SCO, whereas the Finish command has been attached to a button at the end of the SCO content.

LMSGetValue, LMSSetValue and Data-Model

After a SCO is able to open and close communications successfully, the next task is to enable the SCO to communicate its Run-Time variables. This is done using the get and Set method and the data-model.

Get and set uses the same method as the Initialize and Finish commands and employs the data-model to communicate variables to the LMS/LCMS. For example, the code shown below collects the student name parameter from the LMS/LCMS and places it in the get_name dynamic text field.

```
function getvalueLMS() {
  var student_name = "cmi.core.student_name," + "get_name";
  fscommand("LMSGetValue",student_name);
}
```
Publishing Learning Objects using Macromedia Flash MX FS SCORM Extension

The purpose of the FS SCORM template is to make it easy for developers to create and publish Macromedia Flash MX Learning Objects. FS SCORM ‘wraps’ the Macromedia Flash object with the required HTML and JavaScript that enables it to locate and communicate with the LMS/LCMS using the SCORM API object.

Using the Macromedia Flash MX FS SCORM Extension:

Once the Extension Manager and FS SCORM extension have been installed, proceed with these steps:

1. Launch Macromedia Flash MX and open the Learning Object you wish to publish.
2. Select File > Publish Settings.
3 Ensure that both the Flash (.swf) and HTML (.html) options are selected. This will present three tabs in the dialogue box.

4 Choose the HTML tab. Select FS SCORM from the Template drop-down list. Choose the Flash tab and select a Macromedia Flash Player version. Click Publish.
Using Dreamweaver, open up and examine the code and structure of the resulting HTML file. Notice that if you open the HTML file directly in a browser window, an alert message announces, “JavaScript Warning: API object not found in window or opener.” This indicates that JavaScript is attempting without success to connect to the LMS/LCMS API adapter.

How to Group your Media into Structured Learning Content

Once the Macromedia Flash MX SCO is built, you will need to organize the course meta-data in order to facilitate distribution, discovery, use, and management of the learning content. This section will provide an overview of how Learning Object content is structured using a manifest file.

A manifest file is much like an inventory or packing slip used in parcel shipment. The manifest file collects and organizes the following information:

- Course description
- Course organization
- Course resources or files

XML is the format used for the manifest file. Every SCORM-conformant course must include an XML manifest file named 'imsmanifest.xml' in the top level of the course directory.

Creating the manifest file and supporting documents is easily accomplished using the Manifest Maker for ADL SCORM extension for Dreamweaver. This extension makes a simple, minimal ADL Content Packaging Manifest based on the published 1.2 specification. It also copies the XSD control files to the local site folder.

As it creates a manifest, Manifest Maker makes the following assumptions:

- The site name is the course name for the manifest Table of Contents.
- All of the HTML files in the site are valid entry points for the Table of Contents.
- Names of subdirectories and HTML documents become titles in the Table of Contents.
- Any file in the site whose name also appears in an HTML document is considered a supporting file (including links).
- The order of items in the Table of Contents is alphabetical, with files before subdirectories. (This organizes the SCOs in your course alphabetically.)
- Files referenced programmatically, such as roll-overs, will not be automatically included.
- Any referenced Authorware or CourseBuilder support files will be included as support files (*.aam, *.aas, /scripts, *js).
- The manifest file itself (imsmanifest.xml) will not be included in the manifest.

**Using the Dreamweaver Manifest Maker for ADL SCORM Extension:**

Once the Extension Manager and *Manifest Maker for ADL SCORM* extension has been installed, proceed with these steps:

1. Launch Dreamweaver and select the site for which you wish to create the manifest.
2. Select Manifest Maker from the Commands drop-down list.
3. Fill in the Manifest Maker ID, Version, Title, Description, and Keywords, as required. Manifest Maker will automatically fill in some of the fields, taking information from the site name definitions supplied by Dreamweaver. The Description field is used to provide a brief overview summary of the course. The Keywords field is used to collect relevant and specific words and phrases that may be used to help in the indexing, searchability, and retrieval of the LO. Click OK when finished.
Manifest Maker may generate a warning dialogue. Examine the warning notice (warning-mm.txt) in Dreamweaver to determine if it relates to the URL for downloading the Macromedia plug-in, such as http://download.macromedia.com/...". If so, disregard the warning. If, however, your content relies on media or services that are based on another server, you will need to open the manifest in an XML editor and add these items as resources for the corresponding SCOs. Failing to include these resources may cause import or playback problems when working with some LMS/LCMS products. It is safe to delete this text file (warning-mm.txt).

4 Open up and examine the structure of the resulting XML manifest file. Manifest Maker has organized the contents into a text-based, portable, human- and machine readable format. Notice that after the manifest file is created, several XML Schema Definition (XSD) files have been copied to the site root. These XSD files are used to specify message formats, validate data, and act as overall control documents.
Testing your Macromedia Flash MX SCO

Now that the Learning Object is built, it is important to test the SCO to ensure that it is SCORM-conformant. ADL offers several tools in one package, the SCORM Conformance Test Suite, to help developers to test their SCOs. The Test Suite is essentially a stripped-down version of a SCORM-conformant LMS/LCMS management system.

There are four tools in the Conformance Test Suite to test various aspects of SCORM:
- LMS Run-Time Environment Conformance Test
- SCO Run-Time Environment Conformance Test
- Meta-data Conformance Test
- Content Package Conformance Test

Our focus here will be with the SCO Run-Time Environment Conformance Test. The two other tests that may be of use to the Learning Object developer are the Meta-data and the Content Package Conformance Tests.

How to use the ADL SCORM Conformance Test Suite:

Prerequisites

Ensure that the following items are correctly installed on your machine before proceeding with the Conformance Test Suite:
1 Download and install Java 1.3.1:

- The Test Suite requires the Sun Java 2 Standard Edition Runtime Environment Version 1.3.1_02 or the Java Development Kit Version 1.3.1_02. Both are available for download free of charge from http://java.sun.com/j2se/1.3/jre/download-windows.html. If you intend to use this test suite software, then you must perform the download and installation prior to attempting the Test Suite installation. Please note that these are not the latest versions available but are the only ones supported for this release of the Test Suite.

- Double-click the Installer to begin the installation process.

2 Download and install the ADL Conformance Test Suite:

- The test suite is available from the ADL website at http://www.adlnet.org. Select the link for SCORM Downloads and look for “SCORM Version 1.2 Conformance Test Suite Version 1.2.2 (Self Test)”. Click on the link for the Test Suite .zip file. You may be required to register in order to complete the download process. Download the .zip file.

- Unzip the files to the location of your choice. Review the ReadMe.htm file for instructions and other information. Double-click the Installer to begin the installation process.
Testing the SCO

Once the Test Suite is installed, proceed with the following steps to test your Macromedia Flash MX SCO for the SCORM Run-Time Environment conformance:

1. Launch the Test Suite and select the link for Sharable Content Object (SCO) Run-Time Environment Conformance Test.


3. In Section 1, modify the timeout value, if desired. Select the appropriate option for single or multiple SCO testing. Click the Browse button to locate the SCO and ensure that the SCO is located on a local disk. The SCO will not work from a network drive.

4. Fill in any test student meta-data that is required for the SCO currently being tested.

5. Click on the Launch SCO(s) button to open the SCO and begin the testing procedure.
Navigate through the SCO to test the functionality of the mandatory `LMSInitialize/LMSFinish` commands and the optional `LMSGetValue/LMSSetValue` commands. Observe the interactions as they are recorded in live-time by the Test Suite, and note how the SCO and the management system begin communications, transfer data back and forth, and end the session.
Testing the Meta-Data and Content Package

The Test Suite also has the capacity to test the meta-data files and content packaging structure of your course content. To do this, close and then reopen the SCORM Conformance Test Suite. From the home page, click on the link for either the Meta-data Conformance Test or the Content Package Conformance Test. Follow the instructions to complete these tests.

Conclusion

We hope that this document helps make sense of standards issues facing eLearning developers today. It is important to be aware of the continued evolution of both accessibility and reference model standards and to make room for future developments in the eLearning development cycle.

The goal of this paper is to help you get some hands-on experience with creating accessible, SCORM-compliant eLearning content. Accessibility helps ensure access to information for people with disabilities. As developers increasingly incorporate accessibility into the design process, the more likely content is to be accessible. This will provide new opportunities for people with disabilities in job training and university environments. Such opportunities constitute a victory for the entire community.

The current version of SCORM is a well-thought-out, simple set of guidelines that suits the basic needs of managing eLearning content. Considering the relative sophistication of the Macromedia MX technologies, it is easy to see how the many features of MX exceed the SCORM standard. SCORM is not fancy, but it works. When drawing up your eLearning design and development plans, allow for the continued evolution of SCORM, and do not let the current specifications limit your development options. Over time, the standard will evolve and catch up with the increasing innovation of eLearning content and technology. ADL anticipates a new release of SCORM, Version 1.3, for late 2002.

At Macromedia, we see the goal of Learning Object development is to build shareable, rich, interactive learning content of the highest quality and ensure that it is available to the widest possible audience.

Download the supporting assets for this whitepaper with full source code from the Macromedia website: www.macromedia.com/go/objects

Macromedia Learning Objects Development Center

Macromedia is committed to help businesses, educational institutions, and government agencies create personalized, relevant learning experiences. The Macromedia Learning Objects Development Center hosts a series of whitepapers, development assets, and implementation models that will help organizations leverage their technology investments to improve organizational and employee productivity and performance. Visit www.macromedia.com/go/objects to access the full set of resources available.
Jay Heins is an eLearning consultant with a background in fine arts, video and electronic media. Jay provides direction in the appropriate use of emerging web technologies for Numen Communications. He manages all aspects of the studio’s development process, including client relations, site architecture, programming, production, and visual design. His creative leadership provides Numen’s work with a clear sense of structure and visual aesthetics. Jay has a B.F.A. in Fine Arts from the University of Ottawa in Ontario, Canada.

You can contact Jay at jay@numen.ca

Bob Regan is the Senior Product Manager for Accessibility at Macromedia. In that role, he works with designers, developers, and engineers from around the world to communicate existing strategies for accessibility as well as develop new strategies. He works with Macromedia’s engineers and designers to develop new techniques and improve the accessibility of Macromedia tools. Bob has a Masters degree from Columbia University in Education. He is currently a doctoral student in Education at the University of Wisconsin – Madison. His dissertation research looks at accessibility policy implementation strategies.

You can contact Bob at bregan@macromedia.com
Resources

**Accessibility**

Section 508 Available at: [http://www.section508.gov/index.cfm](http://www.section508.gov/index.cfm)

Thatcher, Jim; Bohman, Paul; Burks, Michael; Henry, Shawn Lawton; Regan, Bob; Swierenga, Sarah; Urban, Mark D.; Waddell, Cynthia D. (2002) *Constructing Accessible Websites*, glasshaus, Birmingham, U.K.

US Census Bureau, Americans with Disabilities: 1997 - Table 1. Available at: [http://www.census.gov/hhes/www/disable/sipp/disab97/ds97t1.html](http://www.census.gov/hhes/www/disable/sipp/disab97/ds97t1.html)

W3C Web Accessibility Initiative (WAI) Available at: [http://www.w3.org/WAI/](http://www.w3.org/WAI/)

**SCORM**


