SlideShowPro Customization and How-To Guide

(for SlideShowPro v.1.5.3)

slideshowpro

DYNAMIC, CUSTOMIZABLE PHOTO GALLERIES
FOR FLASH MX 2004, FLASH 8 AND FLASH CS3.

Contents

Introduction

This *SlideShowPro Customization Guide* was written for intermediate to advanced Flash developers (or beginners who enjoy getting their hands dirty). It compliments the *SlideShowPro User Guide* with additional information to trick out SlideShowPro with custom elements and external functionality.

Before diving in, read the "Customization basics" section immediately following this introduction. It lays the groundwork for what you'll need to know in order to use the rest of the guide.

If you run into trouble, check out the Troubleshooting section at the end, or visit the forums at `http://slideshowpro.net/forums/` to see if a fellow user can assist.

Have fun!

Customization basics

You will be required to understand a few basic principles of Flash in order to use this guide. For those with intermediate to advanced knowledge of ActionScript, you can probably skip this section. Everyone else should continue reading before moving on.

INSTANCE NAMES

When using ActionScript to manipulate SlideShowPro you must first assign SlideShowPro an instance name. To do this, select the copy of SlideShowPro on your stage, open the Properties panel (Window > Properties), type a short but memorable title into the text field shown in Figure 1. Appending `_ssp` onto the end of your instance name is recommended for it will activate code hints in the Actions panel, and make your code easier to read.

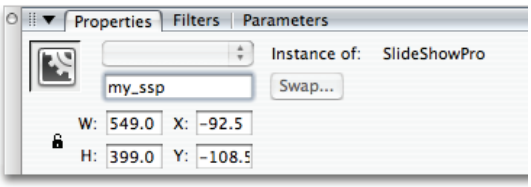


Figure 1: Instance name in Properties panel

SLIDESHOWPRO ACTIONSCRIPT DOCUMENTATION

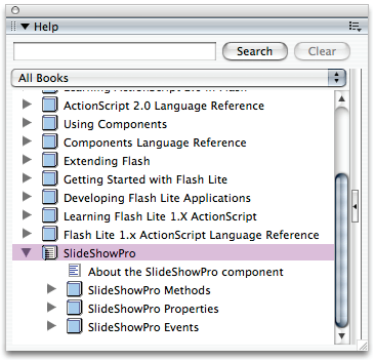


Figure 2: Help panel

To view all of SlideShowPro's properties, events, and public methods, select Help > Flash Help from the top menu in Flash. Then select "All Books" from the pull-down in the left frame. Underneath you will see a booklet named "SlideShowPro" (Figure 2). Inside are three folders: "SlideShowPro Properties," "SlideShowPro Events," and "SlideShowPro Methods." Each help document contains definitions, examples, and acceptable parameters you can use.

ACTIONS PANEL

The Action panel (Window > Actions) is where you enter ActionScript to control your movie, and for assigning properties to SlideShowPro. For example, if I wanted to change the background color of SlideShowPro to white, I'd first create a new timeline layer in my FLA and select the first keyframe of that layer. Then in the Actions panel I'd type my instance name assigned earlier, a dot (".") to access a property, the property name I want to change, and finally a value (Figure 3). Like so:

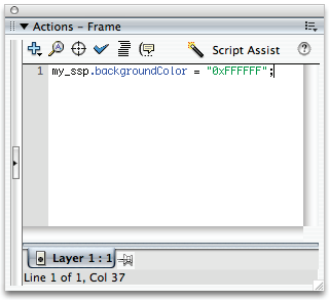


Figure 3: Actions panel

```
my_ssp.backgroundColor = "0xFFFFFF";
```

How to: SlideShowPro Button Packs

SlideShowPro Button Packs provide users with an extra set of navigation icons to replace SlideShowPro's default set. Each pack comes with two icon sizes (24 pixels square and 16 pixels square) for each navigation icon, and are designed to be used on any colored background.

PURCHASE AND DOWNLOAD

If you haven't already, you first need to purchase a button pack from `http://www.slideshowpro.net`. After download, double click on the ZIP file you received to extract a folder containing your icons.

ADD ICONS TO YOUR FLA

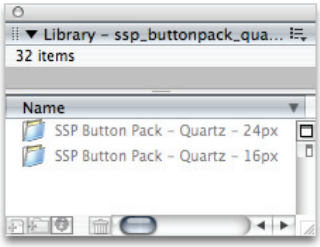


Figure 4: External Library

Launch Flash, and open the FLA that already contains SlideShowPro (or one you plan on adding it to). From the "File" menu, select "Import > Open External Library." Navigate to the FLA that's inside the button pack folder you just unzipped, select it, and click the "Open" button. A new Library panel will appear in Flash containing two icon folders (Figure 4). One contains the 16 pixel square version of the icon, the other 24 pixel square. Drag either folder from the Library panel to the Library panel of your open FLA. The icons will transfer to your FLA's Library.

GIVE SLIDESHOWPRO AN INSTANCE NAME

If you haven't done so already, make sure that SlideShowPro is on the stage of your FLA and has an "instance name" so it can be referenced with ActionScript. See the "Customization basics" section of this guide for instructions.

ACTIONSCRIPT

Now it's time to add a little ActionScript! In the folder you unzipped is a file with an ".as" file xtension. For those with a Professional version of Flash, simply double-click on this file to open. If you receive an error (that your OS doesn't know what to do with it), open the file in any text/html editor, like Notepad, Dreamweaver, TextMate, etc. Once open, copy to your clipboard the block of code for the size of icon you plan to use.

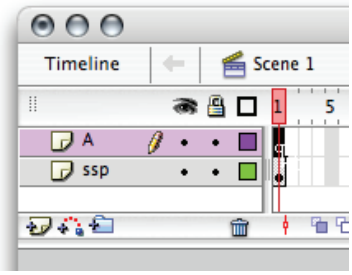


Figure 5: New timeline layer with first keyframe selected

Return to your FLA in Flash. Create a new timeline layer

by clicking on the far left icon at the bottom of the timeline window (it'll display "Insert Layer" on mouseover). A new timeline layer will appear (Figure 5). Click on the first empty keyframe in this new layer, then open the Actions panel. Click inside the right frame of the editor window, and paste the ActionScript you just copied.

PUBLISH

Publish a SWF by selecting File > Publish Preview > Flash from the top application menu. If all goes well, your icons should now be appearing in SlideShowPro!

TROUBLESHOOTING

Don't see the buttons? Here are a couple of things to check.

Instance name: Make sure you gave SlideShowPro an instance name of "my_ssp". If you are already using a different instance name, replace "my_ssp" in the supplied ActionScript with the name you are using.

Timeline: The ActionScript you pasted *must* be in a keyframe directly above or below SlideShowPro. Not before, or after. The ActionScript won't be able to find your instance of SlideShowPro otherwise.

VIDEO

A screencast demonstrating the process outlined in this chapter can be viewed at:
`http://www.slideshowpro.net/screencasts.php`

How to: Custom navigation buttons

SlideShowPro allows you to replace its default navigational buttons with your own. This chapter will show you how.

CREATE A MOVIE CLIP

The first step is to create a new MovieClip in your FLA Library. This clip will contain your first navigation button. To do this, click the "Add symbol" button in the bottom-left corner of the Library panel (Window > Library). A new dialog titled "Create new symbol" will appear. For the purposes of this walkthrough, we'll give it a name of "myicon_gallery." Enter that into the "Name:" field. Next, click on the "Advanced" button in the lower-right of the same dialog. In the new area that appears below, select "Export for ActionScript". Then in the "Identifier:" field enter the same name again ("myicon_gallery"). Your dialog should then look similar to Figure 6. When complete click "OK".

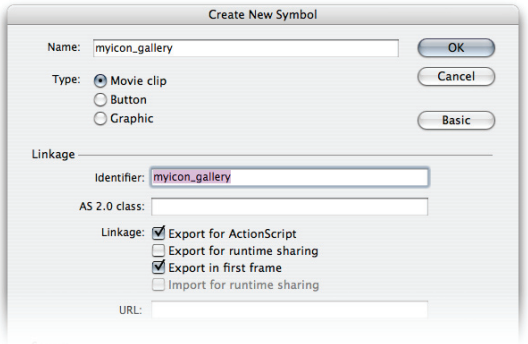


Figure 6: Create new symbol dialog

CREATE/PLACE GRAPHIC IN MOVIE CLIP

After clicking "OK" Flash will create a new MovieClip in your Library and automatically open it for editing. Import a graphic into here, or create your own vector graphic using Flash's drawing tools. The recommended dimensions for navigation icons are 16 pixels high (for the "Numbers" Navigation Type) or 24 pixels high (for the "Thumbnails" Navigation Type). Your artwork should be registered to the top left of the stage. In other words, with "X" and "Y" set to 0.

> *Tip: If you are using text fields for your custom buttons, you should include a shape underneath the text for the button's hit area. Without one, empty spaces in the text won't be clickable. You can make this shape either the same color as your navigation bar, or give the shape an instance name and make the alpha 0 with ActionScript. E.g., "bg_mc._alpha=0"*

CREATE OTHER NAVIGATION BUTTONS

Repeat the previous two steps to create your other buttons. The available buttons are "gallery," "previous image group," "previous image," "next image," "next image group," "pause," and "play."

GIVE SLIDESHOWPRO AN INSTANCE NAME

With your MovieClips complete, give SlideShowPro an instance name (see the "Customization basics" section in this guide if you don't know how). For the purposes of this walkthrough, we'll name the component `my_ssp`.

ACTIONSCRIPT

Now it's time to add the ActionScript that assigns your MovieClips to SlideShowPro. In a new timeline layer, add the following to the first keyframe (more info on this is also in the previous chapter):

```
my_ssp.navIcons = [
  "myicon_gallery",
  "myicon_prevImageGroup",
  "myicon_prevImage",
  "myicon_nextImage",
  "myicon_nextImageGroup",
  "myicon_pause",
  "myicon_play",
  "myicon_audiopause",
  "myicon_audioplay"
];
```

> *Note: Like "myicon_gallery," all of the Linkage Identifiers in the code above are for example only should be replaced with the actual Identifiers you've given your MovieClips.*

This bit of code assigns your MovieClips to an Array named `navIcons` that SlideShowPro will use to populate your navigation buttons. *Your MovieClips must be in this array order!*

PUBLISH

Publish your movie, and your custom icons should appear! If you don't see them, refer to the "Troubleshooting" section at the end of the previous chapter.

How to: External navigation

SlideShowPro supports the use of buttons outside of the component to control playback. Here's how you do it.

HIDE NAVIGATION

First, you'll probably want to remove the embedded navigation in SlideShowPro. Select your instance of SlideShowPro on the stage, open the Component Inspector, and set "Show Navigation" to False.

ASSIGN INSTANCE NAME

Your instance of SlideShowPro on the stage will need an instance name in order for outside elements to communicate with it. Select SlideShowPro, open the Properties panel, and type `my_ssp` into the Instance Name box.

CREATE BUTTONS

Create a new movie clip for each control you'd like to replace. Your options include "Previous Image", "Next Image", "Previous Image Group", "Next Image Group", "Toggle Display Mode" (Play/Pause) and "Toggle Gallery" ("Toggle Navigation" is also available, but for now we won't use it). When finished, place your movie clips anywhere on your stage, and (like we did earlier) give each an instance name through the Properties panel.

For the sake of this walkthrough, I've created four Movie Clips, and assigned them instance names of `next_btn`, `prev_btn`, `display_btn`, and `gallery_btn`.

(Note: In case you don't know, adding `_btn` to each of the instance names isn't just good organizational form, but it enables button-related code hints in the Actions panel).

ATTACH ACTIONSCRIPT

Create a new layer in your timeline, and name it "A." Select the first empty frame, open the Actions panel (Window > Actions), and enter the following:

```
next_btn.onRelease = function() {
  my_ssp.nextImage();
}
prev_btn.onRelease = function() {
  my_ssp.previousImage();
}
display_btn.onRelease = function() {
  my_ssp.toggleDisplayMode();
}
```

```
gallery_btn.onRelease = function() {
  my_ssp.toggleGallery();
}
```

You're done! Publish a new movie and your own buttons will control the main navigation of the component.

INCLUDE NAVIGATION

If you would like to keep the embedded navigation, and toggle its appearance with your own button, you may do so with the "Toggle Navigation" method. Simply create a new movie clip like the others, and attach the following:

```
nav_bttn.onRelease = function() {
  my_ssp.toggleNav();
}
```

DEMO

To see a working demo of custom navigation in SlideShowPro, login to your user account, click on the "SlideShowPro" text link, and on the next page look for "Custom Navigation FLA."

How To: Change default English text

As part of SlideShowPro's user interface there are a handful of static text fields (uncontrolled by XML) that are written in English. These values may be changed to any language or alternate text. This section will show you how.

TEXTSTRING ARRAY

The default English values for these static text fields are contained in an array named `textStrings`. These are the default values:

0 - "Previous Screen"
1 - "Next Screen"
2 - "Screen"
3 - "Image"
4 - "of"
5 - "No image caption"

The value of `[0]` is used for the left button in the gallery screen. The value of `[1]` is used for the right button in the gallery screen. The value of `[2]` is used as the screen-count field in the gallery screen. The value of `[3]` is used at the beginning of the caption header. The value of `[4]` is used in both the caption header and the screen-count field in the gallery screen. The value of `[5]` is used if an image does not have a caption and the caption area is displayed.

GIVE SLIDESHOWPRO AN INSTANCE NAME

If you haven't done so already, make sure that SlideShowPro is on the stage of your FLA and has an "instance name" so it can be referenced with ActionScript. See the "Customization basics" section of this guide for instructions.

ACTIONSCRIPT

Now it's time to add the ActionScript to change the values in the `textStrings` array. In a new timeline layer, modify the array by assigning new values for each array index:

```
my_ssp.textStrings = [
  "Pantalla Anterior",
  "Pantalla Siguiente",
  "Pantalla",
  "Imagen",
  "de",
  "Ningun subtítulo de la imagen"
];
```

This bit of code assigns SlideShowPro new values for the `textStrings` array. Do make sure that you include a value for all 6 indicies. You cannot assign a partial array.

PUBLISH

Publish your movie, and your replacement text should appear! Felicitaciones!

How to: Prevent XML caching

There's no fool-proof way to prevent your data from being cached in all web browsers. It's a simple fact of life when deploying content online. That said, there is something you can do to help *force* a user's web browser to not use its cache, and instead load unique data from you every time. Here's how.

ASSIGN INSTANCE NAME

Your instance of SlideShowPro on the stage will need an instance name in order for outside elements to communicate with it. Select SlideShowPro, open the Properties panel, and type `my_ssp` into the Instance Name box.

ATTACH ACTIONSCRIPT

Create a new layer in your timeline, and name it "A." Select the first empty frame, open the Actions panel (Window > Actions), and enter the following:

```
var xmlFile="images.xml";
var cacheBust:String = (_root._url.indexOf("file:") >= 0) ? "" :
        "?rn="+new Date().getTime()+(Math.random()*100);
var src:String = xmlFile + cacheBust;

my_ssp.xmlFilePath= src;
```

Next, if your XML file isn't named `images.xml`, or you're using an absolute path, edit the `xmlFile` parameter with the actual path to your XML file.

PUBLISH

Publish a new movie, and view the HTML it creates in a web browser. Your slideshow should work the same as it did before, but is now appending random numbers as a parameter onto the end of the XML file path. This fools the browser into thinking that the requested file is different than the one in its cache, and loads it as if it were a new asset.

> *Note: If you are using SlideShowPro Director, there's no need to add this to your FLA. XML caching prevention is built-in so that a unique file is set to your slideshow everytime.*

How to: Dynamically assign an XML file

SlideShowPro allows you to dynamically assign its XML File Path through the HTML document that's embedding it. Here's how you do it.

Open in a text editor the HTML document used to load SlideShowPro. Inside will be a bundle of `object`, `param`, and `embed` elements. It is here we will assign our XML file using FlashVars -- a handy mechanism for importing variables into an embedded movie without re-opening a FLA and publishing a new movie.

Add the following `param` element before, after, or inbetween any of your existing `param` elements:

```
<param name="FlashVars" value="xmlfile=myXML.xml" />
```

This creates a unique variable named `xmlfile`, which is then assigned the value of our XML file.

Next, do this a second time by adding a new FlashVars attribute to your existing `embed` element:

```
<embed FlashVars="xmlfile=myXML.xml" ... (other attributes)>
```

Your HTML edits are now complete. Save the HTML file and close it.

Next, open the FLA containing SlideShowPro. Open the Properties panel (Windows > Properties) and give the component instance a unique variable name. For this tutorial, I'll name it "my_ssp."

Now we need to write some ActionScript. Create a new layer in your movie timeline, and click its first frame so it's highlighted. Open the Actions panel and enter this:

```
my_ssp.xmlFilePath=xmlfile;
```

That's it! To test your work, open the HTML file in a web browser.

EXTRA TIP: SWFOBJECT

If you're using SWFObject (`http://blog.deconcept.com/swfobject/`) to embed your SWF (which is recommended to avoid Internet Explorer "Activate" clicking), you'd pass the variable like so:

```
so.addVariable("xmlfile", "myXML.xml");
```

How to: Load XML data from a different domain

By default, the Flash Player has a security wall that prevents the loading of data from a domain that is different from the one a SWF resides on. In other words, if your SWF were hosted on `www.mydomain.com` and either your static XML file or installation of Director were on `www.myotherdomain.com`, then the SWF wouldn't have permission to load images.

So how to do you grant permission? It's done with what's called a "cross domain policy file." It's a simple XML file that the Flash Player automatically looks for at the outside domain (the one Director or your XML is hosted at), and if the file grants permission to the domain hosting your SWF, your images are allowed to load.

CREATE A CROSS DOMAIN POLICY FILE

Here's what you do. In your favorite text editor, create a new file and name it `crossdomain.xml` (it *must* be named this). Then add the following:

```
<cross-domain-policy>
<allow-access-from domain="" />
</cross-domain-policy>
```

This is the basic template. In the `domain` attribute, add the domain where your SWF resides. For example:

```
<cross-domain-policy>
<allow-access-from domain="www.mydomain.com" />
<allow-access-from domain="mydomain.com" />
</cross-domain-policy>
```

Notice that the domain is listed twice -- once with `www` and once without. This is in case the SWF is loaded from a URL that doesn't include `www`.

UPLOAD

Save the file. Upload crossdomain.xml to the root of the domain where your XML file or Director installation resides. For example, `http://www.myotherdomain.com/crossdomain.xml`.

Reload your HTML/SWF in your web browser. Images should now be loading using data from the other domain.

How to: Preload SlideShowPro

Most Flash developers add preloaders to the beginning of their movies so that all content is loaded before playback begins. By default however, Flash exports component data in the first keyframe (before your preloader), which defeats the purpose. Here's what you do so that component data is loaded *after* your preloader.

UNDO DEFAULT EXPORT SETTING

Right click on the copy of SlideShowPro in your FLA's Library panel. Select "Linkage". Uncheck the "Export in First Frame" option. Click OK.

CHANGE CLASS FRAME EXPORT SETTING

Open File > Publish Settings. Click on the Flash tab. Then click on the "Settings" button next to "ActionScript Version". In the settings dialog, change the "Export frame for classes" from 1 to an empty keyframe on your timeline that's *after* your preloader, but *before* SlideShowPro on your timeline. For example, if your preloader was on frame 2, and your instance of SlideShowPro was on frame 10, you could pick any number between 3 and 9.

PUBLISH

Now when you publish your FLA the class files are exported after the preloader, thus creating a more accurate preloader at the beginning of your movie.

How to: Control SlideShowPro using HTML/Javascript

This walkthrough will show you how to setup SlideShowPro, and the HTML document embedding your movie, so that hyperlinks in your HTML can be used to interact with SlideShowPro. By the end of the walkthrough we'll create one HTML link that calls SlideShowPro's `toggleDisplayMode()` method to control playback.

> *Note: This will only work if you use Flash 8 (or higher), and publish your SWF to Flash Player 8 (or higher). It will not work in Flash MX 2004 or a SWF published to Flash Player 7.*

CREATE FLA

Create a new FLA, and drag a new instance of SlideShowPro onto the Stage. Position it accordingly, and give it an instance name (see "Customization Basics" earlier in this guide for instructions). For the purposes of this walkthrough we'll name it `my_ssp`.

ADD ACTIONSCRIPT TO FLA

Create a new timeline layer above the one containing SlideShowPro and click in the first empty keyframe. Add the following:

```
import flash.external.*;

ExternalInterface.addCallback("sspToggleDisplayMode", null,
         sspToggleDisplayMode);

function sspToggleDisplayMode() {
  my_ssp.toggleDisplayMode();
}
```

This little bit of ActionScript will setup your SWF so that Javascript can communicate with it.

ADD JAVASCRIPT TO HTML

In the HTML document embedding your SWF, add the following anywhere before the closing `</head>` element:

```
<script type="text/javascript">
  function sspToggleDisplayMode() {
    thisMovie("ssp").sspToggleDisplayMode();
  }
  function thisMovie(movieName) {
    if (navigator.appName.indexOf("Microsoft") != -1) {
         return window[movieName]
  }
```

```
      else {
          return document[movieName]
      }
  }
</script>
```

This is what we'll call from our hyperlinks to communicate with the SWF that has been assigned an ID of `ssp`. Flash automatically creates the ID for you as part of your player embed code using the title of your SWF (minus the ".swf"). If you're using the code Flash publishes to embed your movie, check to see what this ID is. Or if you're using a third party embedding script (like SWFObject), name the ID yourself. Doesn't matter what ID is assigned, as long as Javascript knows what it is.

ADD HYPERLINK

Now for the final piece of the puzzle -- a hyperlink. Anywhere in your HTML document, add the following:

```
<a href="#" title="Toggle Display Mode"
  onclick="sspToggleDisplayMode();">
    Toggle Display Mode
</a>
```

TEST IT OUT

Open the HTML document in your browser, wait for an album to load in SlideShowPro, then click on the hyperlink. You should now be controlling playback in SlideShowPro from your text link!

TROUBLESHOOTING

Link not working?

• Check the ID of your SWF against the one you entered in the Javascript. They must match exactly.

• Check to make sure Javascript is turned on in your browser.

• Hyperlinking like this only works in Flash Player 8 or higher. Check to make sure you're publishing to at least that.

DEMO

A working demo of this walkthrough is also available in the user section of `slideshowpro.net`.

Event listeners

SlideShowPro supports the use of ActionScript Event Listeners. This enables events that occur within the component to be responded to with outside objects. For example, you could populate a ComboBox with gallery data and use it to navigate between albums. Or, you could set up a text field that automatically displays image captions. There are countless ways you can use events (and the data they return) to customize SlideShowPro.

Note: This section expects you to be comfortable coding ActionScript and to have at least an intermediate understanding of arrays, loops and objects.

DATA EVENTS

`onLoadXML`
Triggered when SlideShowPro attempts to load an XML file. Event object contains a Boolean value (`false`, `true`) to indicate whether an XML file was found.

`albumData`
Triggered when an album is loaded. Event object contains the following properties:

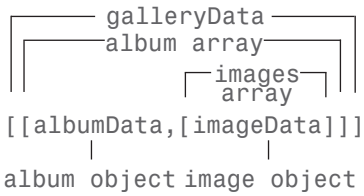
- `id` (string; the album ID -- not supported by Flickr RSS)
- `description` (string; the album's description)
- `lgpath` (string; the album's lgPath -- not supported by Flickr RSS)
- `tnpath` (string; the album's tnPath -- not supported by Flickr RSS)
- `title` (string; the album's title)
- `tn` (string; the album's thumbnail)
- `totalImages` (the number of images in the album)

`imageData`
Triggered when an image is loaded. Event object contains the following properties:

- `caption` (string; the image's caption)
- `link` (string; the image's hyperlink)
- `number` (number; the image's numerical ID)
- `pause` (number; the image's pause override value -- not supported by Flickr RSS)
- `src` (string; the image's file name)
- `target` (string; the link's browser window target -- not supported by Flickr RSS)
- `tn` (string; the image's thumbnail URL)

`galleryData`
Triggered when SlideShowPro successfully parses an XML document. Event object contains all `albumData` and `imageData` objects segmented into a multidimensional array.

The structure of the array is as follows:



PLAYBACK EVENTS

`albumEnd`
Triggered when SlideShowPro has displayed the final image in an album and engages "Auto Finish Mode" to decide what to do next. This event triggers with the "Display Mode" parameter set to either "Auto" or "Manual." Returns no event object.

`displayModeChange`
Triggered anytime the "Display Mode" parameter changes. Event object contains the strings "Auto" or "Manual".

`galleryState`
Triggered anytime the position of the gallery changes. Event object contains the strings "Open", "Closed" and "Hidden".

`imageClick`
Triggered when a mouse clicks on a loaded image. Returns nothing.

`imageRoll`
Triggered when a mouse pointer passes in and out of a loaded image's area. Event object contains the strings "over" when the pointer has rolled-over an image, and "out" when rolled-out.

`navButtonDisabled`
Triggered anytime SlideShowPro disables the forward or back navigation buttons. Event object contains the strings "previous" (the minus button) and/or "next" (the plus button) to indicate which button was disabled.

`navButtonEnabled`
Triggered anytime SlideShowPro enables the forward or back navigation buttons. Event object contains the strings "previous" (the 'minus' button) and/or "next" (the 'plus' button) to indicate which button was enabled.

`preloadStart`
Triggered anytime SlideShowPro engages its preloading animation. Event object contains the percentage of loaded bytes.

`preloadEnd`
Triggered when the preloading of on-demand content ends. Returns nothing.

HOW TO CREATE AN EVENT LISTENER

Event Listeners are created outside of the component in a timeline frame using the ActionScript editor. They are simple objects whose sole purpose is to sit quietly and listen to what's going on inside a component, and respond if called by name.

First, you need to create a listener object:

```
var myListener = new Object();
```

Then, using the supported events listed above, create a object property with a name exactly the same as the event you want to listen to, like so:

```
myListener.onLoadXML = function(eventObject):Void {
}
```

Finally, register the object as a listener for your instance of SlideShowPro:

```
my_ssp.addEventListener("onLoadXML", myListener);
```

Now when the `onLoadXML` event is triggered inside SlideShowPro, the listener property with the same name will be automatically called. This is the basic model for all event listeners.

If you want to create two listeners, you can re-use the same listener object by adding another property, like so:

```
myListener.albumEnd = function(eventObject):Void {
}
my_ssp.addEventListener("albumEnd", myListener);
```

EVENT OBJECTS

Whenever an event is triggered, it automatically passes an event object to your listener object. Each event object has properties that contain information about the event. Using these properties, you may write ActionScript to use the data any way you like.

For example, in the case of `onLoadXML`, the listener object receives a boolean value to indicate whether an XML file was found. This is how you retrieve that value:

```
myListener.onLoadXML = function(eventObject):Void {
  trace(eventObject.data);
}
```

Either "true" or "false" will be written to your output window. You could also use this data to set up a conditional:

```
myListener.onLoadXML = function(eventObject):Void {
```

```
  if (eventObject.data==true) {
    // show success alert
  } else {
    // show error alert
  }
}
```

In addition to Boolean values, event objects can also contain complex arrays or objects (yes, an object inside an object!) as we'll see in the next two examples.

EVENT OBJECTS: GALLERYDATA

The `galleryData` event broadcasts an event object `(eventObject)` containing a multidimensional array with all your gallery data. The array is filled with album arrays, each with two 'slots' -- the first `(albumData)` containing your album's data, and the second containing an array filled with the album's image data `(imageData)`. (See beginning of this section for a diagram).

For example, if you wanted to return the album title of the third album, you'd retrieve it like so:

```
myListener.galleryData = function(eventObject):Void {
  trace(eventObject.data[2][0].title);
}
slideshowproinstance.addEventListener("galleryData", myListener);
```

Or, if you wanted to return the "src" value of the third image in the third album, you'd retrive it like so:

```
myListener.galleryData = function(eventObject):Void {
  trace(eventObject.data[2][1][2].src);
}
```

EVENT OBJECTS: IMAGEDATA

A subset of the same data broadcasted in `galleryData`, `imageData` broadcasts data only for the current image loaded in SlideShowPro, and is triggered each time an image loads (as opposed to once at the beginning like `galleryData`).

To retrieve the data for the currently loaded image, you'd first create a listener object:

```
myListener.imageData = function(eventObject):Void {
}
my_ssp.addEventListener("imageData", myListener);
```

The event object received contains an image object with properties named exactly the same as the "img" elements in your XML file. So if you wanted to retrieve the caption for the current image, you'd do it like this:

```
myListener.imageData = function(eventObject):Void {
  trace(eventObject.data.caption);
}
```

EVENT OBJECTS: ALBUMDATA

`albumData` is also a subset of `galleryData`, broadcasts data for the current album loaded in SlideShowPro, and is trigerred each time an album loads.

To retrieve the data for the current album, follow the exact same walkthrough as `imageData` above, but subsitute the data property with the album property you need. In other words, `eventObject.data.totalImages`.

ADDITIONAL HELP

For more information about Event Listeners, see "Using event listeners" in the Flash ActionScript help documentation.

ActionScript parameter list

For those of you dynamically assigning values to SlideShowPro with ActionScript, here's a list of all modifyable parameters (with default values assigned) you can copy/paste into your code. For a list of acceptable values, open the "SlideShowPro" booklet inside Flash's Help panel.

```
my_ssp.albumActiveColor = "0x262626";
my_ssp.albumBackgroundAlpha = 100;
my_ssp.albumBackgroundColor = "0x1A1A1A";
my_ssp.albumDescColor = "0xCCCCCC";
my_ssp.albumDescSize = 9;
my_ssp.albumStrokeColor = "0x333333";
my_ssp.albumStrokeWeight = 1;
my_ssp.albumThumbStrokeColor = "0xFFFFFF";
my_ssp.albumThumbStrokeWeight = 1;
my_ssp.albumTitleColor = "0xFFFFFF";
my_ssp.albumTitleSize = 9;
my_ssp.albumType = "Text and Thumb"
my_ssp.audioBackgroundAlpha = 80;
my_ssp.audioBackgroundColor  = "0x000000";
my_ssp.audioLoop = false;
my_ssp.audioPause = true;
my_ssp.audioVolume = 50;
my_ssp.autoFinishMode = "Switch";
my_ssp.backgroundAlpha = 100;
my_ssp.backgroundColor = "0x1A1A1A";
my_ssp.cacheImages = "None";
my_ssp.captionBackgroundAlpha = 75;
my_ssp.captionBackgroundColor = "0xFFFFFF";
my_ssp.captionHeaderType = "Image Count";
my_ssp.captionStrokeColor = "0xFFFFFF";
my_ssp.captionTextColor = "0x333333";
my_ssp.captionTextSize = 9;
my_ssp.displayMode = "Auto";
my_ssp.feedbackScale = 100;
my_ssp.fxClickSound = "";
my_ssp.fxRolloverSound = "";
my_ssp.fxSlideSound = "";
my_ssp.galleryBackgroundAlpha = 100;
my_ssp.galleryBackgroundColor = "0x000000";
my_ssp.galleryColumns = 2;
my_ssp.galleryOrder = "Left to Right";
my_ssp.galleryRows = 4;
my_ssp.imageAlign = "Center";
my_ssp.imageOrder = "Sequential";
my_ssp.imageScale = "Scale";
my_ssp.imageStrokeColor = "0x333333";
my_ssp.imageStrokeWeight = 0;
my_ssp.keyboardControl = true;
my_ssp.navActiveColor = "0xFFFFFF";
my_ssp.navBackgroundColor = "0x333333";
my_ssp.navIconInactiveAlpha  = 40;
my_ssp.navIconShadowAlpha  = 60;
my_ssp.navLinkCurrentColor = "0xFFFFFF";
```

```
my_ssp.navLinkType = "Numbers";
my_ssp.navLinksBackgroundAlpha = 25;
my_ssp.navLinksBackgroundColor = "0x000000";
my_ssp.navNumberLinkColor = "0x999999";
my_ssp.navNumberLinkSize = 9;
my_ssp.navThumbBackgroundColor = "0xFFFFFF";
my_ssp.navThumbLinkHeight = 25;
my_ssp.navThumbLinkInactiveAlpha = 100;
my_ssp.navThumbLinkStrokeWeight = 1;
my_ssp.navThumbLinkWidth = 25;
my_ssp.navThumbPreviewStrokeWeight = 2;
my_ssp.permalinks = false;
my_ssp.preloadAlpha = 50;
my_ssp.preloadColor = "0xFFFFFF";
my_ssp.preloaderStyle = "Pie";
my_ssp.showAudio = true;
my_ssp.showCaptionHeader = true;
my_ssp.showCaptions = "Overlay Mouse Over Top (If Available)";
my_ssp.showDisplayChange = true;
my_ssp.showGallery = "Always";
my_ssp.showGalleryNavigation = "Auto";
my_ssp.showNavigation = true;
my_ssp.showNavDisplayControl = true;
my_ssp.showPreloader = true;
my_ssp.showNavThumbPreview = true;
my_ssp.strokeColor = "0x262626";
my_ssp.strokeWeight = 1;
my_ssp.textAlignment = "Left";
my_ssp.transitionLength = 2;
my_ssp.transitionPause = 4;
my_ssp.transitionStyle = "Cross Fade";
my_ssp.xmlFilePath = "images.xml";
my_ssp.xmlFileType = "XML";
```