SlideShowPro for Flash Customization Guide

(Version 1.7.x)

slideshowpro

Contents

Introduction 3
Customization basics 4
How to: SlideShowPro for Flash Button Packs 5
How to: Custom navigation buttons 7
How to: Embed SWF in a separate HTML document 9
How to: External navigation 12
How To: Change default English text 13
How to: Prevent XML caching 14
How to: Dynamically assign an XML file 15
How to: Load XML data from a different domain 16
How to: Create permalinks 17
How to: Preload SlideShowPro for Flash 20
How to: Control SlideShowPro for Flash using HTML/Javascript 21
Event listeners 23
ActionScript parameter list 28

Introduction

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

Here are the typographical conventions you can expect so see:

Parameter name
: Indicates parameter names that are assigned to SlideShowPro in Flash's Component Inspector panel. They will also be used (sporadically) for typographical emphasis.

`Constant width`
: Used for samples of XML, JavaScript, ActionScript, and other code.

`Constant width italic`
: A placeholder within code samples that should be substituted with your own value.

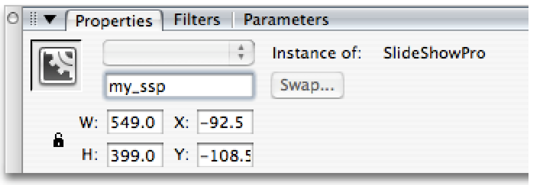
Have fun!

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

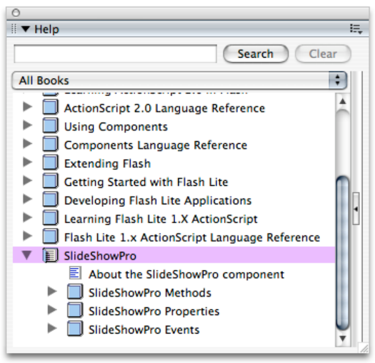
When using ActionScript you must first assign SlideShowPro for Flash an instance name. To do this, select the copy of SlideShowPro for Flash on your Stage, open the Properties panel (Window > Properties), type a short but memorable title into the text field shown right. Appending `_ssp` onto the end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



Instance of: SlideShowPro
my_ssp
W: 549.0 X: -92.5
H: 399.0 Y: -108.5

Figure: Instance name in Properties panel

SLIDESHOWPRO ACTIONSCRIPT DOCUMENTATION



Help
Search
Clear
All Books
ActionScript 2.0 Language Reference
Using Components
Components Language Reference
Extending Flash
Getting Started with Flash Lite
Developing Flash Lite Applications
Learning Flash Lite 1.X ActionScript
Flash Lite 1.x ActionScript Language Reference
SlideShowPro
About the SlideShowPro component
SlideShowPro Methods
SlideShowPro Properties
SlideShowPro Events

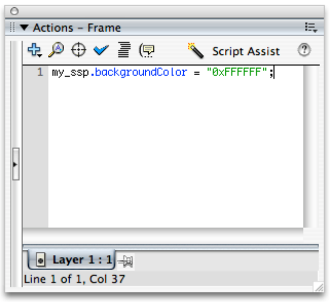
Figure: Help panel

To view all of SlideShowPro for Flash'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 for Flash." Inside are three folders: "SlideShowPro for Flash Properties," "SlideShowPro for Flash Events," and "SlideShowPro for Flash 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 Flash. For example, if I wanted to change the background color of SlideShowPro for Flash 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. Like so:

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



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

Figure: Actions panel

How to: SlideShowPro for Flash Button Packs

SlideShowPro for Flash Button Packs provide users with an extra set of navigation icons to replace SlideShowPro for Flash'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.

Note: SlideShowPro Button for Flash Packs require at least version 1.5.0 of SlideShowPro for Flash. Login to your profile at `http://slideshowpro.net` to obtain the latest version if necessary.

PURCHASE AND DOWNLOAD

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

ADD ICONS TO YOUR FLA

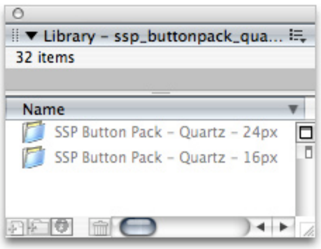

Library – ssp_buttonpack_qua...
32 items
Name
SSP Button Pack – Quartz – 24px
SSP Button Pack – Quartz – 16px

Figure: External Library

Launch Flash, and open the FLA that already contains SlideShowPro for Flash (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 (see left). 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 FOR FLASH AN INSTANCE NAME

If you haven't done so already, make sure that SlideShowPro for Flash 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 extension. 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.

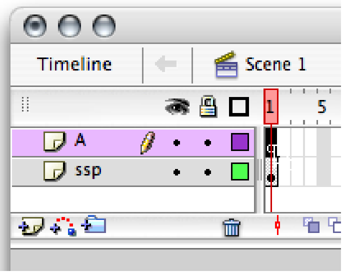


Figure: New timeline layer with first keyframe selected

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. 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 for Flash!

TROUBLESHOOTING

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

Instance name: Make sure you gave SlideShowPro for Flash 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 for Flash. Not before, or after. The ActionScript won't be able to find your instance.

VIDEO

A screencast demonstrating the process outlined in this chapter can be viewed here:

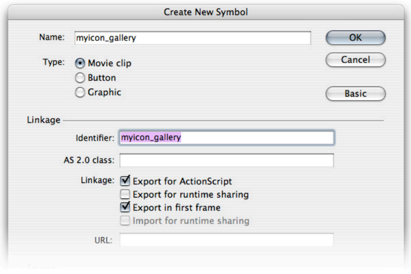
`http://slideshowpro.net/help/screencasts/02_SSP_Button_Packs.php`

How to: Custom navigation buttons

SlideShowPro for Flash 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”). When complete click “OK”.


Create New Symbol

Name: myicon_gallery

Type: [✓ Movie clip] Button Graphic

Linkage

Identifier: myicon_gallery

AS 2.0 class:

Linkage: [✓ Export for ActionScript] Export for runtime sharing [✓ Export in first frame] Import for runtime sharing

URL:

Figure: 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 Link Appearance*). 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,” “play,” “full screen” and “normal screen.”

GIVE SLIDESHOWPRO AN INSTANCE NAME

With your MovieClips complete, give SlideShowPro for Flash an instance name (see the “Customization basics” chapter in this guide if you don’t know how). For the purposes of this walkthrough, we’ll name the component `my_ssp`.

ADD ACTIONSCRIPT

Now it's time to add the ActionScript that assigns your MovieClips to SlideShowPro for Flash. 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",
       "myicon_fullScreen",
       "myicon_normScreen"
];
```

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 for Flash 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: Embed SWF in a separate HTML document

It's assumed in most of SlideShowPro for Flash's documentation you'll be using the HTML document Flash publishes when presenting your slideshow online. But what about viewing your slideshow in a *different* HTML document? Even one that's in a different directory than your SWF? This chapter will show you how.

THE HTML PROBLEM

One of the most confusing things about embedding SWFs, especially for Flash beginners, is what happens when you embed a SWF in an HTML document. Here's the deal. By default, when a SWF is embedded in HTML, all URLs called from both inside the SWF, and the content the SWF loads (including our XML file), become relative to the directory location of the HTML document.

This means that if your XML document were constructed using relative links, and not absolute links, those relative links would likely break because *they'd now be relative to the HTML document*, and SlideShowPro for Flash wouldn't be able to find what it needs.

Note: See the "How SlideShowPro for Flash works" chapter at the beginning of the *SlideShowPro for Flash User Guide* for examples of relative and absolute XML links.

If your XML document is constructed with relative links, don't panic! A little later we'll override this and tell the SWF that all links are relative to itself, not the HTML it's loaded into. But first we first need to embed our SWF in the other HTML document. Here's how to do it.

STEP ONE: EDIT PLAYER EMBED CODE (FLASH MX 2004 / FLASH 8)

If you are using Flash MX 2004 or Flash 8, the HTML document Flash published contains a block of code between `<body>` and `</body>`. Select all of this content and copy it. Open the other HTML document you want to view the slideshow in, and paste the block of code wherever you like. You can now skip to Step Two.

STEP ONE: EDIT PLAYER EMBED CODE (FLASH CS3)

If you are using Flash CS3, things are little more complicated. Flash CS3 publishes a Javascript file ("AC_RunActiveContent.js") alongside your HTML and SWF that's used to embed the SWF. This file is required in order for your SWF to appear. Take the javascript file and either move it into the same directory as the HTML document you're loading the SWF in, or place it somewhere you can link to.

Next, open the HTML document Flash published and look for this in the `<head>`:

```
<script language="javascript">AC_FL_RunContent = 0;</script>
<script src="AC_RunActiveContent.js" language="javascript"></script>
```

Select all of this and copy. Paste it into the `<head>` of the other HTML document. Modify the path to the `AC_RunActiveContent.js` if you moved the file somewhere outside of the HTML directory.

Return to the HTML document Flash published and copy all of the code between `<body>` and `</body>`. Paste this wherever you like inside of your other HTML document.

STEP TWO: EDIT SWF LINKS

Now we need to edit the code we just pasted so that it points to our SWF. This is the same for all versions of Flash. Look for this in your player embed code:

```
<param name="movie" value="slideshow.swf" />
```

A little ways later, find this:

```
<embed src="slideshowpro.swf"
```

Edit *both* of these with the new path to the SWF in the "slideshow" folder. This may be a relative path (e.g., `slideshow/slideshow.swf`) or an absolute path (e.g., `http://www.mydomain.com/slideshow.swf`).

ABSOLUTE PATHS IN XML?

If your XML was created with absolute links (http://), your slideshow should now work. If your XML was created with relative links, we need to make one small modification to get it to work.

RELATIVE PATH FIX: ADD BASE ATTRIBUTE

The `base` parameter is a special override forces the Flash Player into making all links relative to the SWF, not the HTML document.

Add the following alongside the other existing `<param>` elements in your HTML document:

```
<param name="base" value="." />
```

Then add the following anywhere inside your existing `<embed>` element:

```
base="."
```

Your `embed` element should then look similar to this:

```
<embed base="." src="slideshow.swf" quality="best" scale="noborder"
       bgcolor="#666666" width="550" height="400" name="slideshow"
       align="middle" allowScriptAccess="sameDomain"
       type="application/x-shockwave-flash"
       pluginspage="http://www.macromedia.com/go/getflashplayer" />
```

If you copied your code from Flash MX 2004 or Flash 8, save your HTML document, and load it in your web browser. Your slideshow should be working again! Flash CS3 users need to perform one extra step though.

In the code you pasted after the `<body>` element, look for `AC_FL_RunContent`. There should be a long list of attributes running down the page after it. At the end of that list, before the closing parenthesis, add:

```
'base', '.'
```

DONE!

If all went well, you should be able to load your SWF now in the other HTML document.

TROUBLESHOOTING

If however the slideshow doesn't appear, go back over these instructions to ensure that you copied/ modified everything correctly.

If your XML file was constructed using relative paths, and you're linking to your SWF using a relative path from your HTML document, consider switching all your links to absolute. This includes the link to the SWF from your HTML document, the path to the XML file that SlideShowPro for Flash loads (assigned through *XML File Path*), as well as the image links in the XML file.

If you are trying to embed your SWF in an HTML document that's under a different domain than where your XML file and images live, then the Flash Player may be denying you from loading the content because it didn't receive permission to do so. How do you grant permission? See the "How to: Load XML data from another domain" chapter in this guide.

How to: External navigation

SlideShowPro for Flash 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 for Flash on the Stage, open the Component Inspector, and set *Navigation Appearance* to "Hidden."

ASSIGN INSTANCE NAME

Click on the instance of SlideShowPro for Flash on the Stage, open the Properties panel, and type `my_ssp` into the Instance Name box.

CREATE BUTTONS

Create a new MovieClip for each control you'd like to replace. Your ActionScript options include `previousImage()`, `nextImage()`, `previousImageGroup()`, `nextImageGroup()`, `toggleDisplayMode()` and `toggleGallery()`. 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 MovieClips, 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.

How To: Change default English text

There are a handful of static text fields (uncontrolled by XML) in SlideShowPro for Flash that are written in English. This chapter will show you how to change them to any language or alternate text.

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 - "of"
4 - "No caption"
5 - "No title"
```

GIVE SLIDESHOWPRO AN INSTANCE NAME

Click on the instance of SlideShowPro for Flash on the Stage, open the Properties panel, and type `my_ssp` into the Instance Name box.

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",
        "de",
        "Ningun subtítulo de la imagen",
        "Ningun titulo"
];
```

This bit of code assigns SlideShowPro for Flash new values for the `textStrings` array. Do make sure that you include a value for all 6 indices. 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

Click on the instance of SlideShowPro for Flash on the Stage, 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:String = "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 every time.

How to: Dynamically assign an XML file

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

STEP ONE: OPEN HTML DOCUMENT

Open the HTML document Flash published in your favorite text editor. Notepad, TextWrangler, Dreamweaver, etc will work fine.

STEP TWO: EDIT PARAM AND EMBED ELEMENTS

Variables are passed into SWFs using FlashVars. It's a handy way to pass data into a SWF by editing the parent HTML document embedding it. How your SWF is embedded in the HTML depends on the version of Flash you're using. All supported versions (Flash MX 2004, Flash 8 and Flash CS3) publish a bundle of `object`, `param`, and `embed` elements. Flash CS3 however adds an additional embed method using Javascript, and requires an extra step for editing.

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

```
<param name="FlashVars" value="xmlfile=http://mydomain.com/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=http://mydomain.com/myXML.xml" ... (other attributes)>
```

STEP THREE (FLASH CS3 ONLY): EDIT JAVASCRIPT

Flash CS3 users have an additional step -- modifying the inline Javascript. Towards the top of your HTML document you should see a method named `AC_FL_RunContent`, followed by a descending list of comma-separated parameters. Anywhere in that list (the bottom is fine), add the following:

```
'FlashVars','xmlfile=http://mydomain.com/myXML.xml'
```

Make sure when adding this parameter that you include a comma at the end if necessary.

STEP FOUR: EDIT FLA

Open the FLA containing SlideShowPro for Flash. 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.

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`. When complete, save.

UPLOAD

Upload crossdomain.xml to the root of the domain where your XML file or SlideShowPro 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: Create permalinks

SlideShowPro for Flash allows you to automatically publish links to albums and images. These links (or, to borrow a phrase from the weblog world, "permalinks") produce web-friendly URLs that allow SlideShowPro for Flash to behave just like HTML, whereby a specific album and/or image will be shown as the first image in your slideshow.

HOW TO USE PERMALINKS

Before we begin with the walkthrough, a note about how you should and should not use permalinks. The URLs that permalinks create are designed for linking from a separate HTML document to the HTML document containing your slideshow. They should *not* be used as hyperlinks in the HTML document embedding the slideshow. The reason? Some browsers will ignore the request, for it thinks you're already on the requested page. If you need to incorporate hyperlinks to albums or images in SlideShowPro for Flash from your HTML document, an alternate solution can be found in the *Customization Guide*.

HOW TO CREATE PERMALINKS

This walkthrough requires an intermediate level of web development experience, so only attempt this if you're fairly comfortable hand-coding HTML. With that, let's go.

STEP ONE: MODIFY YOUR XML DOCUMENT

If you are editing your XML document by hand, open it and add an `id` attribute to each album element. This `id` will contain the unique identifier used in your permalinks. Fill it with a short identifier without spaces or ampersands. For example:

```
<album id="nature" title="Nature" description="Images of trees, lakes and flowers"
lgpath="gallery/album1/large/">
```

Repeat this for each one of the albums in your XML document.

STEP TWO: SETUP SLIDESHOWPRO FOR FLASH

Open your FLA, and select the instance of SlideShowPro for Flash on the Stage. Open the Component Inspector, and ensure that the *Permalinks* parameter is set to "On". When finished, publish a SWF and upload it to your server.

STEP THREE: SETUP YOUR HTML DOCUMENT

In order for your permalinks to be created, you need to add some scripts to your HTML document. Anywhere before `</head>` in your document, insert the following:

```
<script language="VBScript">
 <!--
 Sub swfname_FSCommand(ByVal command, ByVal args)
 select case command
 case "putHREF" location.href = args
 case "putTitle" document.title = args
 end select
 end sub
 -->
 </script>

 <script type="text/javascript">
 <!--
 function flashPutHref(href) { location.href = href; }
 function flashPutTitle(title) { document.title = title; }
 -->
 </script>
```

This chunk of code will provide a gateway for your Flash movie to communicate with Internet Explorer (VBScript) and other browsers (Firefox, Mozilla, Opera, Safari, etc.). You may copy this as-is, but you must replace `swfname` in the VBScript to the name of your Flash movie. This is typically the file name of your SWF, minus the ".swf" extension. This topic will be revisited, so if you're confused return to this part in a minute.

For embedding Flash content, Geoff Stearns' SWFObject is the recommended solution. It detects Flash player versions, provides alternate content, is standards-compliant, and makes the embedding of movies a whole lot more straightforward. It's not absolutely required, but it makes this process much easier. You may download it at: `http://blog.deconcept.com/swfobject/`

After you've downloaded the `swfobject.js` script, place it anywhere you like on your server. Next, include the script in the head of your document, like so:

```
<script type="text/javascript" src="http://www.yourdomain.com/js/swfobject.js">
       </script>
```

Next, create a named element anywhere after the `body` element in your HTML that will serve as a container for your Flash content, and fill it (optionally) with content for users without the Flash plugin. The non-Flash content will be overwritten with your SWF.

```
<div id="flashcontent">
       <!-- insert non-Flash content here -->
</div>
```

Then embed the SWF in the container by adding the following:

```
 <script type="text/javascript">
 // <![CDATA[
 var so = new SWFObject("slideshowpro.swf", "slideshowpro", "555", "379", 7,
"#666666");
 so.addVariable("initialURL", escape(document.location));
 so.write("flashcontent");
 // ]]>
 </script>
```

Replace the attributes above with the path to your SWF, the name of your movie, its width, height, the Flash Player version, and the background color of your movie. Modify the `write` method to include the name of the container element created one step back.

Next up, the VBScript. This is where the name of your Flash movie is important. Copy the name of your movie (the second attribute in `new SWFObject()`, which is `slideshowpro` in this example) and replace `swfname` at the beginning of the VBScript noted earlier. For example, `swfname_FSCommand` would now be `slideshowpro_FSCommand`.

STEP FOUR: DONE!

Your document is now completely setup for permalinks. Whenever a user clicks on an image link, the browser's document path will automatically update with the applicable album id and image number so you (or anyone) can reference it externally.

EXTRA NOTES

Test online: It is important that you test permalinks from a web server (`http://`) and not on your local machine. SlideShowPro for Flash will not attempt to update your browser's document path unless it originates from http. This is to avoid annoying errors and warnings when working locally.

Extra modifications: If you view the source of the demo above, you'll notice that the write() method contains an attribute of `flashcontent`. This is part of an optional feature of SWFObject, whereby you can specify a `div` in your document to write your SWF into. In doing so, you can fill your div with content for users without the Flash plug-in. If they have the plug-in, the SWF simply overwrites the content. See the SWFObject documentation for more information.

Do you have to use SWFObject?: If you know what you're doing, no. The most important part is that your must use JavaScript or something applicable to assign the document.

How to: Preload SlideShowPro for Flash

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 for Flash 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 for Flash on your timeline. For example, if your preloader was on frame 2, and your instance of SlideShowPro for Flash 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 for Flash using HTML/Javascript

This walkthrough will show you how to setup SlideShowPro for Flash, and the HTML document embedding your movie, so that hyperlinks in your HTML can be used to interact with SlideShowPro for Flash. By the end of the walkthrough we'll create one HTML link that calls SlideShowPro for Flash'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 for Flash onto the Stage. Position it accordingly, and give it an instance name (see the "Customization Basics" chapter 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 for Flash, then click on the hyperlink. You should now be controlling playback 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 Account Center of `slideshowpro.net`.

Event listeners

SlideShowPro for Flash 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 for Flash.

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 for Flash attempts to load an XML file. Event object contains a Boolean value (`false`, `true`) to indicate whether an XML file was found.

`onAlbumData`
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)

`onImageData`
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)

`onGalleryData`
Triggered when SlideShowPro successfully parses an XML document. Event object contains all `onAlbumData` and `onImageData` objects segmented into a multidimensional array. The structure of the array is as follows:

```
┌───────── onGalleryData ─────────┐
│ ┌──────── album array ────────┐ │
│ │                ┌─ images ─┐ │ │
│ │                │  array   │ │ │
[[album  data,[image  data]]]
        |             |
album object   image object
```

PLAYBACK EVENTS

`onAlbumEnd`
Triggered when SlideShowPro for Flash 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.

`onDisplayModeChange`
Triggered when the "Display Mode" parameter changes. Event object contains "Auto" or "Manual".

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

`onImageClick`
Triggered when a mouse clicks on slideshow content. Returns nothing.

`onImageRoll`
Triggered when a mouse pointer passes in and out of the slideshow content area. Event object contains the strings "over" when the pointer has rolled-over an image, and ″out″ when rolled-out.

`onNavButtonDisabled`
Triggered anytime SlideShowPro for Flash disables the forward or back navigation buttons. Event object contains the strings "previous" and/or "next" to indicate which button was disabled.

`onNavButtonEnabled`
Triggered anytime SlideShowPro for Flash enables the forward or back navigation buttons. Event object contains the strings "previous" and/or "next" to indicate which button was enabled.

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

`onPreloadEnd`
Triggered when the preloading of slideshow content ends. Returns nothing.

`onTransPauseEnd`
Triggered when a transition applied to slideshow content completes. Returns nothing.

`onTransPauseStart`
Triggered when a transition applied to slideshow content begins. Event object contains the length (in seconds) of the transition.

`onVideoEnd`
Triggered when a video has completed playback. Returns nothing.

`onVideoMetaData`
Triggered when a video has completed playback. Event object contains FLV metadata.

`onVideoEnd`
Triggered when a video has completed playback. Returns nothing.

`onVideoStart`
Triggered when a video begins playback. 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 for Flash:

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

Now when the `onLoadXML` event is triggered inside SlideShowPro for Flash, 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.onAlbumEnd = function(eventObject):Void {
}
my_ssp.addEventListener("onAlbumEnd", 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: ONGALLERYDATA

The `onGalleryData` 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 containing your album's data, and the second containing an array filled with the album's image data. (See beginning of this chapter for a diagram).

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

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

Or, if you wanted to return the "src" value of the third image in the third album...

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

EVENT OBJECTS: ONIMAGEDATA

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

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

```
myListener.onImageData = function(eventObject):Void {
}
my_ssp.addEventListener("onImageData", 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.onImageData = function(eventObject):Void {
	trace(eventObject.data.caption);
}
```

EVENT OBJECTS: ONALBUMDATA

`onAlbumData` is also a subset of `onGalleryData`, broadcasts data for the current album loaded in SlideShowPro for Flash, and is triggered each time an album loads.

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

ActionScript parameter list

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

```
my_ssp.albumBackgroundAlpha = 100;
my_ssp.albumBackgroundColor = "0x1A1A1A";
my_ssp.albumDescColor = "0xCCCCCC";
my_ssp.albumDescSize = 9;
my_ssp.albumPadding = 6;
my_ssp.albumPreviewScale = "Proportional";
my_ssp.albumPreviewSize = [54,41];
my_ssp.albumPreviewStrokeColor = "0xFFFFFF";
my_ssp.albumPreviewStrokeWeight = 1;
my_ssp.albumPreviewStyle = "Inline Left";
my_ssp.albumRolloverColor = "0x262626";
my_ssp.albumStrokeAppearance = "Visible";
my_ssp.albumStrokeColor = "0x333333";
my_ssp.albumTitleColor = "0xFFFFFF";
my_ssp.albumTitleSize = 9;
my_ssp.audioLoop = "Off";
my_ssp.audioPause = "Off";
my_ssp.audioVolume = 50;
my_ssp.autoFinishMode = "Switch";
my_ssp.cacheContent = "None";
my_ssp.captionAppearance = "Overlay Mouse Over (If Available)";
my_ssp.captionBackgroundAlpha = 75;
my_ssp.captionBackgroundColor = "0xFFFFFF";
my_ssp.captionHeaderAppearance = "Image Count";
my_ssp.captionPadding = [5,5,5,5];
my_ssp.captionPosition = "Top";
my_ssp.captionTextAlignment = "Left";
my_ssp.captionTextColor = "0x333333";
my_ssp.captionTextSize = 9;
my_ssp.contentAlign = "Center";
my_ssp.contentAreaBackgroundAlpha = 100;
my_ssp.contentAreaBackgroundColor = "0x1A1A1A";
my_ssp.contentAreaStrokeAppearance = "Visible";
my_ssp.contentAreaStrokeColor = "0x262626";
my_ssp.contentFormat = "Normal";
my_ssp.contentFrameAlpha = 100;
my_ssp.contentFrameColor = "0x262626";
my_ssp.contentFramePadding = 0;
my_ssp.contentFrameStrokeAppearance = "Hidden";
my_ssp.contentFrameStrokeColor = "0x333333";
my_ssp.contentOrder = "Sequential";
my_ssp.contentScale = "Downscale Only";
my_ssp.contentWatermark = "";
my_ssp.contentWatermarkAlign = "Bottom Right";
my_ssp.directorLargeImageSettings = [80,1,100];
my_ssp.directorThumbImageSettings = [50,0];
my_ssp.displayMode = "Auto";
my_ssp.feedbackBackgroundAlpha = 30;
my_ssp.feedbackBackgroundColor = "0x000000";
my_ssp.feedbackHighlightAlpha = 80;
my_ssp.feedbackHighlightColor = "0xFFFFFF";
my_ssp.feedbackPreloaderAlign = "Center";
my_ssp.feedbackPreloaderAppearance = "Pie";
my_ssp.feedbackPreloaderPosition = "Center";
my_ssp.feedbackScale = 100;
my_ssp.feedbackTimerAlign = "Top Right";
my_ssp.feedbackTimerAppearance = "Visible";
my_ssp.feedbackTimerPosition = "Top Right";
my_ssp.galleryAppearance = "Closed on Startup";
my_ssp.galleryBackgroundAlpha = 100;
my_ssp.galleryBackgroundColor = "0x000000";
my_ssp.galleryColumns = 2;
my_ssp.galleryOrder = "Left to Right";
my_ssp.galleryPadding = 10;
my_ssp.galleryRows = 4;
my_ssp.galleryScreenNavAppearance = "Visible";
my_ssp.iconInactiveAlpha = 40;
my_ssp.iconShadowAlpha = 60;
my_ssp.keyboardControl = "On";
my_ssp.mediaPlayerAppearance = "Visible";
my_ssp.mediaPlayerBackgroundAlpha = 25;
my_ssp.mediaPlayerBackgroundColor = "0x000000";
my_ssp.mediaPlayerBufferColor = "0x000000";
my_ssp.mediaPlayerControlColor = "0xFFFFFF";
my_ssp.mediaPlayerElapsedBackgroundColor = "0xFFFFFF";
my_ssp.mediaPlayerElapsedTextColor = "0x000000";
my_ssp.mediaPlayerIconColor = "0xCCCCCC";
my_ssp.mediaPlayerPosition = "Bottom";
my_ssp.mediaPlayerProgressColor = "0xCCCCCC";
my_ssp.mediaPlayerScale = 80;
my_ssp.mediaPlayerTextColor = "0x999999";
my_ssp.mediaPlayerTextSize = 9;
my_ssp.mediaPlayerVolumeBackgroundColor = "0x000000";
my_ssp.mediaPlayerVolumeHighlightColor = "0xCCCCCC";
my_ssp.navAppearance = "Always Visible";
my_ssp.navActiveColor = "0xFFFFFF";
my_ssp.navBackgroundAlpha = 100;
my_ssp.navBackgroundColor = "0x121212";
my_ssp.navButtonsAppearance = "Always Visible";
my_ssp.navGradientAlpha = 30;
my_ssp.navGradientAppearance = "Glass Dark";
my_ssp.navIconColor = "0xEEEEEE";
my_ssp.navLinkAppearance = "Numbers";
my_ssp.navLinkCurrentColor = "0xEEEEEE";
my_ssp.navLinkPreviewAppearance = "Visible";
my_ssp.navLinkPreviewBackgroundAlpha = 100;
my_ssp.navLinkPreviewBackgroundColor = "0xFFFFFF";
my_ssp.navLinkPreviewScale = "Proportional";
my_ssp.navLinkPreviewSize = [80,60];
my_ssp.navLinkPreviewStrokeWeight = 1;
my_ssp.navLinkRolloverColor = "0xFFFFFF";
my_ssp.navLinksBackgroundAlpha = 100;
my_ssp.navLinksBackgroundColor = "0x000000";
my_ssp.navLinkSpacing = 10;
my_ssp.navNumberLinkColor = "0x999999";
my_ssp.navNumberLinkSize = 9;
my_ssp.navPosition = "Bottom";
my_ssp.navThumbLinkBackgroundColor = "0x666666";
my_ssp.navThumbLinkInactiveAlpha = 100;
my_ssp.navThumbLinkShadowAlpha = 60;
my_ssp.navThumbLinkSize = [20,20];
my_ssp.navThumbLinkStrokeWeight = 1;
my_ssp.permalinks = "Off";
my_ssp.soundEffects = "";
my_ssp.textStrings = ["Previous Screen","Next Screen","Screen","of","No caption"];
my_ssp.transitionLength = 2;
my_ssp.transitionPause = 4;
my_ssp.transitionStyle = "Cross Fade";
my_ssp.typeface = "Lucida Grande,Lucida Sans Unicode,Verdana,Arial,_sans";
my_ssp.typefaceEmbed = "Off";
my_ssp.videoAutoStart = "On";
my_ssp.videoBufferTime = 0.1;
my_ssp.videoSmoothing = "On";
my_ssp.xmlFilePath = "images.xml";
my_ssp.xmlFileType = "Default";
```