

# PhotoFlowPRO

User Guide revision 1.0

[www.flashloaded.com](http://www.flashloaded.com)

# Table of Contents

|  |           |
|--|-----------|
| <b>Installation</b>                          | <b>3</b>  |
| <b>Getting started</b>                       | <b>4</b>  |
| <b>Component Inspector Parameters</b>        | <b>10</b> |
| <b>Using XML</b>                             | <b>17</b> |
| <b>Working with Video</b>                    | <b>28</b> |
| <b>Working with YouTube</b>                  | <b>33</b> |
| <b>Working with Reflections</b>              | <b>36</b> |
| <b>Working with Names &amp; Descriptions</b> | <b>38</b> |
| <b>Working with Scrollbars</b>               | <b>41</b> |
| <b>Skinning the Components</b>               | <b>44</b> |
| <b>ActionScript events</b>                   | <b>45</b> |
| <b>PhotoFlowProEvent</b>                     | <b>45</b> |
| <b>PhotoFlowProItemEvent</b>                 | <b>46</b> |
| <b>ActionScript properties</b>               | <b>50</b> |
| <b>ActionScript methods</b>                  | <b>65</b> |
| <b>Help</b>                                  | <b>75</b> |

# Installation

You will need Adobe Extension Manager in order to install this component. Extension Manager should have been installed by default when you installed Flash. You may download the latest version of Extension Manager for free from the [Adobe website](#).

1. Ensure that Flash is closed before installing the PhotoFlowPRO component.
2. Unzip/extract the PhotoFlowPRO.zip file that you downloaded. You will find a file called PhotoFlowPRO.mxp. Double click on this file in order to install the component using Extension Manager.

The PhotoFlowPRO should now be installed in your Flash Components Panel.

# Getting started

Once you have installed the component by following the instructions in the [Installation section](#) of this user guide you're ready to begin using the component.

The simplest way to use the component is to add images to it using the component's parameters, this tutorial will teach you to do that.

*You will find a working example file for this tutorial named 'getting\_started fla' in the Examples folder that came with your download.*

1. The first step is to prepare the images that you wish to include. You can prepare these in an application like PhotoShop or in Flash or to save time you can use the example images provided with the component in the Examples folder.
2. Save your images in a folder called 'images'. For the purposes of this tutorial you'll need at least three images named 'image1.jpg', 'image2.jpg' and 'image3.jpg'.
3. Open your copy of Adobe Flash.
4. Start a new Actionscript 3 document.

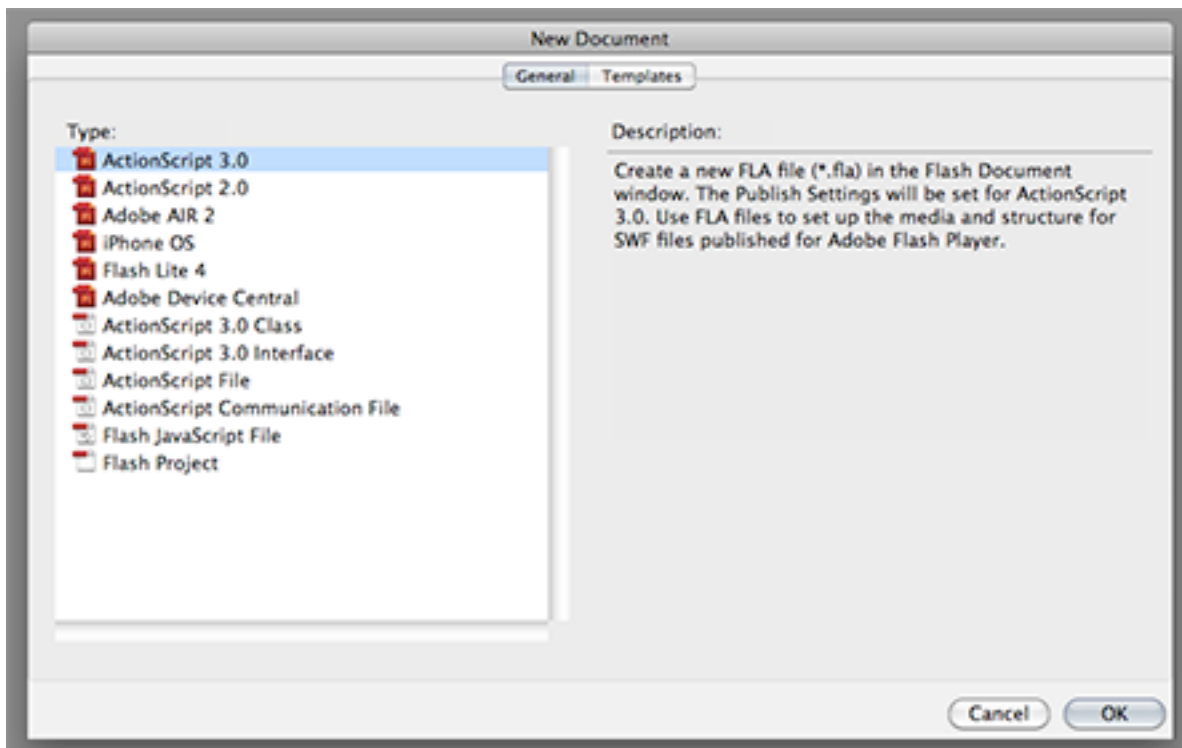


Fig. 1: Starting a new Actionscript 3 file in Flash CS5.

5. Open the properties panel (Ctrl+F3 Windows or Cmnd+F3 on Mac) and set your stage to the desired size. For the purposes of this tutorial set it to 900px by 700px.

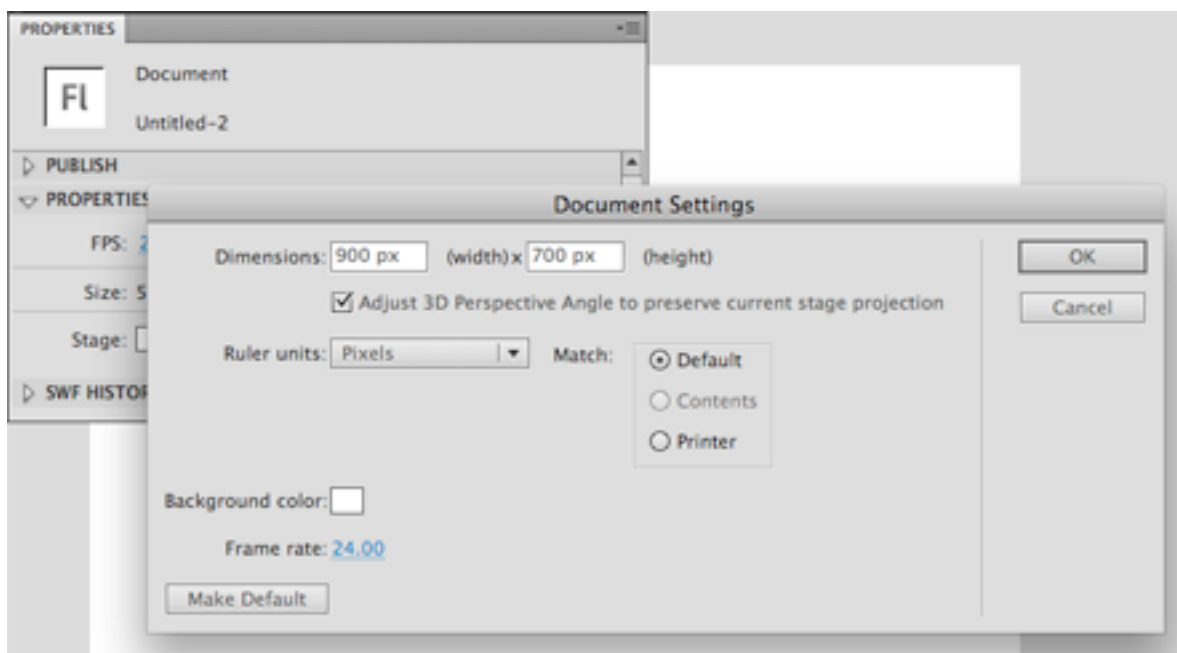


Fig. 2: Setting the size of your document in Flash CS5.

6. We recommend that you set the FPS (frames per second) of your file to 31 for smooth animation.
7. Open your components panel (Ctrl+F7 on Windows or Cmnd+F7 on Mac).
8. Locate the 'PhotoFlowPRO' component inside the 'photoFlowPRO' folder.

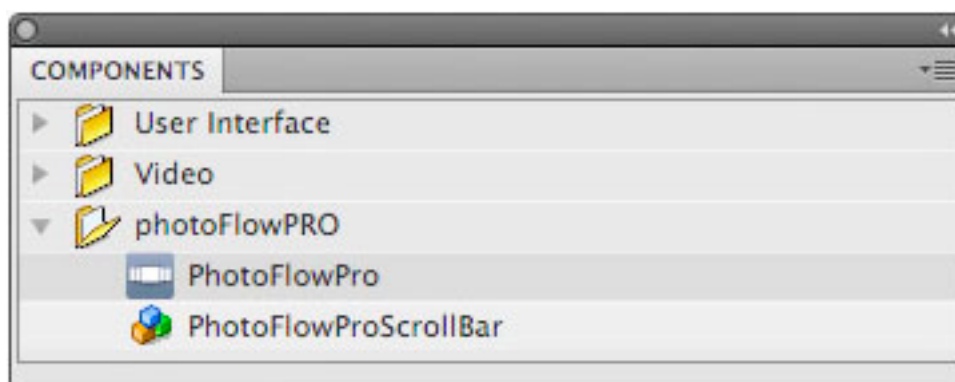


Fig. 3: The PhotoFlowPRO component inside the photoFlowPRO folder in the components panel.

9. Drag a copy of the component onto the stage.
10. With the component still selected, open the properties panel and set the component's x and y to 0. (The component's width and height are 550 x 400 by default so the component should now fill your stage.)

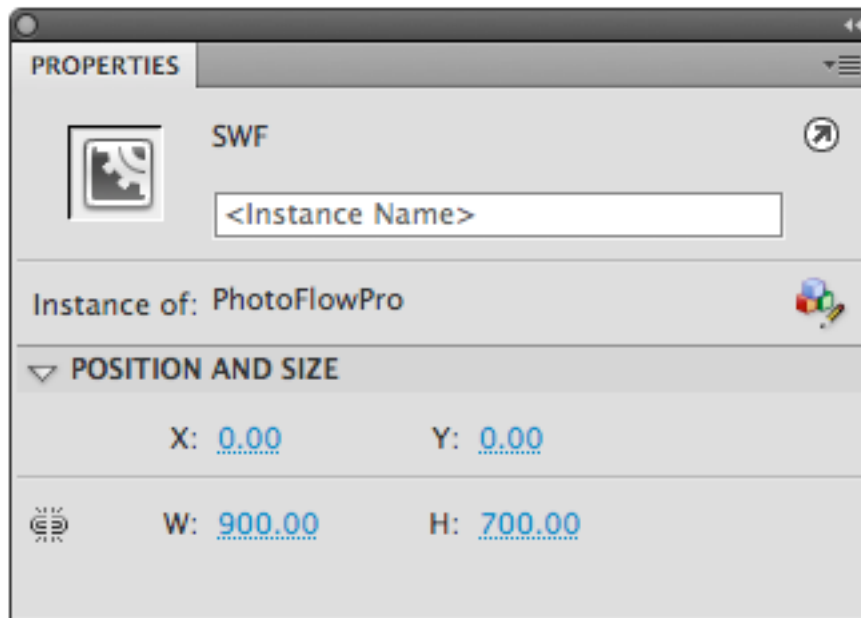


Fig. 4: Setting the PhotoFlowPRO position in the properties panel in Flash CS4.

11. With the component still selected on the stage locate its 'Photos' parameter. If you are using Adobe Flash CS5 the parameters are in the properties panel, if you are using Adobe Flash CS4 or CS3 you'll find the parameters in the component inspector panel (Shift+F7).
12. In Flash CS5 click the pencil to the right of the parameter to open the values dialog box, in CS4 or CS3 click the magnifying glass.
13. In the values dialog box click the '+' button to add an item. Enter a "Image 1" for the name of the item, 'This is the first image' for the description and 'image1.jpg' for the url.

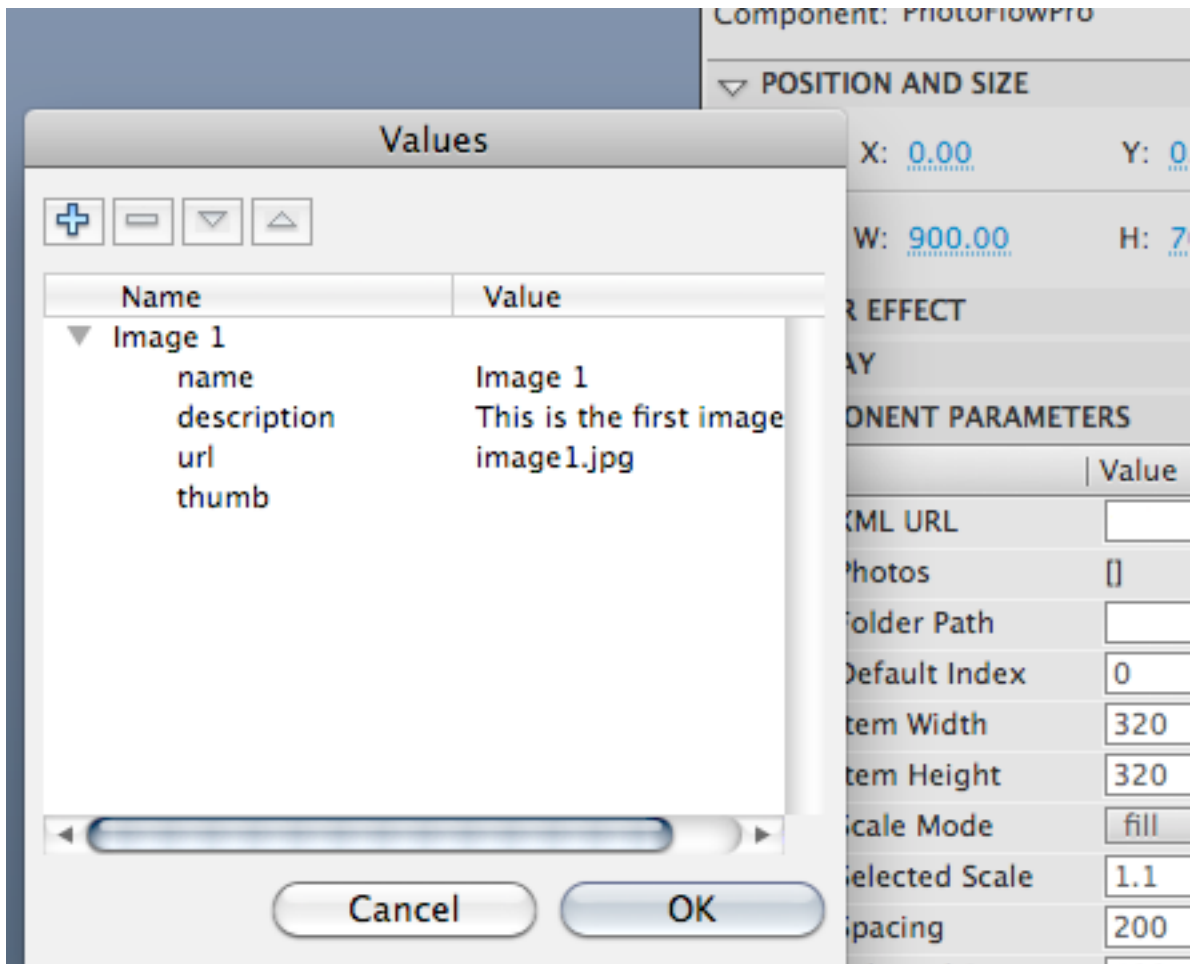


Fig. 5: Adding an item with the Photos parameter in Flash CS5.

14. Click the '+' button twice more to add two new items and add names, descriptions and enter the file names of your other two images for the urls.

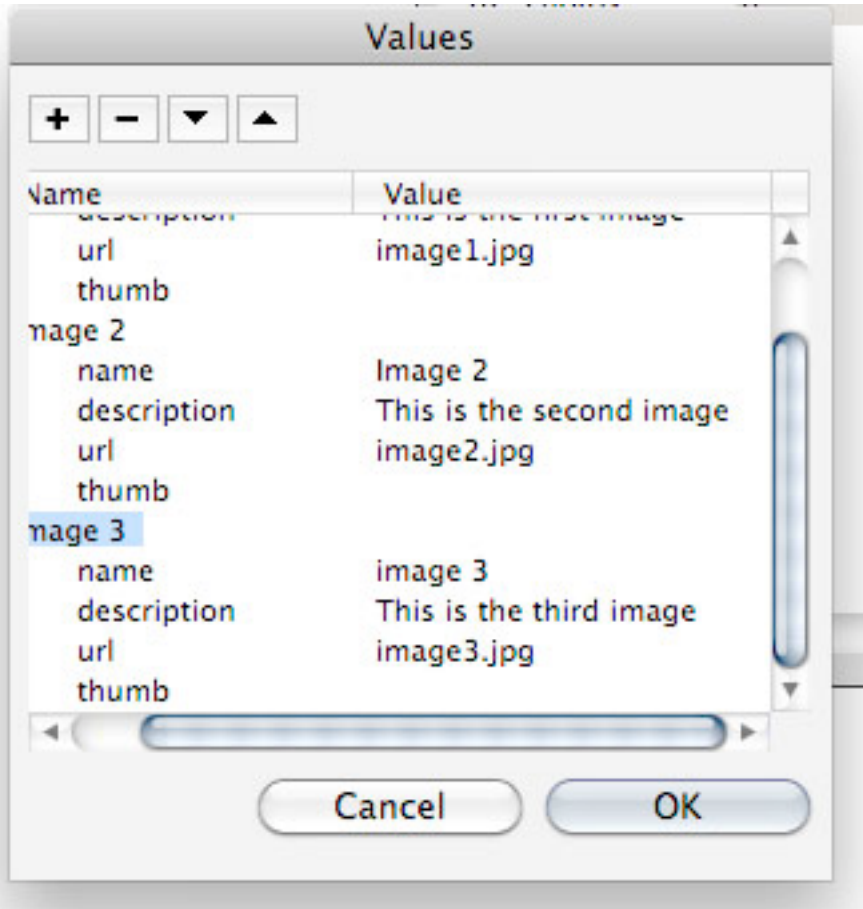


Fig. 6: Adding items with the Photos parameter in Flash CS4.

15. Click 'OK'.
16. With the component still selected locate the 'Folder Path' parameter in the properties panel (if you're using CS5) or the component inspector (if you're using CS4 or CS3).
17. Enter the name of the folder where you've saved your images. It should be named 'images/'.

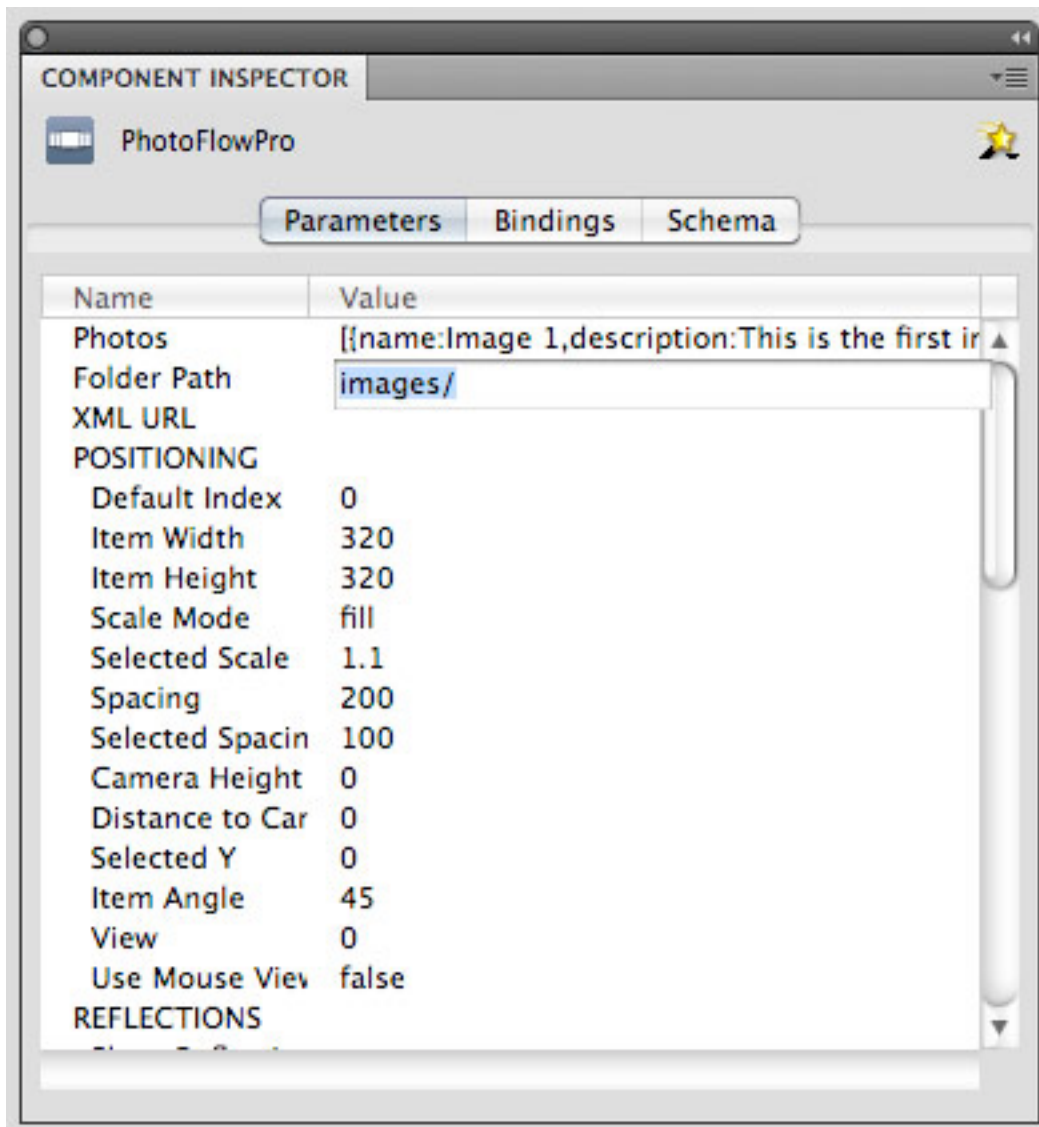


Fig. 7: Adding the path to your images in the component inspector in Flash CS4.

18. Save your file in the same location as the 'images' folder (not inside the 'images' folder).
19. You can now test the component using the default settings by pressing Ctrl+Enter (Windows) or Cmnd+Enter (Mac).

# Component Inspector Parameters

| Parameter   | Description   | Example   |
|-------------|---|---|
| Photos      | A list of items to be displayed if not specifying an XML URL. Each item includes name, description, url and thumb.  | name:My Image<br>description:An Image<br>url:image.jpg<br>thumb:thumb.jpg |
| Folder Path | The path to the folder that contains your images, SWFs or videos. If you're using XML this setting is specified in the XML file.                                  | folderName/   |
| XML URL     | Specifies an XML file that defines the items to be displayed by the component. For more information on using XML refer to the <a href="#">Using XML section</a> . | folder/file.xml   |

## POSITIONING

| Parameter        | Description  | Example |
|------------------|--|---------|
| Default Index    | The default image, swf or video to be selected when the component first initializes.   | 3       |
| Item Width       | The width of each item.  | 320     |
| Item Height      | The height of each item  | 240     |
| Scale Mode       | The mode of scaling to be applied to the images, SWFs and videos displayed by the component. Possible values are fill, fit and no scale. | fill    |
| Selected Scale   | The scale of the selected item.  | 1.5     |
| Spacing          | The amount of spacing between items  | 50      |
| Selected Spacing | The amount of spacing between the selected item and the non-selected items.  | 100     |
| Camera Height    | The height of the 3D point of view.  | 100     |

| Parameter          | Description  | Example |
|--------------------|--|---------|
| Distance to Camera | The depth of the 3D point of view.   | 50      |
| Selected Y         | The position of the selected item on the y axis.                           | 0       |
| Item Angle         | The angle of the non-selected items.                                       | 60      |
| View               | The 3D point of view.  | 0       |
| Selected View      | The 3D point of view for the selected image, swf or video.                 | 0       |
| Use Mouse View     | Tilts the perspective of the camera relative to the position of the mouse. | TRUE    |

## REFLECTIONS

| Parameter                 | Description  | Example |
|---------------------------|--|---------|
| Show Reflection           | Adds a reflection to each image.   | TRUE    |
| Reflection Refresh        | <p>The frequency at which the reflection is updated, measured in milliseconds.</p> <p>To ensure that the reflection updates only when loaded set this property to 0.</p> <p>If the item being reflected is an image (as opposed to a SWF or FLV file) this property is automatically set to 0.</p> <p>Lower values cause the reflection to update more frequently, Higher values reduce the system resources consumed.</p> <p>For more information refer to the <a href="#">Working with Reflections section</a> of this user guide.</p> | 84      |
| Reflection Alpha          | The transparency of the reflection.  | 0.8     |
| Selected Reflection Alpha | The transparency of the reflection of the currently selected item.   | 1       |
| Reflection Depth          | The distance down the y axis that the reflection extends measured in pixels.   | 50      |

| Parameter                 | Description   | Example |
|---------------------------|---|---------|
| Selected Reflection Depth | The distance down the y axis that the reflection extends for the currently selected item, measured in pixels. | 75      |
| Reflection Distance       | The gap between the item and its reflection, measured in pixels.  | 10      |

## PLACEHOLDERS

| Parameter           | Description   | Example  |
|---------------------|---|----------|
| Holder Color        | The color of the item's background.                       | 0xdddddd |
| Holder Alpha        | The transparency of the item's background                 | 0.5      |
| Holder Border       | The thickness of the border around the item's background. | 2        |
| Holder Border Color | The color of the border around the item's background.     | 0xcccccc |

## TEXT

| Parameter  | Description   | Example |
|------------|---|---------|
| Name Field | <p>A text field to display the name of the currently selected item.</p> <p>The Name Field property will accept any DisplayObject that has a 'text' property, including Classic and TLF text.</p> <p>For more information refer to the <a href="#">Working with Names &amp; Descriptions section</a> of this user guide.</p> | nameTxt |

| Parameter         | Description  | Example        |
|-------------------|--|----------------|
| Description Field | <p>A text field to display the description of the currently selected item.</p> <p>The Description Field will accept any DisplayObject that has a 'text' property, including Classic and TLF text.</p> <p>For more information refer to the <a href="#">Working with Names &amp; Descriptions section</a> of this user guide.</p> | descriptionTxt |

## ANIMATION

| Parameter       | Description   | Example        |
|-----------------|---|----------------|
| Slide Time      | The number of seconds taken to animate to the next item.  | 0.5            |
| Slide Easing    | The kind of easing used to animate to the next item.  | Sine.easeInOut |
| Slide Direction | <p>The initial direction of movement when autoflipping to the next item.</p> <p>Valid values are left and right.</p>  | left           |
| Auto Flip       | Specifies whether the component should automatically animate to the next item (true) or only animate when an item is clicked (false).                         | TRUE           |
| Auto Flip Delay | <p>The amount of time to wait before animating to the next item. Measured in milliseconds.</p> <p>Only applies if the Auto Flip parameter is set to true.</p> | 4000           |

## SOUND AND VIDEO

| Parameter             | Description  | Example |
|-----------------------|--|---------|
| Video Always Restarts | Determines whether video should loop (true) or not (false).  | TRUE    |
| Smoothing             | Specifies whether smoothing should be applied to images and videos loaded into the component.  | TRUE    |
| Flip Sound Class      | A reference to a Sound class to play when the component animates to another item. If no Sound class is specified then no sound will be played. | MySound |

## INTERACTION

| Parameter               | Description   | Example  |
|-------------------------|---|----------|
| Scrollbar               | <p>The instance name of a scrollbar to scroll through the items.</p> <p>The Scrollbar parameter will accept any DisplayObject that has a maximum; maxScrollPosition or maxScroll property and a minimum; minScrollPosition or minScroll property and a value; scrollPosition or position property.</p> <p>For more information refer to the <a href="#">Working with Scrollbars section</a> of this user guide.</p> | mySlider |
| Show Video Controls     | Specifies whether to add controls for any videos (true) or not (false).   | FALSE    |
| Autohide Video Controls | Specifies whether the automatically show or hide themselves on mouse over (true) or not (false).  | TRUE     |
| Autohide Delay          | The number of milliseconds to delay before fading out the video controls when the mouse rolls off the component.  | 2500     |
| Video Controls Distance | The distance to position the video controls from the vertical center of the component.  | 250      |

| Parameter       | Description   | Example |
|-----------------|---|---------|
| Use Keyboard    | Specifies whether you can navigate to the next and previous items using the keyboard's left and right arrow keys & the distance to camera using the up and down arrows. | TRUE    |
| Use Mouse Wheel | Specifies whether you can navigate between items using the mouse wheel.   | FALSE   |

## Enabling mouse wheel scrolling for Mac browsers

The current version of the Flash Player does not natively support mouse wheel scrolling in Mac browsers. We have built a solution for this into PhotoFlowPRO. In order to use this solution, you must construct your HTML file like this:

1. Copy the **js** folder, that was included with your download, to the same folder in which your HTML file will reside.
2. Write the following code in the <head></head> section of your HTML file:

```
<script type="text/javascript" src="js/swfobject.js"></script>
<script type="text/javascript" src="js/swfmacmousewheel2.js"></script>
<script type="text/javascript">
    var vars = {};
    var params = { scale:'noScale', salign:'lt', menu:'true' };
    var attributes = { id:'pfObject', name:'pfObject' };
    swfmacmousewheel.registerObject(attributes.id);
</script>
```

3. Write the following code in the body of your HTML file, where you would like the Flash SWF to be located:

```
<script>swfobject.embedSWF("pfexample.swf", "flashContent", "1000",
"600", "9.0.0", "js/expressInstall.swf", vars, params, attributes );</
script>
```

Note: Change the items marked in bold to match your SWF filename, height and width.

This will work when testing online only. You must ensure that you upload the **js** folder with your HTML file.

## Using XML

The best way to maximize the component's flexibility is to set the component up using an XML file. By using XML to define the images in the PhotoFlowPRO you can publish your SWF file once and update it whenever you like using the XML file.

This tutorial will teach you how to define the component using XML.

*You will find working example files for this tutorial named 'using\_xml fla' and 'using\_xml.xml' in the Examples folder that came with your download.*

1. Firstly you need to create the images you wish to load into the component. Create them in an application like Photoshop or in Flash or for the purposes of this tutorial use the images from the example files that came with your download.
2. Save your images in a folder called 'images'. For the purposes of this tutorial you'll need at least three images named 'image1.jpg', 'image2.jpg' and 'image3.jpg'.
3. Open your copy of Adobe Flash.
4. Start a new Actionscript 3 document.

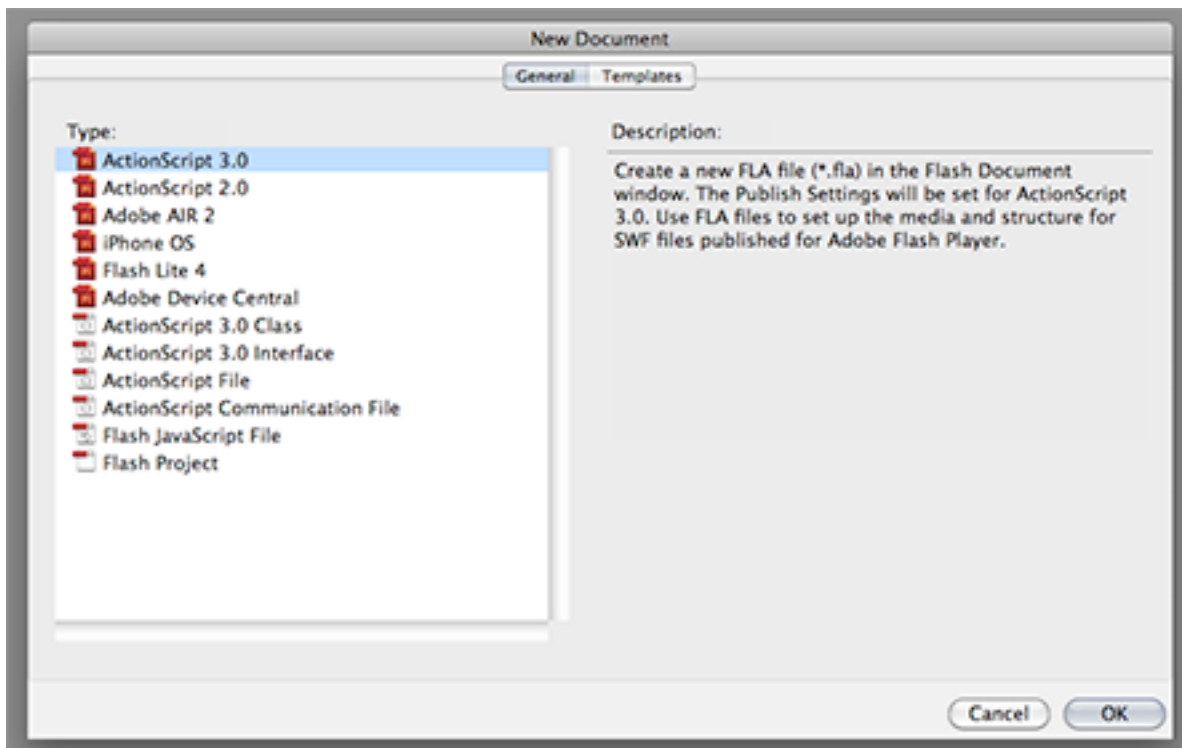


Fig. 8: Starting a new Actionscript 3 file in Flash CS5.

5. Open the properties panel (Ctrl+F3 Windows or Cmnd+F3 on Mac) and set your stage to the desired size. For the purposes of this tutorial set it to 900px by 700px.

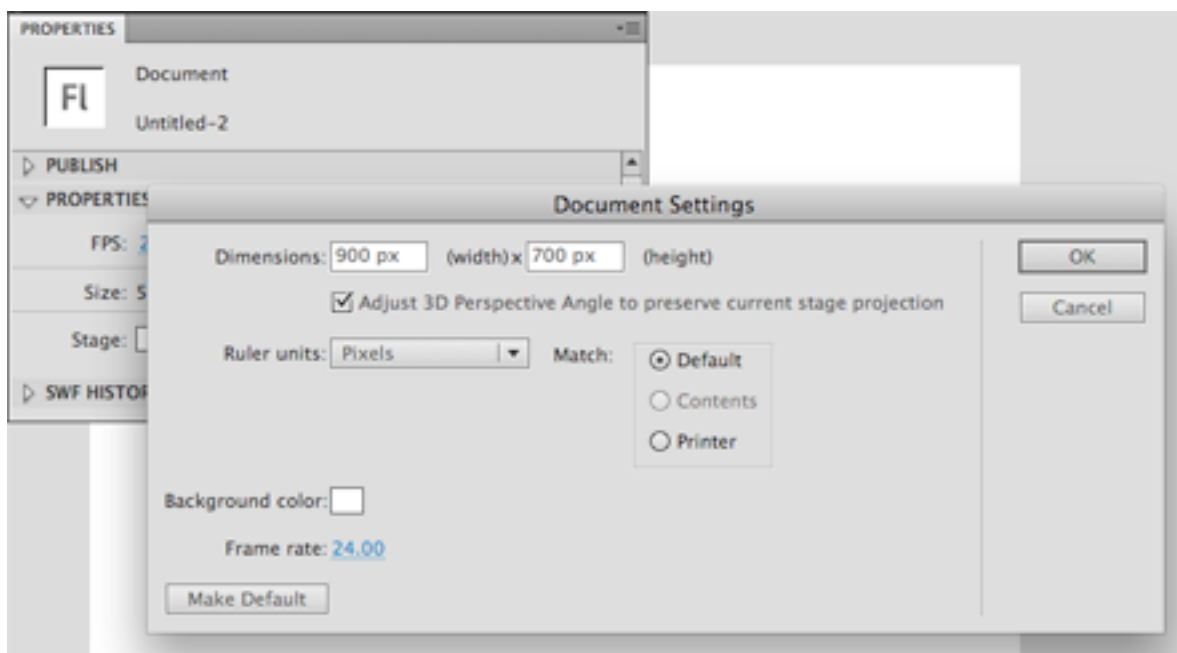


Fig. 9: Setting the size of your document in Flash CS5.

6. We recommend that you set the FPS (frames per second) of your file to 31 for smooth animation.
7. Open your components panel (Ctrl+F7 on Windows or Cmnd+F7 on Mac).
8. Locate the 'PhotoFlowPRO' component inside the 'photoFlowPRO' folder.

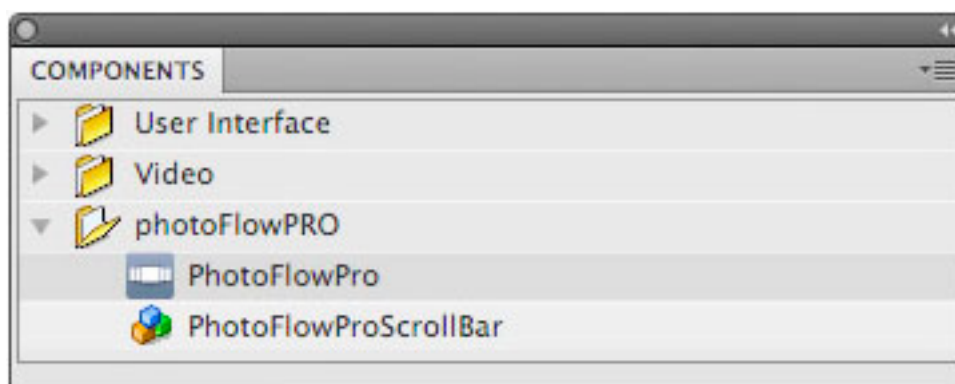


Fig. 10: The PhotoFlowPRO component inside the photoFlowPRO folder in the components panel.

9. Drag a copy of the component onto the stage.
10. With the component still selected, open the properties panel and set the component's x and y to 0. (The component's width and height are 550 x 400 by default so the component should now fill your stage.)

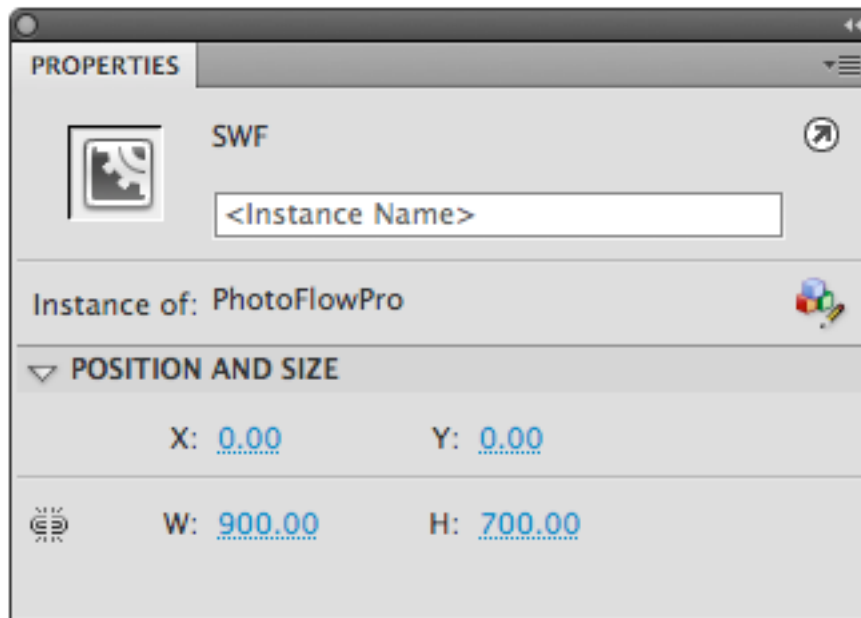


Fig. 11: Setting the PhotoFlowPRO position in the properties panel in Flash CS4.

11. With the component still selected on the stage locate the 'XML URL' parameter, if you're using Flash CS5 the parameters are located in the properties panel, if you're using CS4 or CS3 then the parameters are located in the component inspector panel (Shift+F7).
12. Enter the name of your XML file. As you haven't created an XML file yet enter 'using\_xml.xml' (without the quotes).

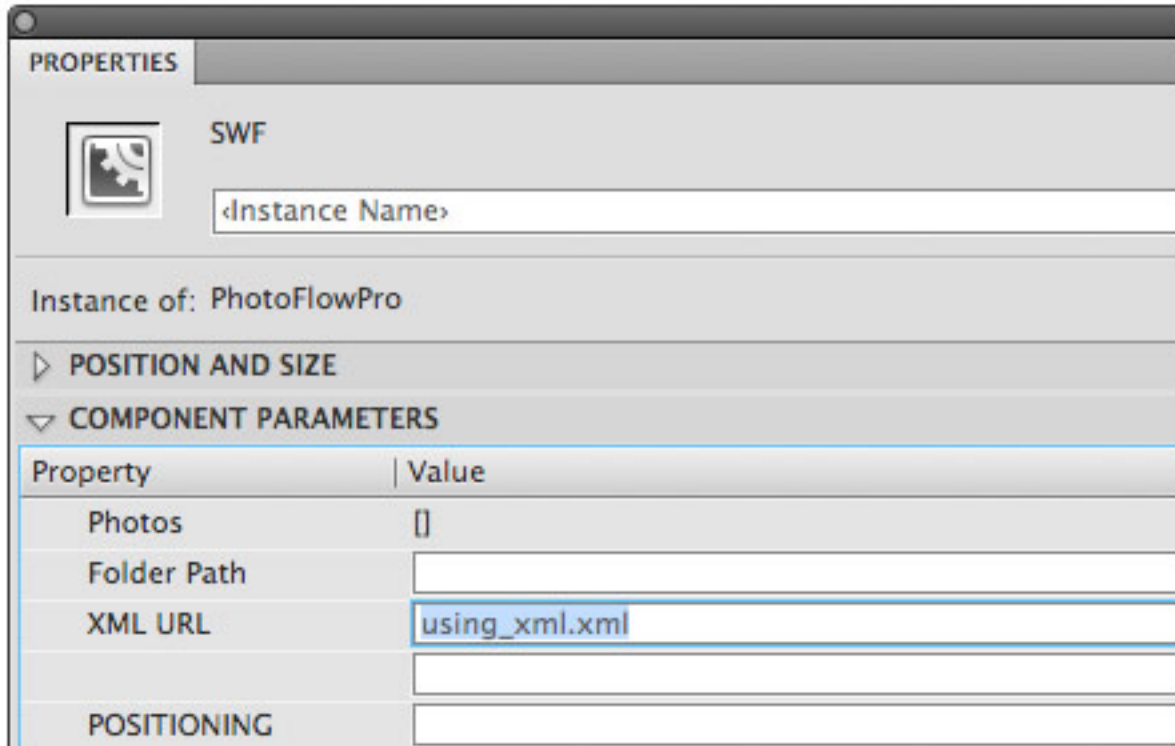


Fig.12: Setting the XML URL in the properties panel in Flash CS5.

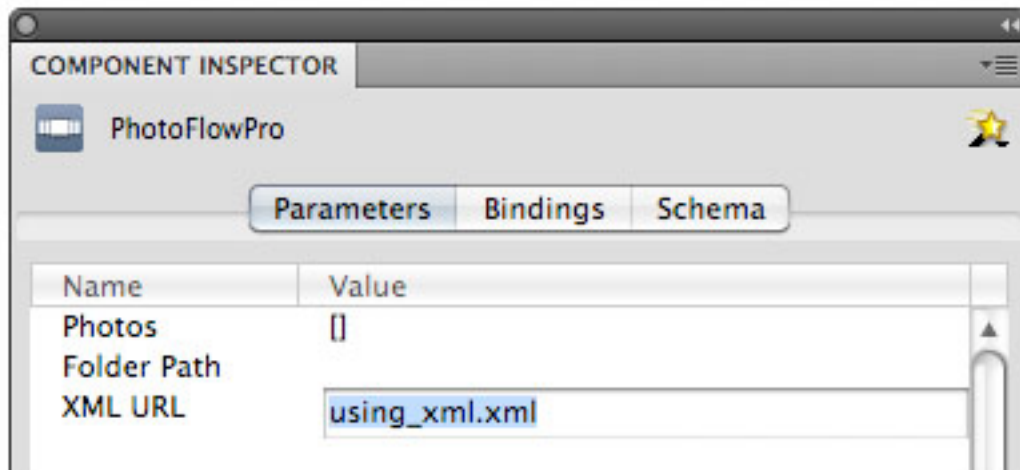


Fig. 13: Setting the XML URL in the properties panel in Flash CS4.

13. Save your .fla file in the same directory as your images folder (not inside the images folder).

14. Open an application that produces plain text, for example on Mac that might be TextEdit, on Windows that might be Notepad. If you have Dreamweaver, you could use that. *Note that you should not use a word processor such as Microsoft's Word or Apple's Pages as the formatting will break your XML document.*

15. Start a new file.

16. Enter the following text:

```
<?xml version="1.0" encoding="utf-8"?>
```

This is the standard XML opening line.

17. Next add a 'photos' element like so (the new lines are in bold):

```
<?xml version="1.0" encoding="utf-8"?>
<photos>
</photos>
```

This element defines the component as a whole. (Later, we will add additional attributes to the photos element.)

18. At this point the only attribute you need to add is the path to your images which you do like this (the change is highlighted in bold):

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="images/">
</photos>
```

19. Next we need to add a child element to the photos element to define an individual item (the new code is in bold):

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="images/">
<photo></photo>
</photos>
```

20. To specify the image to be loaded into the component we need to add a url attribute:

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="images/">
<photo url="image1.jpg"></photo>
</photos>
```

21. Next, add a name attribute:

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="images/">
<photo name="Image 1" url="image1.jpg"></photo>
</photos>
```

22. Now, add the description as a child of the photo element:

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="images/">
<photo name="Image 1" url="image1.jpg">This is the 1st image</photo>
</photos>
```

23. Now repeat the last four steps to add two more images so that your XML file looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="images/">
<photo name="Image 1" url="image1.jpg">This is the 1st image</photo>
<photo name="Image 2" url="image2.jpg">This is the 2nd image</photo>
<photo name="Image 3" url="image3.jpg">This is the 3rd image</photo>
</photos>
```

24. Finally save your XML file as 'using\_xml.xml' (without the quotes) in the same directory as the images folder and the .fla

25. Return to your .fla and test your file (Ctrl+Enter on Windows or Cmnd+Enter on Mac).

## Setting Parameters in XML

Up until now you have only set one of the component's parameters, the 'path' parameter inside the XML file. However, it is possible to set all of the component's properties in the XML file.

One of the most useful settings to enter in the XML file is the default index which specifies which item should be selected once the component appears on stage. These steps continue from the steps above:

26. Indexing in Flash begins at zero so to make sure the middle image is open first when our file runs we need to add the index attribute with a value of 1 to the photos element like so:

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="images/" index="1">
<photo name="Image 1" url="image1.jpg">This is the 1st image</photo>
<photo name="Image 2" url="image2.jpg">This is the 2nd image</photo>
<photo name="Image 3" url="image3.jpg">This is the 3rd image</photo>
</photos>
```

27. Save the XML file.

28. Return to your .fla and test your file (Ctrl+Enter on Windows or Cmnd+Enter on Mac).

The following optional attributes can be set in the XML photos element, (you can use one, several, all or none of the optional attributes).

*For a definition of each of these attributes please refer to the property of the same name in the [Actionscript properties section](#) of the API reference.*

### **autoFlip**

```
<photos path="images/" autoFlip="true">
```

Valid values are "true" and "false".

### **autoFlipDelay**

```
<photos path="images/" autoFlipDelay="10">
```

A valid value is any positive number.

### **autohideVideoControls**

```
<photos path="images/" autohideVideoControls="true">
```

Valid values are "true" and "false".

### **autohideDelay**

```
<photos path="images/" autohideDelay="10000">
```

A valid value is any positive number.

### **cameraHeight**

```
<photos path="images/" cameraHeight="100">
```

A valid value is any number.

### **distanceToCamera**

```
<photos path="images/" distanceToCamera="500">
```

A valid value is any positive number.

### **holderAlpha**

```
<photos path="images/" holderAlpha="0.5">
```

A valid value is any number ranging from 0 to 1.

### **holderBorder**

```
<photos path="images/" holderBorder="3">
```

A valid value is any positive number.

### **holderBorderAlpha**

```
<photos path="images/" holderBorderAlpha="0.5">
```

A valid value is any number ranging from 0 to 1.

### **holderBorderColor**

```
<photos path="images/" holderBorderColor="0xFF0000">
```

A valid value is any number ranging from 0 to 0xFFFFFFFF.

### **holderColor**

```
<photos path="images/" holderColor="0x990000">
```

A valid value is any number ranging from 0 to 0xFFFFFFFF.

### **index**

```
<photos path="images/" index="1">
```

A valid value is any number ranging from 0 to one less than the total number of items.

### **itemAngle**

```
<photos path="images/" itemAngle="45">
```

A valid value is any positive number.

### **itemHeight**

```
<photos path="images/" itemHeight="180">
```

A valid value is any positive number.

### **itemWidth**

```
<photos path="images/" itemWidth="180">
```

A valid value is any positive number.

### **reflectionAlpha**

```
<photos path="images/" reflectionAlpha="0.4">
```

A valid value is any number ranging from 0 to 1.

### **reflectionDepth**

```
<photos path="images/" reflectionDepth="125">
```

A valid value is any positive number.

### **reflectionDistance**

```
<photos path="images/" reflectionDistance="20">
```

A valid value is any positive number.

### **reflectionRefresh**

```
<photos path="images/" reflectionRefresh="1000">
```

A valid value is any positive number.

### **scaleMode**

```
<photos path="images/" scaleMode="fill">
```

Valid values are "fit", "fill" and "no scale".

### **selectedReflectionAlpha**

```
<photos path="images/" selectedReflectionAlpha="0.8">
```

A valid value is any number ranging from 0 to 1.

### **selectedReflectionDepth**

```
<photos path="images/" selectedReflectionDepth="75">
```

A valid value is any positive number.

### **selectedScale**

```
<photos path="images/" selectedScale="1.2">
```

A valid value is any number.

### **selectedSpacing**

```
<photos path="images/" selectedSpacing="20">
```

A valid value is any positive number.

### **selectedView**

```
<photos path="images/" selectedView="0">
```

### **selectedY**

```
<photos path="images/" selectedY="-50">
```

A valid value is any number.

### **showReflection**

```
<photos path="images/" showReflection="true">
```

Valid values are "true" and "false".

### **showVideoControls**

```
<photos path="images/" showVideoControls="true">
```

Valid values are "true" and "false".

### **slideEasing**

```
<photos path="images/" slideEasing="Sine.easeInOut">
```

Valid values are contained in the fl.motion.easing package. Refer to [Adobe documentation](#) for more information.

### **slideDirection**

```
<photos path="images/" slideDirection="left">
```

Valid values are "left" and "right".

### **slideTime**

```
<photos path="images/" slideTime="0.25">
```

A valid value is any positive number.

### **smoothing**

```
<photos path="images/" smoothing="true">
```

Valid values are "true" and "false".

### **spacing**

```
<photos path="images/" spacing="20">
```

A valid value is any positive number.

### **useKeyboard**

```
<photos path="images/" useKeyboard="true">
```

Valid values are "true" and "false".

### **useMouseView**

```
<photos path="images/" useMouseView="true">
```

Valid values are "true" and "false".

**useMouseWheel**

```
<photos path="images/" useMouseWheel="true">
```

Valid values are "true" and "false".

**videoAlwaysRestarts**

```
<photos path="images/" videoAlwaysRestarts="true">
```

Valid values are "true" and "false".

**videoControlsDistance**

```
<photos path="images/" videoControlsDistance="180">
```

A valid value is any.number.

**view**

```
<photos path="images/" view="0">
```

A valid value is any positive number.

## Working with Video & SWFs

One of the most exciting features of the PhotoFlowPRO is its ability to play video as well as images and SWFs. Videos and SWFs can be loaded into the component in the same way as images. Images, SWFs and videos can be mixed and matched, you are not restricted to a single kind.

### Adding Videos and SWFs via Parameters

To add a video or SWF to the component using its parameters follow the same steps as described in the Getting Started section of this user guide. Instead of preparing images, save .flv or .swf files in your 'images' folder and instead of specifying file names for images specify file names for FLVs or SWFs.

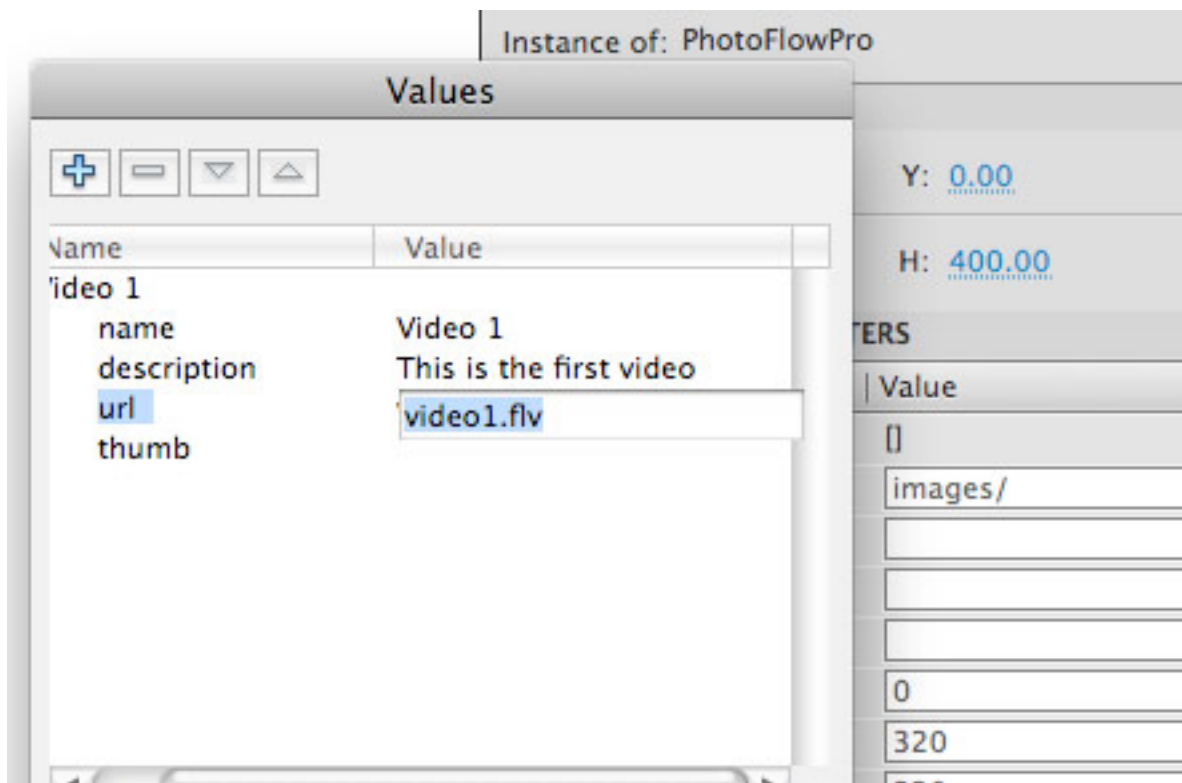


Fig. 14: Setting a video to play using parameters in Flash CS5.

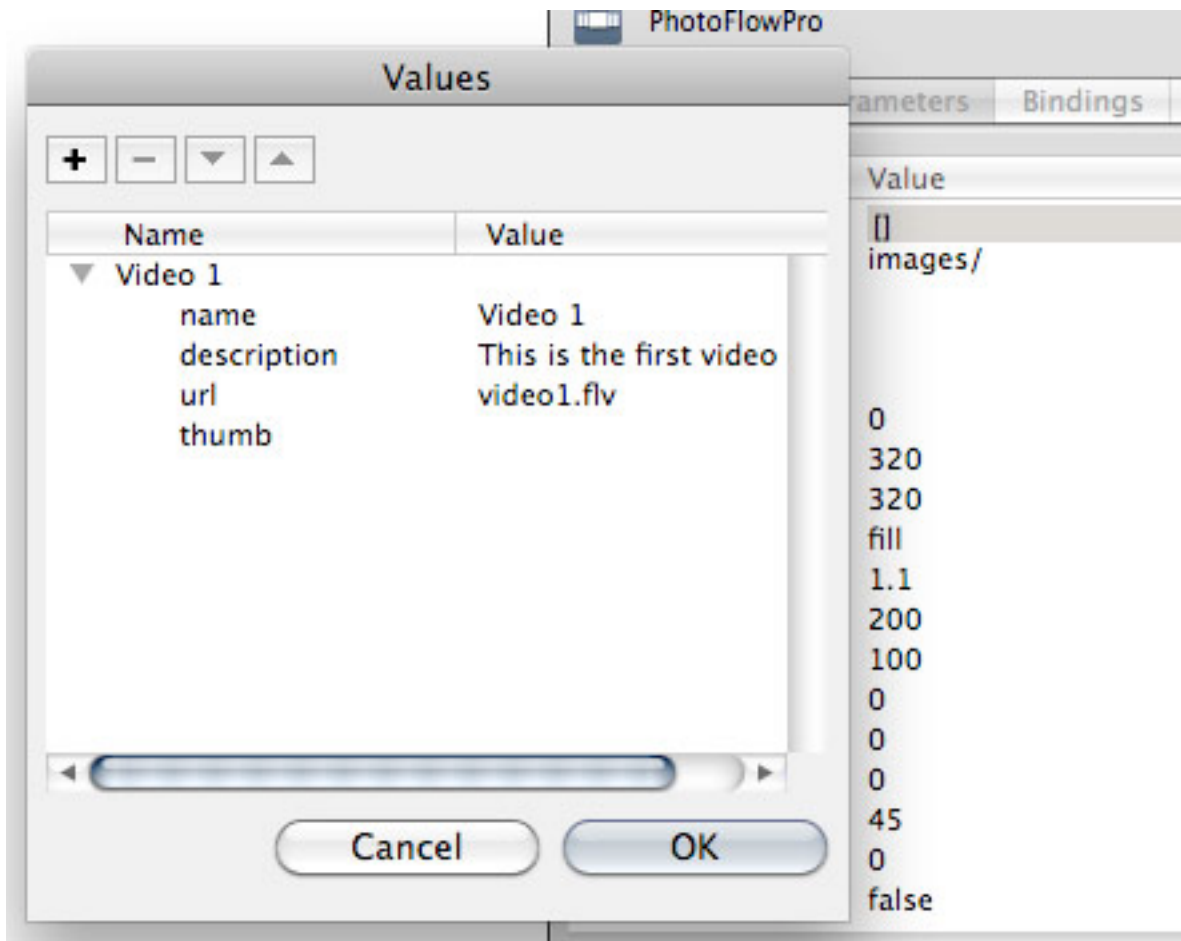


Fig. 15: Setting a video to play using parameters in Flash CS4.

## Adding Videos or SWFs via XML

Adding videos or SWFs via XML is also very simple. You just need to specify the file name of the video in the url attribute of the photo element.

*You will find working example files for this tutorial named 'working\_with\_video.fla' and 'working\_with\_video.xml' in the Examples folder that came with your download.*

1. Inside the folder you're using for testing create a folder named 'videos'.
2. Add three .flv files to the 'videos' folder and rename them 'video1.flv', 'video2.flv' and 'video3.flv'.
3. Open the 'using\_xml.fla' file that you created in the [Using XML section](#) of this user guide.

4. Re-save the file as 'working\_with\_video fla' (without the quotes) in the same directory as the earlier file.
5. Select the PhotoFlowPRO instance on the stage. Open the properties panel (if you're using CS5) or the component inspector panel (if you're using CS4 or CS3).
6. Edit the XML URL parameter so that it reads 'working\_with\_video.xml' (without the quotes).

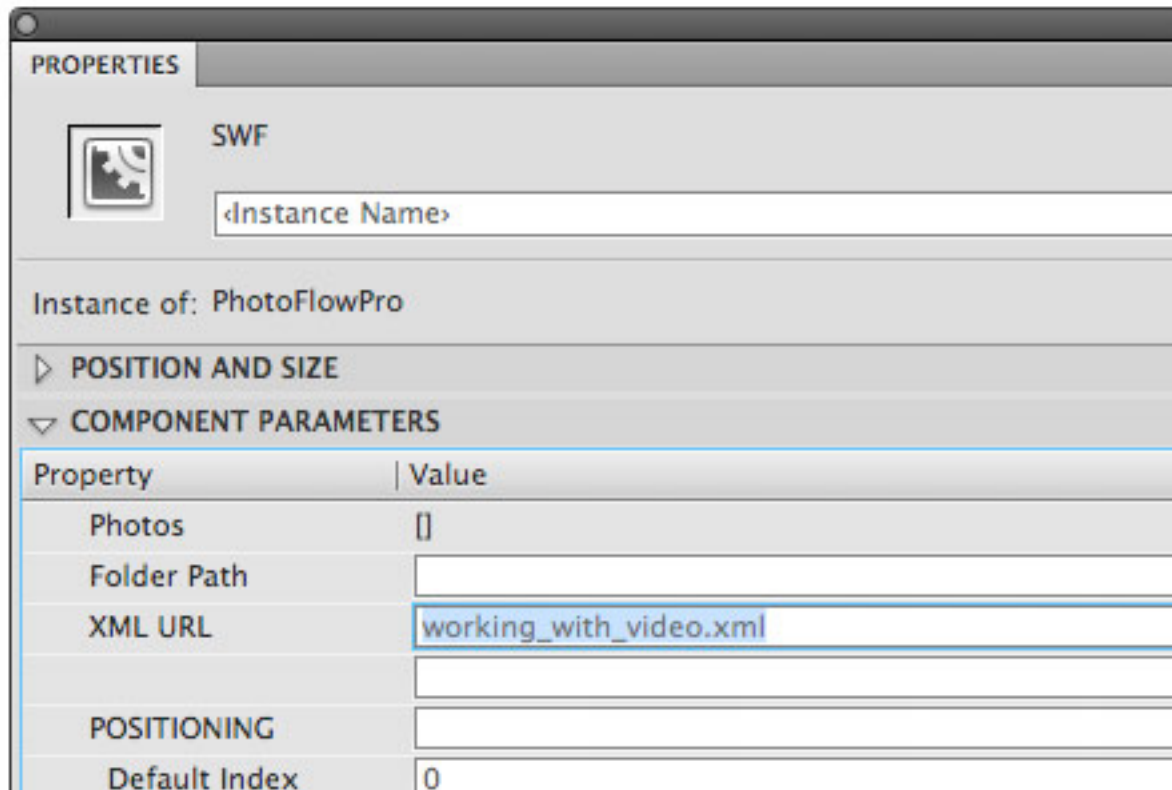


Fig. 16: Setting the XML URL parameter in Flash CS5.

7. Save the .fla.
8. Open the 'using\_xml.xml' file you created in the Using XML section using the application you created it in. Re-save it in the same directory as 'working\_with\_video.xml' (without the quotes).

9. Edit the 'working\_with\_video.xml' file so that it points at the FLV files in your 'videos' folder. It should look like this (the changes are highlighted in bold):

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="videos/">
  <photo name="Video 1" url="video1.flv">This is the 1st video</photo>
  <photo name="Video 2" url="video2.flv">This is the 2nd video</photo>
  <photo name="Video 3" url="video3.flv">This is the 3rd video</photo>
</photos>
```

10. Save the XML file.

11. Return to Flash and test the file (Ctrl+Enter on Windows or Cmnd+Enter on Mac).

## Adding Thumbnails to Videos or SWFs

You can add thumbnails to the component to act as a placeholder when an item is not selected. When a thumbnail is specified for a video it will be used instead of the first frame of the video until the item is selected. Once the item is selected the thumbnail will be removed. If the item is deselected the video will be paused and the thumbnail will not be replaced, the thumbnail will only be displayed again once the video has completed its full length.

In order to add a thumbnail to an item using the component's parameters follow these steps:

1. Prepare the images you wish to use as thumbnails. Save them in the same folder as your videos.
2. Open the file you created in the [Getting Started section](#) of this user guide.
3. Locate the Photos parameter (in CS5 it's in the properties panel, in CS4 and CS3 it's in the component inspector panel).
4. Open the values dialog box and in the 'thumb' fields enter the filename of your thumbnails.

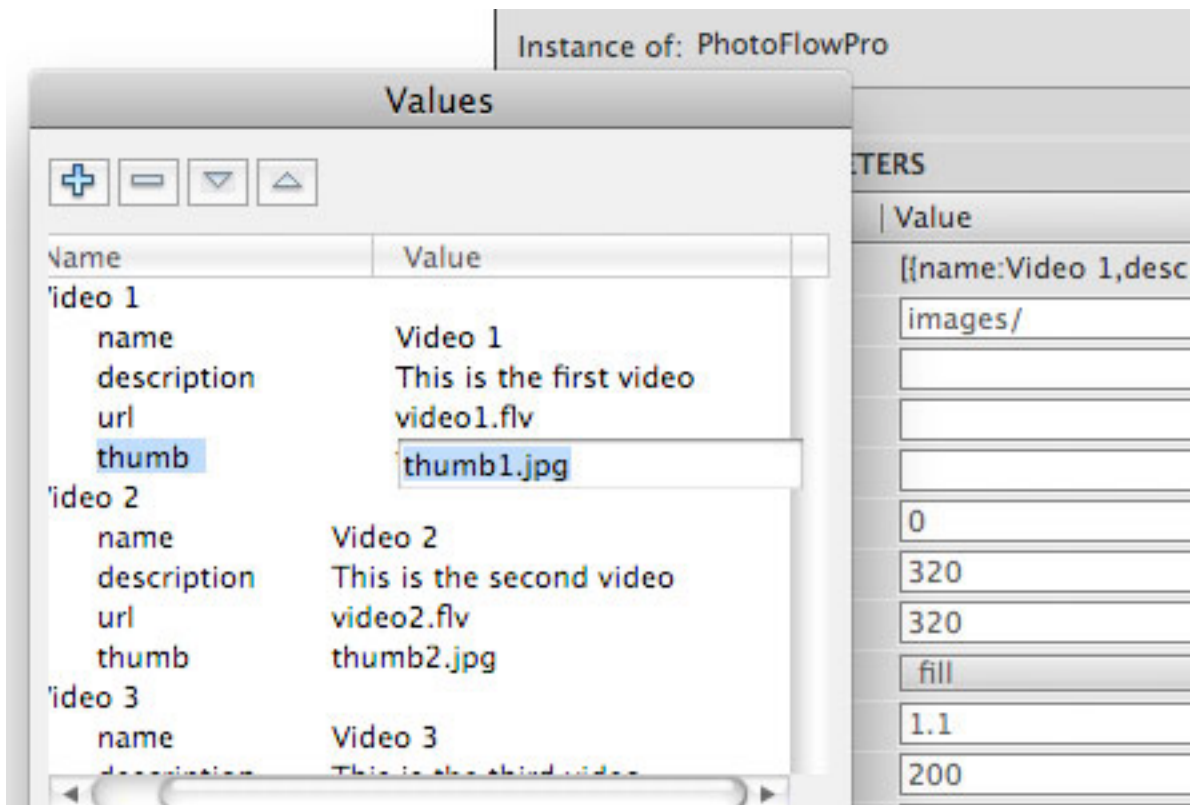


Fig. 17: Setting the thumb parameter in Flash CS5.

5. Click 'OK' and test your file.

If you are specifying your videos or SWFs in XML, once you have saved some images in the same folder as your flvs to use as thumbnails, you can add a thumb attribute to any photo element you wish to:

```
<?xml version="1.0" encoding="utf-8"?>

<photos path="videos/">

<photo name="Video 1" url="video1.flv" thumb="thumb1.jpg">This is the
1st video</photo>
<photo name="Video 2" url="video2.flv" thumb="thumb2.jpg">This is the
2nd video</photo>
<photo name="Video 3" url="video3.flv" thumb="thumb3.jpg">This is the
3rd video</photo>

</photos>
```

You will find working example files for this script named 'adding\_thumbnails fla' and 'adding\_thumbnails.xml' in the Examples folder that came with your download.

# Working with YouTube

The PhotoFlowPRO treats YouTube videos just like standard .flv files. All of the video playback methods work with YouTube.

*Note that due to the way YouTube delivers videos some of the playback methods such as `getMaxPosition()` may return 0 instead of the correct value until all of the video's meta data has been delivered. If one of these methods returns an incorrect value, keep polling the method until it delivers the anticipated value.*

## Finding the Correct address for a YouTube video

When viewing a video on YouTube the format of the URL is, for example, `http://www.youtube.com/watch?v=n6c0lqeAm0E` to use this URL in the PhotoFlowPRO you need to replace the 'watch?v=' with '/v/' so that it is written `http://www.youtube.com/v/n6c0lqeAm0E`.

## Specifying the YouTube video address

When specifying a YouTube video you can either use its full address, for example, "`http://www.youtube.com/v/3oFJDNxKucY`", or you can just use the video's ID "`3oFJDNxKucY`".

You can specify a YouTube video in the component's parameters.

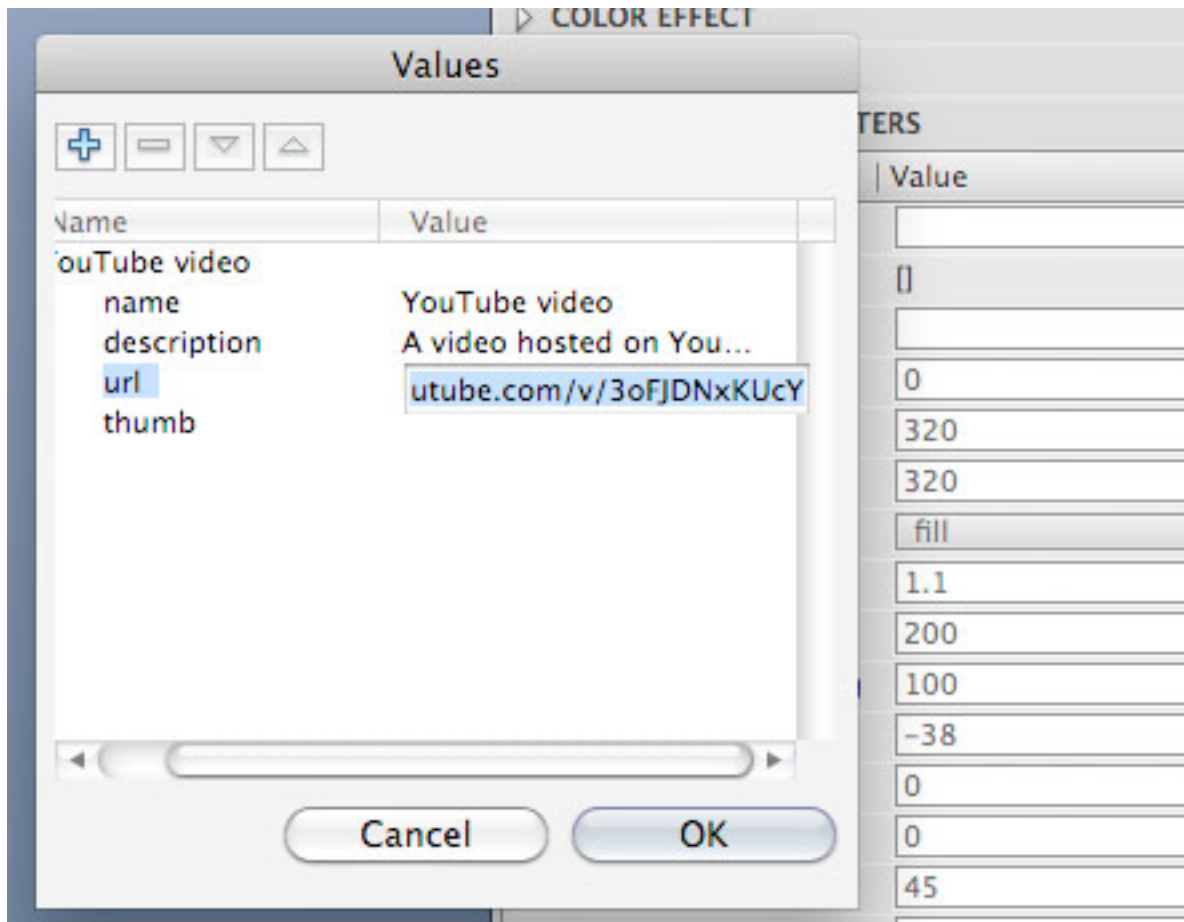


Fig. 20: Setting a YouTube video to play using parameters in Flash CS5.

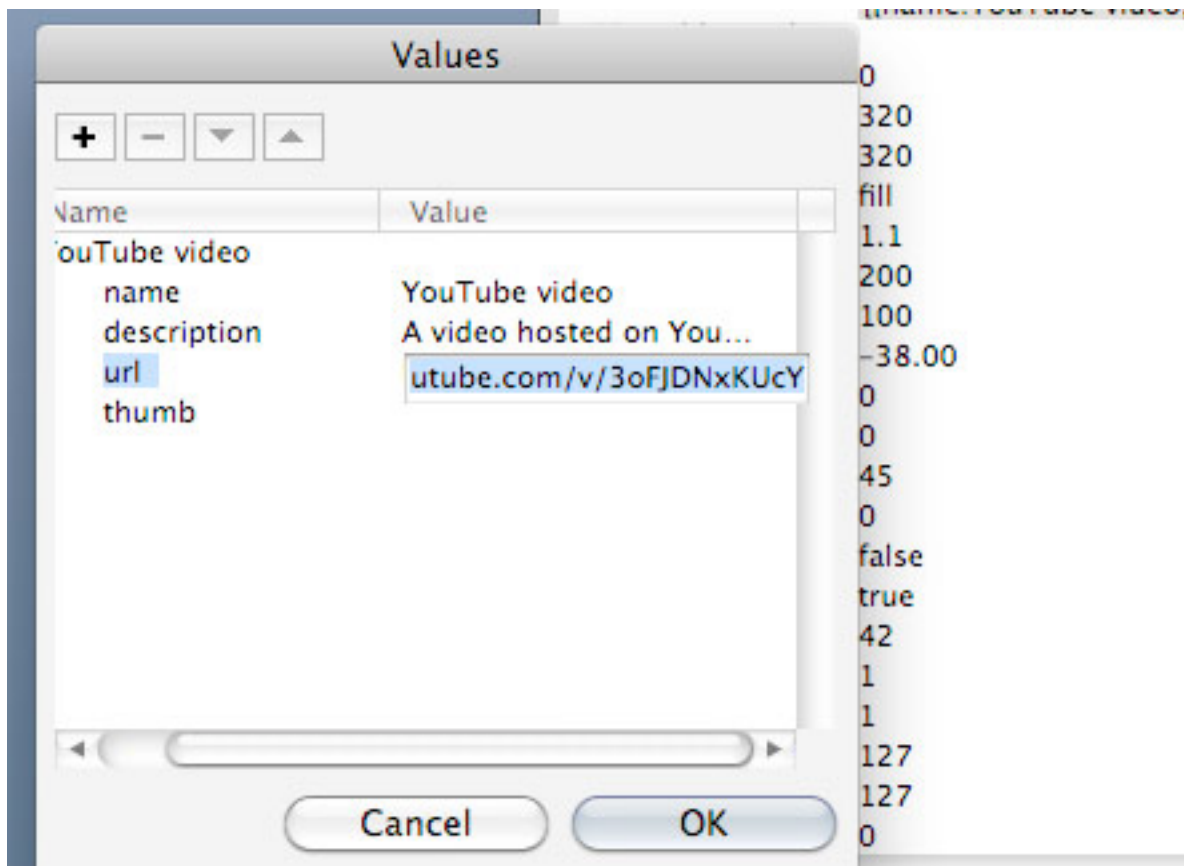


Fig. 21: Setting a YouTube video to play using parameters in Flash CS4.

You can also specify YouTube videos in your XML file like so:

```
<?xml version="1.0" encoding="utf-8"?>
<photos path="">
<photo name="YouTube Video" url="http://www.youtube.com/v/
3oFJDNxKUcY">This is a video hosted on YouTube</photo>
</photos>
```

# Working with Reflections

The PhotoFlowPRO provides the option to add reflections to your items to increase the sense of 3D space.

Reflections are enabled by default, you can turn them off by setting the 'Show Reflections' parameter to false in the component's parameters or the XML file or by setting the property showReflections to false.

## Reflection Distance

The 'Reflection Distance' parameter and reflectionDistance property specify the distance between the bottom of your item and the top of the reflection. This setting creates the illusion of the image, SWF or video floating above the imaginary plane.

## Reflection Alpha

The 'Reflection Alpha' parameter and the reflectionAlpha property specify the overall transparency of the reflection. The reflection is a gradient on the alpha channel from 0 to 1, consequently this setting can usually be treated as the maximum alpha displayed by the reflection.

It is possible to set a different transparency for the selected item using the 'Selected Reflection Alpha' parameter or selectedReflectionAlpha property.

## Reflection Depth

The 'Reflection Depth' parameter and reflectionDepth property specify the distance from the top of the reflection to the bottom. If your item is 100px high and your 'Reflection Depth' parameter is set to 50px then your reflection will be displaying 50% of your item.

It is possible to set a different depth for your selected item using the 'Selected Reflection Depth' parameter and the selectedReflectionDepth property.

*Note that reflection depth is not affected by the reflection distance parameter, the top of the reflection will always reflect the bottom of the item.*

## Reflection Refresh

If you are working with video, the most important reflection parameter is the 'Reflection Refresh' parameter or reflectionRefresh property; this setting determines how frequently the reflection is

redrawn by the PhotoFlowPRO.

When your item is an image the reflection refresh rate is automatically set to zero and the reflection will only redraw itself once; when the image has loaded.

When your item is a video or a SWF the reflection refresh setting causes the component to redraw the reflection for each item every 'n' milliseconds (where 'n' is the value of the reflection refresh setting).

- To update the reflection once every second set the 'Reflection Refresh' parameter to 1000.
- To update the reflection at the speed of video (which is 24 frames per second) set the 'Reflection Refresh' parameter to 42.
- To stop the reflection updating except when it first loads set the 'Reflection Refresh' rate to 0.
- For most purposes you should not need to update the reflection more than 12 times per second, which means setting the 'Reflection Refresh' parameter to 84.

The higher the number, the slower the rate of refresh and the less system resources used by Flash. Slower machines will start to experience lag when the 'Reflection Refresh' setting is set to a low value, this will be exacerbated by the use of videos with large dimensions.

*We strongly suggest that you set the 'Reflection Refresh' parameter to as high a number as quality will allow in order to ensure the file will run smoothly on less powerful machines.*

# Working with Names & Descriptions

The PhotoFlowPRO provides two parameters to allow you to display the name and description you've set up for your items. When the selected item is changed the object specified by the 'Name Field' and 'Description Field' parameters (or the nameField and descriptionField properties) will have their content updated automatically.

*Although the most common usage of the name and description fields is to update a TextField instance the parameters will actually accept any object that exposes a 'text' property.*

To add the fields using the component's parameters follow these steps:

*You will find working example file for this section named 'working\_with\_names\_and\_descriptions.fla' in the Examples folder that came with your download.*

1. Open the file you created in the [Getting Started section](#) of this user guide.
2. Create a new layer on the timeline.
3. From the tools palette select the Text tool.
4. Add a text field to the new layer.
5. With the text field still selected open the properties panel (Ctrl+F3 on Windows or Cmnd+F3 on Mac) and ensure that the text is dynamic. If you wish to embed the font you can do so.

*Note that if you're using CS5 and publishing for Flash Player 10 or later you can use TLF Text rather than Classic text if you prefer.*

6. Give the text field the instance name 'nameField' (without the quotes).

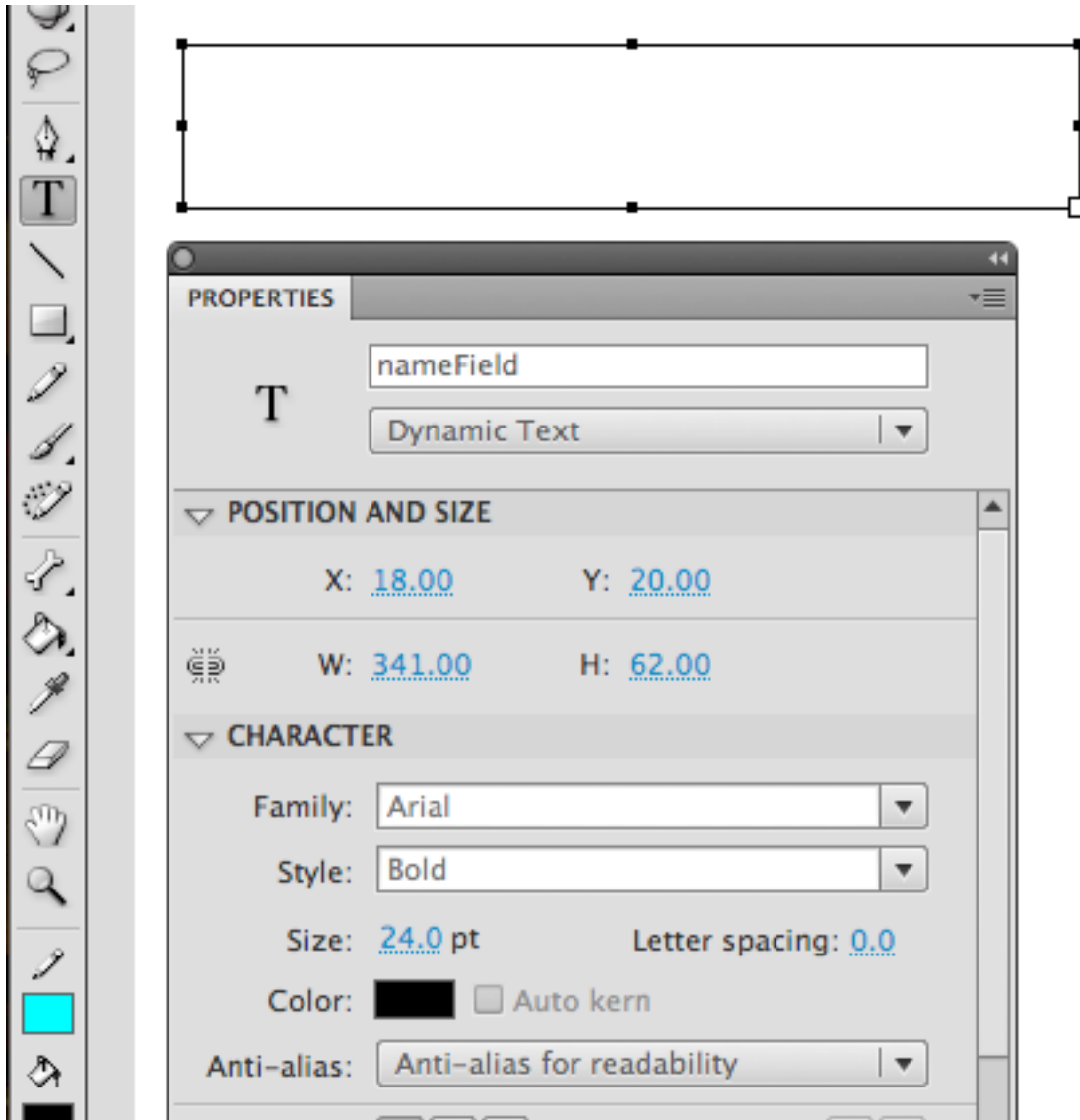


Fig. 22: Setting up the nameField instance in Flash CS4.

7. Repeat the last three steps adding a second text field and giving it the instance name 'descriptionField' (without the quotes).

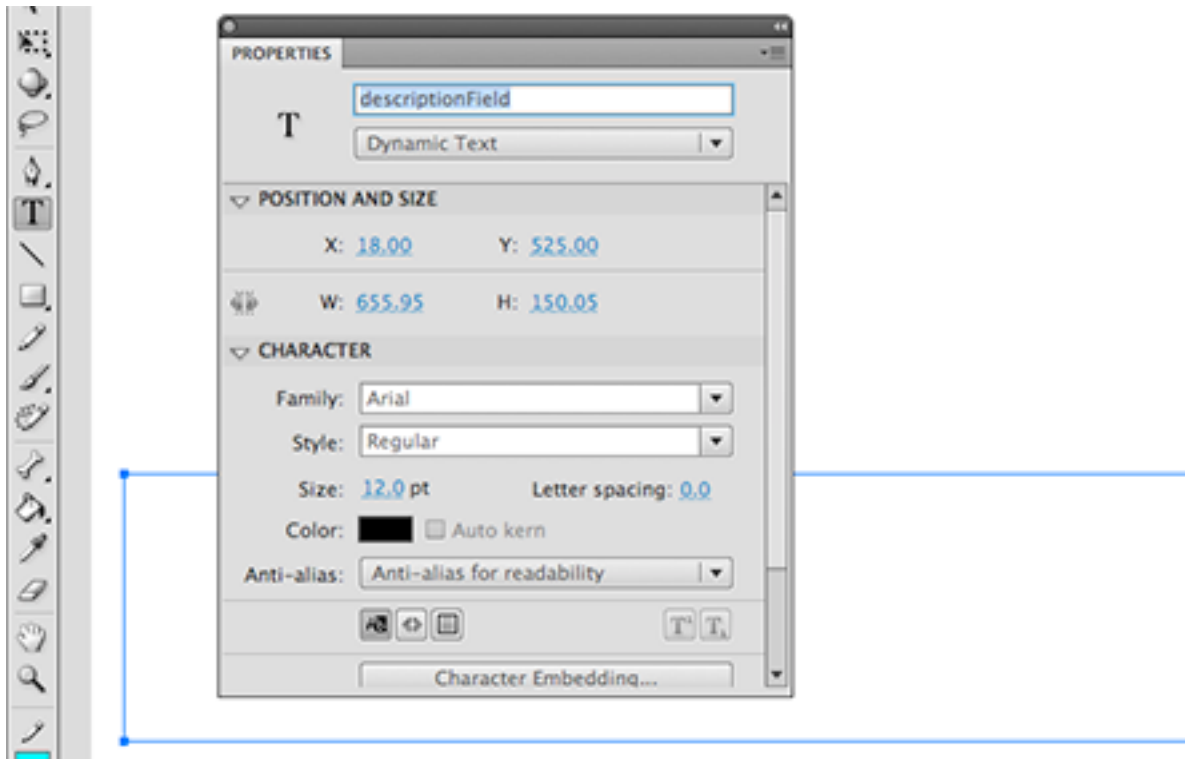


Fig. 23: Setting up the descriptionField instance in Flash CS4.

8. Select the PhotoFlowPRO component on the stage.
9. Locate the 'Name Field' parameter (If you're using Flash CS5 it's in the properties panel, if you're using CS4 or CS3 it's in the component inspector panel).
10. Enter 'nameField' (without the quotes).
11. Locate the 'Description Field' parameter.
12. Enter 'descriptionField' (without the quotes).
13. Test your file and you'll see the name and description update in the fields each time the selected item changes.

# Working with Scrollbars

A scrollbar can be set up to allow users to scroll through the items in your component at speed. You can specify a scrollbar for the component with the 'Scrollbar' parameter or using the scrollbar property with Actionscript.

*Note that when setting the 'Scrollbar' parameter you must use the scrollbar's instance name whereas when using the scrollbar property you must use a reference to the object.*

## Using the PhotoFlowProScrollBar component

The PhotoFlowPRO comes with a scrollbar component designed to work with the scrollbar parameter. To set it up, follow these steps:

1. Open the file you started in the Getting Started section of this user guide.
2. Open the components panel (Ctrl+F7 on Windows Cmnd+F7 on Mac).
3. Locate the 'PhotoFlowProScrollBar' component inside the 'photoFlowPRO' folder and drag it onto the stage.
4. With the component still selected open the properties panel (in CS5) or the component inspector panel (in CS4 or CS3) and locate the 'distance' parameter. This value determines the distance in pixels between the bottom of your items (ignoring any reflection) and the top of the scrollbar. Set this value to 200.
5. With the component still selected give it the instance name 'myScroller' (without the quotes).
6. Select the PhotoFlowPRO instance on the stage. Open the properties panel (in CS5) or the component inspector panel (in CS4 or CS3) and locate the 'Scrollbar' parameter, enter the instance name you gave the PhotoFlowProScrollBar component in the last step (myScroller).
7. Save your file and test it. Drag the scrollbar from side to side to scroll thru your images.

## Using other components

The scrollbar parameter will accept any class that meets the following requirements:

- The scrollbar must extend the DisplayObject class.
- The scrollbar must have a 'maximum', 'maxScrollPosition' or 'maxScroll' property.
- The scrollbar must have a 'minimum', 'minScrollPosition' or 'minScroll' property.

- The scrollbar must have a 'value', 'scrollPosition' or 'position' property.

One example of a scrollbar that meets these requirements is the `fl.controls.Slider` component that comes bundled with Flash, in this tutorial we'll set an instance of the Slider up to control the PhotoFlowPRO component.

*You will find working example file for this section named 'working\_with\_scrollbars.fla' in the Examples folder that came with your download.*

1. Open the file you created in the [Getting Started section](#) of this user guide.
2. Select the instance of the PhotoFlowPRO, open the properties panel (Ctrl+F3 on Windows or Cmnd+F3 on Mac) and give it the instance name 'myPhotoFlowPro' (without the quotes).
3. Open the components panel (Ctrl+F7 on Windows Cmnd+F7 on Mac).
4. Locate the 'User Interface' folder and inside it the 'Slider' component.
5. Drag a copy of the Slider onto your stage and then delete it (this is to add the component to your library).
6. Create a new layer on your timeline, select the keyframe, open the actions panel (Alt+F9) and then enter the following Actionscript:

```
// import classes
import fl.controls.Slider;
```

This will import the Slider class so you can refer to it in Actionscript. Next add:

```
// create a new Slider instance
var mySlider = new Slider();
```

This creates a new instance of the Slider class. Next add:

```
// position the slider
mySlider.x = (stage.stageWidth - mySlider.width) / 2;
mySlider.y = myPhotoFlowPro.y + myPhotoFlowPro.height - 50;
```

This first line positions the Slider instance horizontally in the middle of the stage and the second line positions the Slider instance vertically 50 pixels above the bottom of the PhotoFlowPRO component. Now add this bold line:

```
// add the Slider to the stage
addChild(mySlider);
```

This adds the Slider instance to the stage. Finally add this bold line:

```
// set the Slider instance as the PhotoFlowPRO's scrollbar  
myPhotoFlowPro.scrollbar = mySlider;
```

This sets the Slider instance as the PhotoFlowPRO instance's scrollbar. The PhotoFlowPRO will now take care of all the interaction required to respond to input made with the Slider and when the PhotoFlowPRO is updated in some other way it will update the Slider accordingly.

7. Save your file and test it. When you drag the Slider from side to side the PhotoFlowPRO will respond, if you click one of the non-selected items the Slider will update accordingly.

# Skinning the Components

1. To skin the PhotoFlowPRO or the PhotoFlowProScrollBar double-click on the instance on the stage or the symbol in the library.
2. Select the skin you wish to edit and double-click it to access its timeline.
3. Make the changes you wish to make to the graphics of the skin.
4. Save and test your file, you will see the changes you made reflected in the component's appearance.

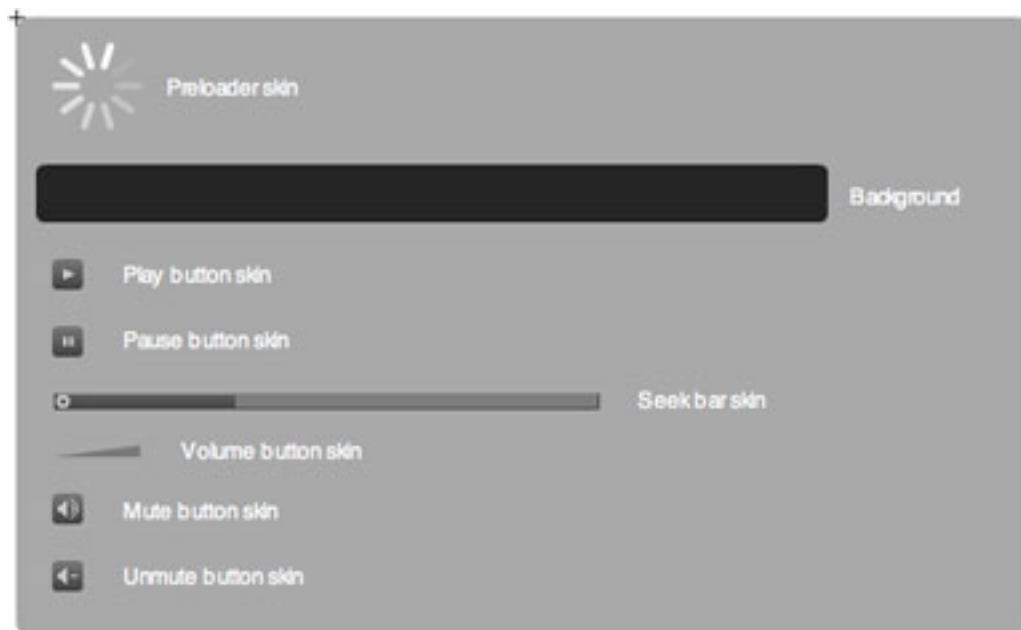


Fig. 24: The skins for the PhotoFlowPRO.



Fig. 25: The skins for the PhotoFlowPROScrollBar.

# ActionScript events

The PhotoFlowPRO includes two kinds of event, ***com.flashloaded.photoFlowPro.PhotoFlowProEvent*** and ***com.flashloaded.photoFlowPro.PhotoFlowProItemEvent***.

## PhotoFlowProEvent

The PhotoFlowProEvent class describes events that relate to the component as a whole.

### PhotoFlowProEvent.INIT:String

#### Availability

Flash Player 9

#### Description

Event; dispatched when the PhotoFlowPRO instance finishes initializing.

#### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProEvent;

myPhotoFlowPro.addEventListener(PhotoFlowProEvent.INIT, initHandler);

function initHandler(e:PhotoFlowProEvent):void {
    trace(e.target.name + " initialized");
}
```

### PhotoFlowProEvent.REMOVE:String

#### Availability

Flash Player 9

#### Description

Event; dispatched when an item is removed from the PhotoFlowPRO instance.

Note that although the event is triggered by the removal of a single item it does not contain a reference to the item being removed, this is because the event is dispatched after removal and there is no longer an item to reference. To determine which item has been removed, retain the index that is passed to the `removeImageAt()` method.

## Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProEvent.REMOVE,
    removeHandler);

function removeHandler(e:PhotoFlowProEvent):void {
    trace(e.target.name + " removed item " + itemToRemove);
}

var itemToRemove:uint = 2;
myPhotoFlowPro.removeImageAt(itemToRemove);
```

## PhotoFlowProItemEvent

The PhotoFlowProItemEvent class describes events that relate to individual items within the component instance.

PhotoFlowProItemEvent events include an 'index' property which specifies which item in the component the event relates to.

## PhotoFlowProItemEvent.ADD:String

### Availability

Flash Player 9

### Description

Event; dispatched when an item is added to the PhotoFlowPro instance.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProItemEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProItemEvent.ADD,
    addHandler);

function addHandler(e:PhotoFlowProItemEvent):void {
    trace("Item added to " + e.target.name + " at " + e.index);
}
```

## PhotoFlowProItemEvent.CLICK\_SELECTED:String

### Availability

Flash Player 9

### Description

Event; dispatched when the mouse button is pressed and released while the cursor is over the selected item.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProItemEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProItemEvent.CLICK_SELECTED,
    clickSelectedHandler);

function clickSelectedHandler(e:PhotoFlowProItemEvent):void {
    trace("Selected item (" e.index + ") clicked");
}
```

## PhotoFlowProItemEvent.PHOTO\_LOADED:String

### Availability

Flash Player 9

### Description

Event; dispatched when an item finishes loading.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProItemEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProItemEvent.PHOTO_LOADED,
    photoLoadedHandler);

function photoLoadedHandler(e:PhotoFlowProItemEvent):void {
    trace("Item (" e.index + ") loaded in " + e.target.name);
}
```

## PhotoFlowProItemEvent.PHOTO\_MOUSE\_OUT:String

### Availability

Flash Player 9

### Description

Event; dispatched when the cursor rolls out of the area occupied by the dispatching item.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProItemEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProItemEvent.PHOTO_MOUSE_OUT,
    mouseOutHandler);

function mouseOutHandler(e:PhotoFlowProItemEvent):void {
    trace("Mouse left item " e.index);
}
```

## PhotoFlowProItemEvent.PHOTO\_MOUSE\_OVER:String

### Availability

Flash Player 9

### Description

Event; dispatched when the cursor rolls over an item.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProItemEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProItemEvent.PHOTO_MOUSE_OVER,
    mouseOverHandler);

function mouseOverHandler(e:PhotoFlowProItemEvent):void {
    trace("Mouse rolled over item " e.index);
}
```

## PhotoFlowProItemEvent.SELECT:String

### Availability

Flash Player 9

### Description

Event; dispatched when an item is selected.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProItemEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProItemEvent.SELECT,
    selectHandler);

function selectHandler(e:PhotoFlowProItemEvent):void {
    trace("Item " + e.index + " selected");
}
```

# ActionScript properties

**autoFlip**:Boolean

## Availability

Flash Player 9

## Description

Property; Determines whether the component should animate to the next item automatically (true) or not (false). The speed at which any animation takes place is determined by the autoFlipDelay property.

## Example

```
myPhotoFlowPro.autoFlip = true;
```

**autoFlipDelay**:Number

## Availability

Flash Player 9

## Description

Property; Specifies the number of seconds delay before automatically animating to the next item. If autoFlip is set to false this property is ignored.

## Example

```
myPhotoFlowPro.autoFlip = true;  
myPhotoFlowPro.autoFlipDelay = 3.5;
```

**cameraHeight**:Number

## Availability

Flash Player 9

## Description

Property; Sets the height of the 3D point of view

## Example

```
myPhotoFlowPro.cameraHeight = 50;
```

## descriptionField:String

### Availability

Flash Player 9

### Description

Property; specifies a text field or component to display the currently selected item's description.

### Example

```
// import TextField class
import flash.display.TextField;

// create TextField instance
var myDescriptionText:TextField = new TextField();

// position TextField instance and add to the stage
myDescriptionText.x = 200;
myDescriptionText.y = 500;
addChild(myDescriptionText)

// set TextField instance as the PhotoFlowPro's descriptionField
myPhotoFlowPro.descriptionField = myDescriptionText;
```

## distanceToCamera:Number

### Availability

Flash Player 9

### Description

Property; specifies the distance to position the items from the screen in the 3D environment. This value effective creates a scaling effect.

### Example

```
myPhotoFlowPro.distanceToCamera = 100;
```

## flipSoundClass:String

### Availability

Flash Player 9

### Description

Property; specifies a sound from the library to be played when the component animates to the next item.

### Example

```
myPhotoFlowPro.flipSoundClass = myLibrarySound;
```

## folderPath:String

### Availability

Flash Player 9

### Description

Property; specifies a path to the images, swfs and videos loaded by the component. When items are specified using XML this property is overridden.

### Example

```
myPhotoFlowPro.folderPath = "images/";
```

## holderAlpha:Number

### Availability

Flash Player 9

### Description

Property; specifies the transparency of each item's background.

### Example

```
myPhotoFlowPro.holderAlpha = 0.8;
```

## holderBorder:Number

### Availability

Flash Player 9

### Description

Property; specifies the thickness of the border around each item's background.

### Example

```
myPhotoFlowPro.holderBorder = 4;
```

## holderBorderAlpha:Number

### Availability

Flash Player 9

### Description

Property; specifies the transparency of the border around each item's background.

### Example

```
myPhotoFlowPro.holderBorderAlpha = 0.5;
```

## holderBorderColor:uint

### Availability

Flash Player 9

### Description

Property; specifies the color of the border around each item's background.

### Example

```
myPhotoFlowPro.holderBorderColor = 0xCCCCCC;
```

## holderColor:uint

### Availability

Flash Player 9

### Description

Property; specifies the color of each item's background.

### Example

```
myPhotoFlowPro.holderColor = 0xDEDEDE;
```

## index:uint

### Availability

Flash Player 9

### Description

Property; sets the currently selected item in the component. If the component has not yet initialized, this property sets the default selected item.

### Example

```
myPhotoFlowPro.index = 4;
```

## itemAngle:Number

### Availability

Flash Player 9

### Description

Property; the angle around the yAxis at which to display the items when not selected.

### Example

```
myPhotoFlowPro.itemAngle = 56;
```

**itemHeight**:Number

**Availability**

Flash Player 9

**Description**

Property; the height of each item.

**Example**

```
myPhotoFlowPro.itemHeight = 180;
```

**itemWidth**:Number

**Availability**

Flash Player 9

**Description**

Property; the width of each item.

**Example**

```
myPhotoFlowPro.itemWidth = 180;
```

**nameField**:String

**Availability**

Flash Player 9

**Description**

Property; specifies a text field or component to display the currently selected item's name.

**Example**

```
// import TextField class
import flash.display.TextField;

// create TextField instance
var myNameText:TextField = new TextField();

// position TextField instance and add to the stage
myNameText.x = 200;
myNameText.y = 500;
addChild(myNameText)
```

```
// set TextField instance as the PhotoFlowPro's nameField  
myPhotoFlowPro.nameField = myNameText;
```

## reflectionAlpha:Number

### Availability

Flash Player 9

### Description

Property; specifies the transparency of each non-selected item's reflection. If the showReflection property is set to false this property is ignored.

### Example

```
myPhotoFlowPro.reflectionAlpha = 0.5;
```

## reflectionDepth:Number

### Availability

Flash Player 9

### Description

Property; specifies the depth of each non-selected item's reflection. The depth is the distance on the y axis measured from the top of the reflection when the item is full on, if the plane is distorted by another setting this value may be foreshortened. If the showReflection property is set to false this property is ignored.

### Example

```
myPhotoFlowPro.reflectionDepth = 127;
```

## reflectionDistance:Number

### Availability

Flash Player 9

### Description

Property; the gap between the bottom of the item and the top of its reflection. If the showReflection property is set to false this property is ignored.

### Example

```
myPhotoFlowPro.reflectionDistance = 20;
```

## reflectionRefresh:uint

### Availability

Flash Player 9

### Description

Property; the frequency, in milliseconds, at which to redraw the reflection. If the item is an image this property will be ignored. If the showReflection property is set to false this property is ignored.

### Example

```
myPhotoFlowPro.reflectionRefresh = 84;
```

## scaleMode:String

### Availability

Flash Player 9

### Description

Property; the method of scaling to apply to the items loaded.

Valid values are PhotoFlowProScaleMode.FILL, which scales the item to match its itemWidth and itemHeight properties but maintains the original proportions and crops if required; PhotoFlowProScaleMode.FIT, which maintains the proportions and scales the item to match the itemWidth and itemHeight but without cropping and PhotoFlowPro.NO\_SCALE, which does not adjust the scale of the item.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProScaleMode;  
myPhotoFlowPro.scaleMode = PhotoFlowProScaleMode.NO_SCALE;
```

## scrollbar.\*

### Availability

Flash Player 9

### Description

Property; specifies a scrollbar to scroll the items back and forth.

### Example

```
// import Slider class
import fl.controls.Slider;

// create Slider instance
var mySlider:Slider = new Slider();

// position Slider instance and add to the stage
mySlider.x = (stage.stageWidth - mySlider.width) / 2;
mySlider.y = myPhotoFlowPro.y + myPhotoFlowPro.height + 20;
addChild(mySlider)

// set Slider instance as the PhotoFlowPro instance's scrollbar
myPhotoFlowPro.scrollbar = mySlider;
```

## selectedReflectionAlpha:Number

### Availability

Flash Player 9

### Description

Property; set the transparency of the selected item's reflection. If the showReflection property is set to false this property is ignored.

### Example

```
myPhotoFlowPro.selectedReflectionAlpha = 1;
```

## selectedReflectionDepth:Number

### Availability

Flash Player 9

### Description

Property; sets the depth of the selected item's reflection. The depth is the distance on the y axis measured from the top of the reflection when the item is full on, if the plane is distorted by another setting this value may be foreshortened. If the showReflection property is set to false this property is ignored.

### Example

```
myPhotoFlowPro.selectedReflectionDepth = 200;
```

## selectedSpacing:Number

### Availability

Flash Player 9

### Description

Property; the additional spacing to be applied between the selected and non-selected items.

### Example

```
myPhotoFlowPro.selectedSpacing = 40;
```

## selectedScale:Number

### Availability

Flash Player 9

### Description

Property; the scaling to be applied to the selected item.

### Example

```
myPhotoFlowPro.selectedScale = 1.5;
```

**selectedView**:Number

**Availability**

Flash Player 9

**Description**

Property; specifies the vertical point of view to apply to the selected image, swf or video.

**Example**

```
myPhotoFlowPro.selectedView = 0;
```

**selectedY**:Number

**Availability**

Flash Player 9

**Description**

Property; the offset on the y axis to be applied to the selected item. Positive values drop the selected item below the non-selected items and negative values raise it above.

**Example**

```
myPhotoFlowPro.selectedY = -50;
```

**showReflection**:Boolean

**Availability**

Flash Player 9

**Description**

Property; specifies whether to display the item reflections (true) or not (false).

**Example**

```
myPhotoFlowPro.showReflection = true;
```

## slideDirection:String

### Availability

Flash Player 9

### Description

Property; the direction in which to autoFlip. Once the direction reverses this property updates itself.

Valid values are PhotoFlowPro.SLIDE\_LEFT and PhotoFlowPro.SLIDE\_RIGHT.

### Example

```
myPhotoFlowPro.slideDirection = PhotoFlowPro.SLIDE_RIGHT;
```

## slideEasing:String

### Availability

Flash Player 9

### Description

Property; the kind of easing to apply to the slide animation.

Valid values are contained in the fl.motion.easing package. Refer to [Adobe documentation](#) for more information.

### Example

```
import fl.motion.easing.*;
myPhotoFlowPro.slideEasing = Sine.easeInOut;
```

## slideTime:Number

### Availability

Flash Player 9

### Description

Property; the time taken to animate from one item to the next, specified as seconds.

### Example

```
myPhotoFlowPro.slideTime = 0.5;
```

**smoothing**:Boolean

**Availability**

Flash Player 9

**Description**

Property; specifies whether to apply smoothing to the images and videos loaded (true) or not (false). If the item is a SWF this property is ignored.

**Example**

```
myPhotoFlowPro.smoothing = true;
```

**spacing**:Number

**Availability**

Flash Player 9

**Description**

Property; the amount of spacing between items, specified in pixels.

**Example**

```
myPhotoFlowPro.spacing = 24;
```

**totalPhotos**:uint

**Availability**

Flash Player 9

**Description**

Property; this property is read only. Returns the total number of items loaded into the component.

**Example**

```
trace("There are " + myPhotoFlowPro.totalPhotos + " items");
```

## useKeyboard: Boolean

### Availability

Flash Player 9

### Description

Property; specifies whether the keyboard can be used to control the component. If set to true the left and right arrow keys move the items left and right and the up and down arrow keys increase and decrease the distanceToCamera property; if set to false the key have no affect.

### Example

```
myPhotoFlowPro.useKeyboard = true;
```

## useMouseView: Boolean

### Availability

Flash Player 9

### Description

Property; specifies whether the 3D environment should tilt to follow the point of view of the mouse cursor (true) or not (false).

### Example

```
myPhotoFlowPro.useKeyboard = true;
```

## useMouseWheel: Boolean

### Availability

Flash Player 9

### Description

Property; specifies whether turning the mouse wheel navigates to the next item (true) or not (false).

### Example

```
myPhotoFlowPro.useMouseWheel = true;
```

**view**:Number

**Availability**

Flash Player 9

**Description**

Property; specifies the vertical point of view to apply to the whole component.

**Example**

```
myPhotoFlowPro.view = 12;
```

**videoAlwaysRestarts**:Boolean

**Availability**

Flash Player 9

**Description**

Property; specifies whether videos loaded by the component should loop (true) or not (false).

**Example**

```
myPhotoFlowPro.videoAlwaysRestarts = true;
```

**xmlURL**:String

**Availability**

Flash Player 9

**Description**

Property; specifies the location of the XML file used to define the component.

**Example**

```
myPhotoFlowPro.xmlURL = "images/flow.xml";
```

# ActionScript methods

## addItem()

### Parameters

item:XML

### Availability

Flash Player 9

### Description

Method; adds an item defined as XML to the component.

### Example

```
var item:XML = <photo name="Item" url="image.jpg">Description</photo>;  
myPhotoFlowPro.addItem(item);
```

## addItemAsObject()

### Parameters

item:Object

index:int

### Availability

Flash Player 9

### Description

Method; adds an item to the component defined as an object at the specified index.

### Example

```
var item:Object = new Object();  
item.name = "Item";  
item.url = "image.jpg";  
item.description = "Description"  
myPhotoFlowPro.addItemAsObject(item, 2);
```

## addItemAt()

### Parameters

item:XML  
index:int

### Availability

Flash Player 9

### Description

Method; adds an item defined as XML to the component at the specified index

### Example

```
var item:XML = <photo name="Item" url="image.jpg">Description</photo>;  
myPhotoFlowPro.addItemAt(item, 2);
```

## addItemAsObjects()

### Parameters

items:Array

### Availability

Flash Player 9

### Description

Method; adds each item, which is defined as an object, in the supplied array to the component.

### Example

```
var item1:Object = {name="Item 1", url="Image 1"};  
var item2:Object = {name="Item 2", url="Image 2"};  
var item3:Object = {name="Item 3", url="Image 3"};  
  
var items:Array = [item1, item2, item3];  
  
myPhotoFlowPro.addItemAsObjects(items);
```

## getPosition()

### Returns

Number

### Availability

Flash Player 9

### Description

Method; if the currently selected item is a SWF, this method returns the current frame, if the currently selected item is a video this method returns the number of seconds that have passed. If the current item is an image this method returns -1.

### Example

```
import flash.events.Event;
import flash.text.TextField;

var frameField:TextField = new TextField();
addChild(frameField);

addEventListener(Event.ENTER_FRAME, frameHandler);
function frameHandler(e:Event):void
{
    frameField.text = "Position = " + myPhotoFlowPro.getPosition();
}
```

## getVolume()

### Returns

Number

### Availability

Flash Player 9

### Description

Method; if the currently selected item is a video this method returns the current volume of the video, otherwise this method returns -1.

### Example

```
trace(myPhotoFlowPro.getVolume());
```

## kill()

### Availability

Flash Player 9

### Description

Method; this function removes the internal event listeners. This method should be called before removing the component to ensure there is no unnecessary memory usage.

### Example

```
myPhotoFlowPro.kill();  
myPhotoFlowPro.parent.removeChild(myPhotoFlowPro);
```

## moveIndex()

### Parameters

change:int

### Returns

uint

### Availability

Flash Player 9

### Description

Method; moves the current index by the specified number. A Positive number slides items to the left and a negative number slides items to the right. This method returns the resulting index.

### Example

```
var newIndex:uint = myPhotoFlowPro.moveIndex(-2);  
if(newIndex == myPhotoFlowPro.totalPhotos / 2)  
{  
    trace("Component has returned to the center");  
}
```

## mute()

### Returns

Boolean

### Availability

Flash Player 9

### Description

Method; if the currently selected item is a video this method silences the video and returns true, otherwise it returns false.

### Example

```
if(myPhotoFlowPro.mute())
{
    trace("Video is muted");
}
```

## next()

### Availability

Flash Player 9

### Description

Method; moves the component to the next item.

### Example

```
import fl.controls.Button;
import flash.events.MouseEvent;

var nextButton:Button = new Button();
nextButton.label = "Next";
addChild(nextButton);
nextButton.addEventListener(MouseEvent.CLICK, nextHandler);

function nextHandler(e:Event):void
{
    myPhotoFlowPro.next();
}
```

## pause()

### Returns

Boolean

### Availability

Flash Player 9

### Description

Method; if the currently selected item is a SWF this method stops the timeline playing and returns true, if the currently selected item is a video this method pauses the play back. Otherwise the method returns false.

### Example

```
import com.flashloaded.photoFlowPro.PhotoFlowProItemEvent;

myPhotoFlowPro.addEventListener(
    PhotoFlowProItemEvent.SELECT,
    selectHandler);

function selectHandler(e:PhotoFlowProItemEvent):void {
    myPhotoFlowPro.pause();
};
```

## play()

### Returns

Boolean

### Availability

Flash Player 9

### Description

Method; if the currently selected item is a SWF or video this method begins play back and returns true, otherwise it returns false.

### Example

```
import fl.controls.Button;
import flash.events.MouseEvent;

var playButton:Button = new Button();
playButton.label = "Play";
addChild(playButton);
playButton.addEventListener(MouseEvent.CLICK, playHandler);
```

```
function playHandler(e:Event):void
{
    if(myPhotoFlowPro.play()){
        trace("Current item is playing");
    }
    else
    {
        trace("Could not play the current item");
    }
}
```

## previous()

### Availability

Flash Player 9

### Description

Method; moves the component to the previous item.

### Example

```
myPhotoFlowPro.previous();
```

## removeAll()

### Availability

Flash Player 9

### Description

Method; removes all items from the component.

### Example

```
myPhotoFlowPro.removeAll();
```

## removeItemAt()

### Parameters

index:uint

### Availability

Flash Player 9

### Description

Method; removes the item at the specified index.

### Example

```
myPhotoFlowPro.removeItemAt(2);
```

## setPosition()

### Parameters

position

### Returns

Boolean

### Availability

Flash Player 9

### Description

Method; if the currently selected item is a SWF, this method sets the frame to the specified position, if the currently selected item is a video this method sets the time of the video to the specified position measured in seconds, otherwise it returns false.

### Example

```
myPhotoFlowPro.setPosition(12);
```

## setVolume()

### Parameters

volume:Number

### Returns

Boolean

### Availability

Flash Player 9

### Description

Method; if the currently selected item is a video this method sets its volume to the specified value and returns true, otherwise it returns false.

### Example

```
if(myPhotoFlowPro.setVolume(0))
{
    trace("Volume has been set to 0");
}
else
{
    trace("Volume could not set");
}
```

## unMute()

### Returns

Boolean

### Availability

Flash Player 9

### Description

Method; if the current item is a video the method undoes any mute() method that has been applied, otherwise the method returns false.

### Example

```
import fl.controls.Button;
import flash.events.MouseEvent;

if(myPhotoFlowPro.mute())
{
    var unMuteBtn:Button = new Button();
    unMuteBtn.label = "Unmute";
```

```
        addChild(unMuteBtn);
        unMuteBtn.addEventListener(MouseEvent.CLICK, unmuteHandle);
    }

function unmuteHandler(e:Event):void
{
    if(myPhotoFlowPro.unMute())
    {
        unMuteBtn.removeEventListener(MouseEvent.CLICK, unmuteHandle);
        unMuteBtn.parent.removeChild(unMuteBtn);
    }
}
```

# Help

This component is fully supported by the Flashloaded support team through our support forum. You will also find tips and additional information in the forum as well as announcements for version updates: [PhotoFlowPRO Support Forum](#)

*Note: In order to post a question in the forum, you will need to [register](#) by creating a username and password. This registration differs from your account login.*