

# fCMS2

## Component User Guide

User Guide revision 1.1  
[www.flashloaded.com](http://www.flashloaded.com)

# Table of Contents

|  |           |
|--|-----------|
| <b>Installation</b>  | <b>4</b>  |
| <b>Getting Started with the fCMS2 components</b>                     | <b>5</b>  |
| fCMS2Master component  | 5         |
| fCMS2Text component  | 7         |
| fCMS2Pager component   | 8         |
| <b>Creating a Slide Show</b>   | <b>9</b>  |
| <b>Setting Auto Image Creation Options (Auto Thumbnail Creation)</b> | <b>11</b> |
| <b>Editing Content</b>   | <b>12</b> |
| <b>Component Inspector Parameters</b>                                | <b>13</b> |
| fCMS2Pager   | 16        |
| <b>Skinning</b>  | <b>17</b> |
| Skinning the fCMS2File component                                     | 17        |
| Skinning the fCMS2Image component                                    | 17        |
| Skinning the fCMS2Master component                                   | 17        |
| Skinning the fCMS2Pager component                                    | 18        |
| <b>Using other components with fCMS2</b>                             | <b>19</b> |
| <b>FCMS API</b>  | <b>20</b> |
| FCMS Properties  | 20        |
| <b>FCMSMaster API</b>  | <b>21</b> |
| FCMSMaster Events  | 21        |
| FCMSMaster Properties  | 24        |

|                       |           |
|-----------------------|-----------|
| FCMSMaster Methods    | 25        |
| <b>FCMSText API</b>   | <b>26</b> |
| FCMSText Events       | 26        |
| FCMSText Properties   | 26        |
| FCMSText Methods      | 29        |
| <b>FCMSImage API</b>  | <b>30</b> |
| FCMSImage Events      | 30        |
| FCMSImage Methods     | 33        |
| <b>FCMSFile API</b>   | <b>34</b> |
| FCMSFile Events       | 34        |
| FCMSFile Properties   | 35        |
| FCMSFile Methods      | 36        |
| <b>fCMS2Pager API</b> | <b>37</b> |
| fCMS2Pager Events     | 37        |
| fCMS2Pager Properties | 37        |
| <b>FCMSURL API</b>    | <b>38</b> |
| FCMSURL Methods       | 38        |
| <b>Help</b>           | <b>39</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 fCMS2 component.
2. Unzip/extract the fCMS2.zip file that you downloaded. You will find a file called fCMS2.mxp. Double click on this file in order to install the component using Extension Manager.

The fCMS2 components should now be installed in your Flash Components Panel.

Before proceeding with the implementation of fCMS2 on your Flash project, you must set up the backend. Please click on the following links to download the PDF user guide that is appropriate for your web server:

[PHP backend installation for Unix/Linux servers user guide](#)

[.NET backend installation for Windows servers user guide](#)

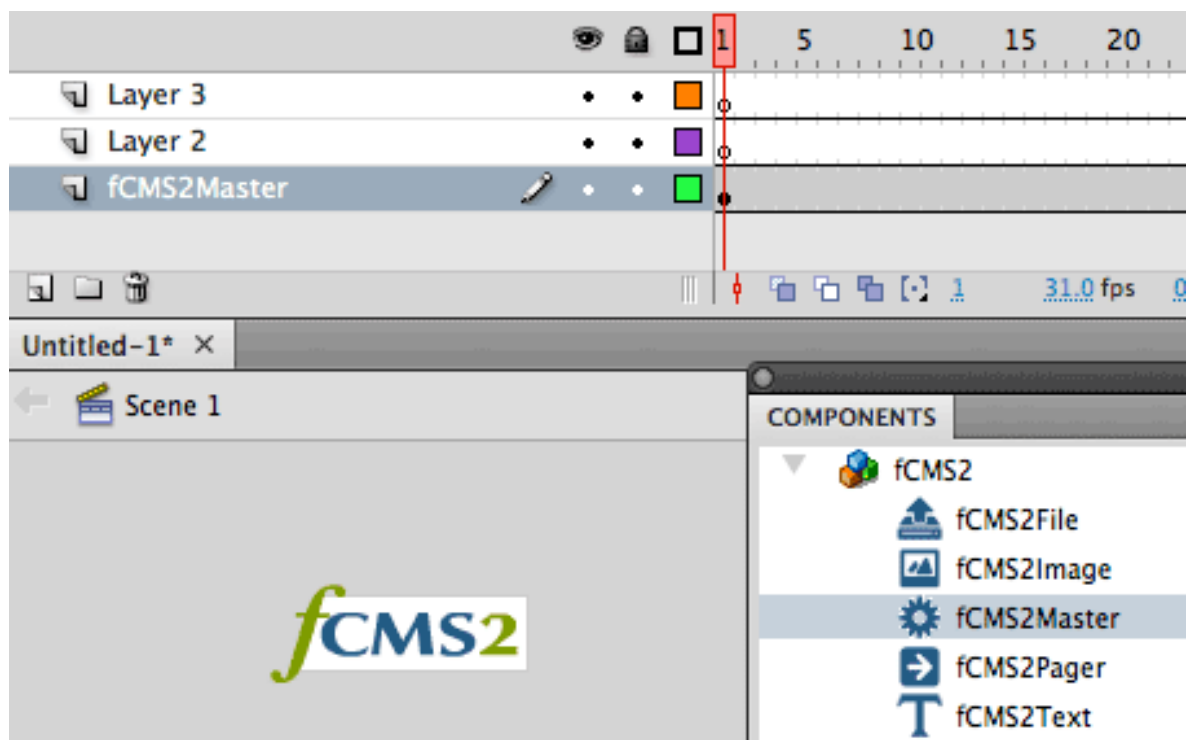
# Getting Started with the fCMS2 components

## fCMS2Master component

The **fCMS2Master** component controls interaction with the backend. It must initialize before any other fCMS2 components.

*Note that only one instance of the fCMS2Master component can be on stage at any one time.*

1. Having installed the fCMS2 component package, restart Flash and start a new **Actionscript 3** file.
2. Name the bottom layer of your timeline 'fCMS2Master'.
3. Drag a copy of the fCMS2Master out of your component panel and onto the stage.



4. With the component still selected on the stage open the component inspector panel (shift + F7).

fCMS2Master has four parameters:

### admin key sequence

This parameter sets the key sequence that should be typed to display the fCMS2 admin panel in a browser. For the purposes of this tutorial leave it at the default 'admin'.

### sequential loading

This parameter sets whether the content should load from the backend as soon as the fCMS2Master loads (false) or whether it should load when required by other components in the file. For the purposes of this tutorial leave it set to 'true'.

### backend path

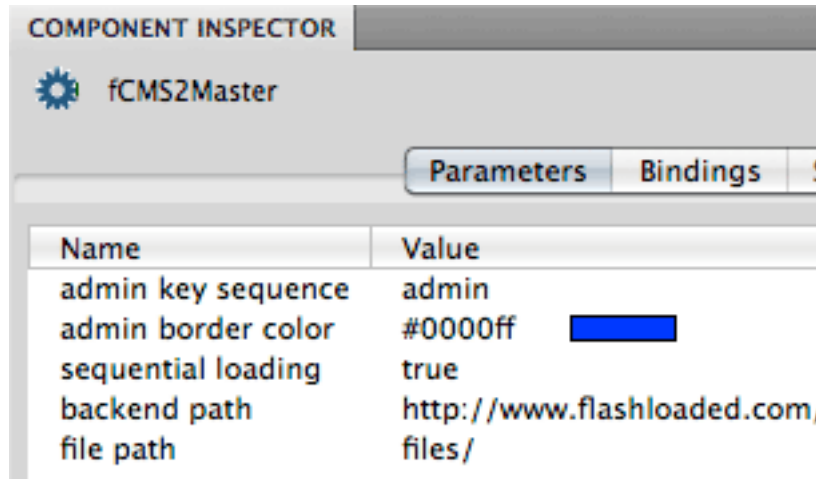
This is the URL of the backend installation. Set this value to the location of your **fcms\_php** or **fcms\_net** folder (the URL must end with a forward slash "/").

*Note: If you are using IIS5 you must add 'index.aspx' (without the quotes) to the end of this value.*

### file path

This is a relative path describing the location of the files folder in relation to the fCMSBacked folder. For the purposes of this tutorial leave this value as 'files/'.

*Note: This value should be identical to the root upload folder value in the config.xml file.*



5. Lock the layer and save your file.
6. Open the backend folder (either *fcms\_php* or *fcms\_net*) and copy the **modules** folder to the same location as your .fla and local swf. You'll need this in the same location as your local swf for testing inside the Flash IDE.

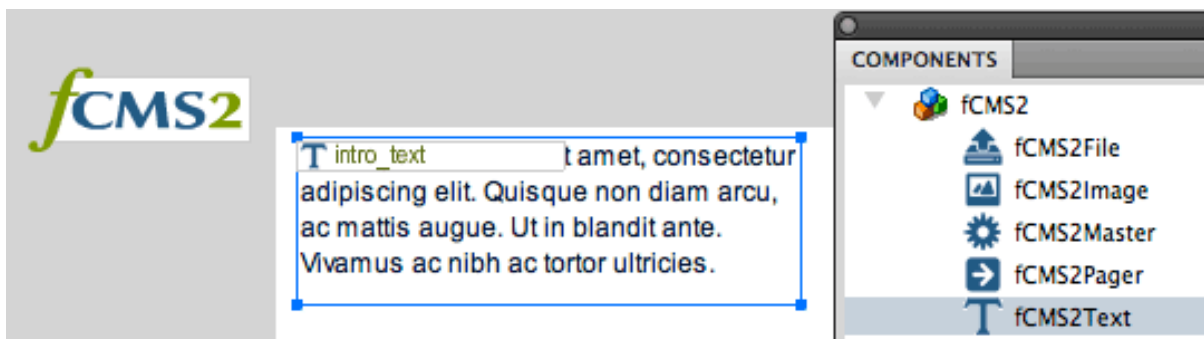
## fCMS2Text component

The fCMS2Text component is used to make textfield content editable. It can be attached to any text field.

1. Create a new layer.

*Note: By default Flash loads from the bottom up. To ensure that the fCMS2Master is the first component initialized it is considered best practice to create all new layers above the layer you created for the fCMS2Master.*

2. Press 'T' to select the Text tool and create a dynamic text field on the new layer.



3. Set the properties of your textfield, font, font size, font color etc. as per normal.
4. Give the text field the instance name 'myTextField' (without the quotes).
5. Drag a copy of the fCMS2Text component out of the fCMS2 folder in the component's panel and drop it onto the text field.
6. With the fCMS2Text instance still selected open the component inspector panel. The fCMS2Text component has ten parameters, for the purposes of this tutorial you only need to use four of them:

### **field name**

This is a unique data identifier. For the purposes of this tutorial enter 'myText' (without the quotes).

### **target text field**

This is the name of the text field associated with this fCMS2Text instance, it is assigned automatically when the fCMS2Text component is dropped onto a text field. Double-check that the value of this field matches the instance name of the text field you created in steps 2-4 (myTextField).

### allow new pages

This parameter controls whether new pages can be created (true) or not (false). Ensure this is set to the default 'true'.

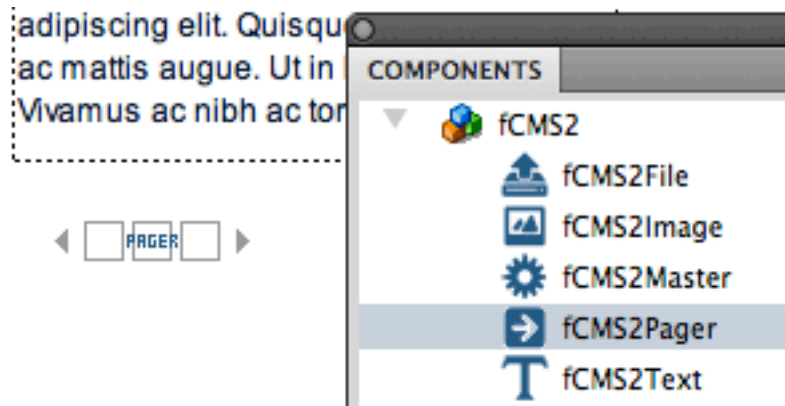
### pager instance name

This parameter is the instance name of the pager component you'll add below to enable you to navigate between pages. For the purposes of this tutorial enter 'myPager' (without the quotes).

## fCMS2Pager component

The *fCMS2Pager* component is used to allow an end user to navigate through pages created by the *fCMS2Text*, *fCMS2Image* or *fCMS2File* components. For the purposes of this tutorial it will be controlling the *fCMS2Text* component you added above.

1. Open the component panel and drag a copy of the *fCMS2Pager* out of the *fCMS2* folder and onto the stage.

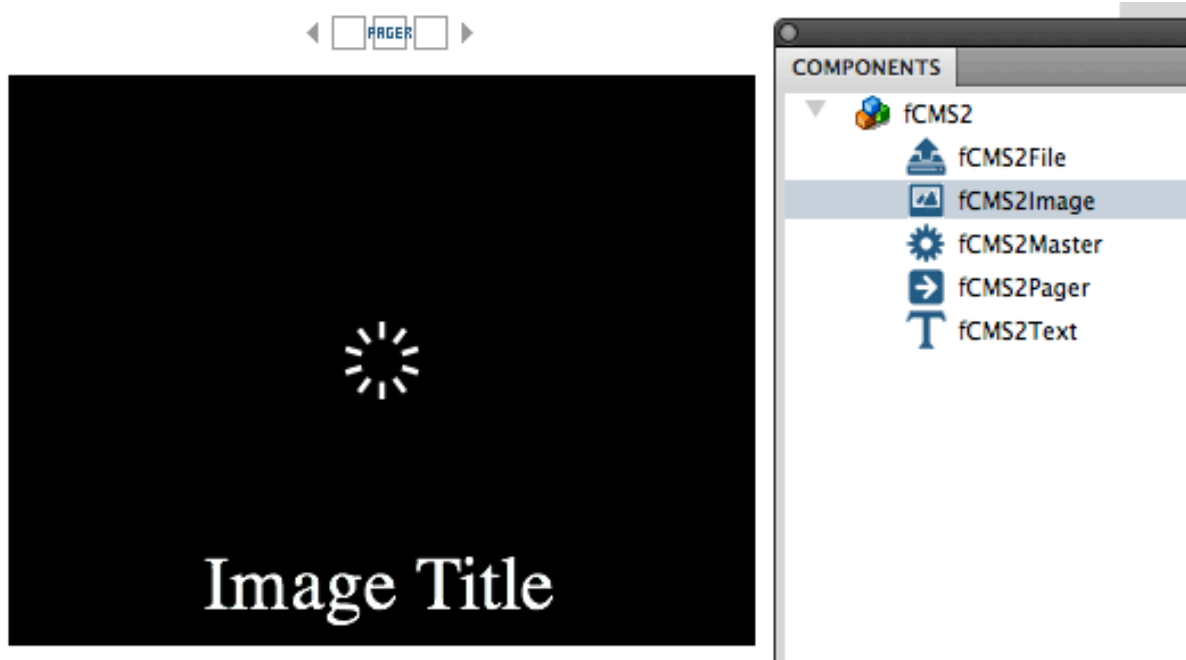


2. Give it the instance name 'myPager'.
3. You can now save and test your file.

# Creating a Slide Show

The fCMS2Image component allows you to upload, display and navigate through images. This tutorial will demonstrate how to set up a simply slide show.

1. Repeat steps 1 – 6 of the Getting Started tutorial to set up your fCMS2Master component.



2. Open the components panel and drag a copy of the fCMS2Image component out of the fCMS2 folder and onto the stage.
3. The fCMS2Image component has nine parameters but for the purposes of this tutorial you will only need three:

**field name**

This is a unique identifier for the image data. For the purposes of this tutorial enter 'myImages' (without the quotes).

**slideshow**

This is a Boolean value determining whether the slideshow should transition through the images automatically (true) or not (false). For the purposes of this tutorial set this value to 'true'.

**pager instance name**

This is the name of the fCMS2Pager instance that will allow you to manually click between images. For the purposes of this tutorial enter 'myPager' (without the quotes).

4. Repeat the steps above to set up an instance of the [fCMS2Pager](#) component.
5. Save and test your file.

# Setting Auto Image Creation Options (Auto Thumbnail Creation)

Each time you use the fCMS2's built-in file manager to upload files to your web server additional images are created according to the definition inside thumbs.xml (inside the config folder). These files are usually used as thumbnails but can also be used as a way to process images to fit your design.

The default thumbs.xml file looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Use http://www.fcmspro.com/thumbExplorer/ to easy define thumbnail size -->
-->
<thumbs version="1.0">
  <!-- Default size. Do not change it as it is used by fcms -->
  <thumb width="40" height="40" crop="yes" type="jpg" />
  <!-- Default preview size -->
  <thumb minSize="80" type="jpg" />
</thumbs>
```

This default definition instructs the fCMS2 backend to create two additional image files after first uploading the original.

Both of these image sizes are used as thumbnails by the file manager and should not be changed.

Once you've finished designing your site, you may wish to add a third definition that will resize the image to the exact area used by the fCMS2Image in your design. Your site will then be as optimised as possible.

For assistance creating a definition to add to your thumbs.xml visit <http://www.fcmspro.com/thumbExplorer/>

For best result we recommend that you always create images in jpg format.

When uploading images using the file manager you will notice that there is a button labelled "Center Image". This button will allow you to define the center of the image, an image centered in the middle will crop at both edges equally, an image centered at the top will crop at the bottom and so forth.

# Editing Content









The principal role of the fCMS2 is to allow easy content management within your flash file. Once you've set up the fCMS2 as outlined above then you need to test the file in order to add new content.

1. Test your movie and when it's loaded type the admin key sequence set in the fCMS2Master's parameters. The default value is "admin".

*If you have difficulty, try retyping the word slowly, for example 'a...d...m...i...n'.*

*You can also initiate a login dialog box by calling the fCMS2Master.login() function.*

2. The login panel will then appear, enter the username and password you set when installing the backend (the default values are 'admin' and 'admin'). Then click OK to log in.
3. Click on an element in your fCMS2 enabled movie to bring up it's administration panel.
4. Edit your content.
5. Click the save button (the icon is a floppy disk).
6. Click the close button (the icon is an 'x').

|   |   |
|---|---|
|  | Close, exits administration mode and prompts for save if there are any changes.                         |
|  | Back, returns to the previous administration bar.   |
|  | Save, saves content to the fCMS2 backend.   |
|  | New page, adds a new page to the currently edited field.  |
|  | Delete page, removes the currently edited page  |
|  | Page controls, allows you to navigate through field pages or sort them by dragging the red page button. |
|  | File Browse, displays the FileManager.  |
|  | File Manager Controls, up one folder; create folder; upload file; re-center image; delete file.         |

# Component Inspector Parameters

## fCMS2Master

| Parameter          | Description   | Example                       |
|--------------------|---|-------------------------------|
| admin key sequence | The key sequence to type to invoke the login dialog.  | "admin"                       |
| admin border color | The border color of the editable text fields in admin mode.   | #ffff00                       |
| sequential loading | All backend data loads when the component is initialised if set to false, if set to true data is loaded when required by other fCMS2 components or scripts. | true                          |
| backend path       | The URL of the fcms folder. <i>When using IIS5 it is necessary to also include the script name, which is index.aspx</i>                                     | "http://www.example.com/fcms" |
| file path          | The relative path to the file directory from the fcms.  | ../                           |

## fCMS2Text

| Parameter         | Description   | Example       |
|-------------------|---|---------------|
| field name        | A unique identifier for this field of data.   | "myText"      |
| target text field | The instance name of the text field that this fCMS2Text instance is linked to.  | "myTextField" |
| ui present        | The controls to display when editing content. Possible values are 'clean', which disables rich text formatting; 'mini', which displays bold, italic, underline, left aligned and justified buttons; 'basic', which displays bold, italic, underline, color, alignment and link buttons; or 'full', which displays all formatting options. | "full"        |
| fonts             | A list of fonts available to the end user. <i>If the text field this fCMS2Text component is linked to uses embedded fonts then only those fonts embedded will be listed.</i>  | "arial"       |
| link color        | The color of text when used as a link.  | #0000ff       |
| admin bg color    | The color of the text field background in admin mode.   | #ffffff       |

| Parameter           | Description   | Example   |
|---------------------|---|-----------|
| allow new pages     | Sets whether new pages can be added to the field (true) or not (false).   | true      |
| pager instance name | The instance name of the fCMS2Pager instance used to navigate through pages.  | "myPager" |
| visible             | Sets the visibility of the component. Possible values are 'always', when the component will be visible to the end user and the admin; 'admin', when the component will only be visible to someone logged in; 'no admin', when the component will only be visible if the user is not logged in; or 'never', in which case the component will always be hidden. | "always"  |

## fCMS2Image

| Parameter             | Description  | Example    |
|-----------------------|--|------------|
| field name            | A unique data identifier for this data.  | "myImages" |
| image size            | The size of image to use. Set to 0 for the original size of the image. Use larger numbers for larger images. See <a href="#">Setting Auto Image Creation</a> for more information.   | 0          |
| slideshow             | Whether the images should auto-transition as a slideshow (true) or not (false).  | true       |
| - delay               | When slideshow is set to true this sets the time in seconds that the image will be displayed for.  | 4.2        |
| - transition duration | Sets the duration (in seconds) of the image transition.  | 1.5        |
| on click              | The action to take when the image is clicked. The possible values are 'next', which transitions to the next image; 'open link', which opens the URL defined for the image; 'event', which triggers the actionscript event; or 'nothing', which does nothing. | "event"    |
| allow new pages       | Sets whether new pages can be added to the field (true) or not (false).  | true       |
| pager instance name   | The instance name of the fCMS2Pager instance used to navigate through pages.   | "myPager"  |

| Parameter | Description   | Example  |
|-----------|---|----------|
| visible   | Sets the visibility of the component. Possible values are 'always', when the component will be visible to the end user and the admin; 'admin', when the component will only be visible to someone logged in; 'no admin', when the component will only be visible if the user is not logged in; or 'never', in which case the component will always be hidden. | "always" |

## fCMS2File

| Parameter           | Description   | Example    |
|---------------------|---|------------|
| field name          | Unique data identifier for this data.   | "myFile"   |
| on click            | The action that should be performed when clicked. The options are 'download' which initiates the file download, 'event' which triggers the actionsript event, or 'nothing' which performs no action.  | "download" |
| display as          | Sets the display mode for the component. The options are 'single page' which displays the current page only (you must then use the fCMS2Pager to navigate the files) or 'list' which displays all pages in the field as a list.   | "list"     |
| roll over alpha     | The alpha value of the background.  | 0.5        |
| - color             | The color of the background.  | #ffffff    |
| allow new pages     | Sets whether new pages can be added to the field (true) or not (false).   | TRUE       |
| pager instance name | The instance name of the fCMS2Pager instance used to navigate through pages. <i>Note that this property will have no effect if the display as parameter is set to 'list'.</i>   | "myPager"  |
| visible             | Sets the visibility of the component. Possible values are 'always', when the component will be visible to the end user and the admin; 'admin', when the component will only be visible to someone logged in; 'no admin', when the component will only be visible if the user is not logged in; or 'never', in which case the component will always be hidden. | "always"   |

## fCMS2Pager

| Parameter        | Description   | Example  |
|------------------|---|----------|
| visible controls | The controls that the component displays. Possible values are 'all', which displays all buttons; 'prev pages next', when the first and last buttons are not used; 'pages', when only the page numbers are used; or 'prev next', when only the previous and next buttons are used.   | "all"    |
| align            | The alignment of the page navigation. Possible values are 'left', 'center' and 'right'.   | "left"   |
| loop pages       | Whether to employ endless paging so that the page after the last is the first (true) or not (false).  | true     |
| num of pages     | The maximum number of pages to display.   | 10       |
| page distance    | The distance between the buttons, measured in pixels.   | 2        |
| over color       | The pager control color when the cursor hovers over it.   | #00ffff  |
| selected color   | The color of the page that has been selected.   | #00ff00  |
| visible          | Sets the visibility of the component. Possible values are 'always', when the component will be visible to the end user and the admin; 'admin', when the component will only be visible to someone logged in; 'no admin', when the component will only be visible if the user is not logged in; or 'never', in which case the component will always be hidden. | "always" |

# Skinning

## Skinning the fCMS2File component

1. Double click the fCMS2File component to access its timeline.
2. Double-click on the default icon to access its timeline.
3. Make your desired changes to the title text field.

*Note you should not change the text field's instance name.*

4. Double-click on the icon again to access the icons for each file type.
5. Make your desired changes to the graphics.

*Note you should leave the frame labels in place.*

## Skinning the fCMS2Image component

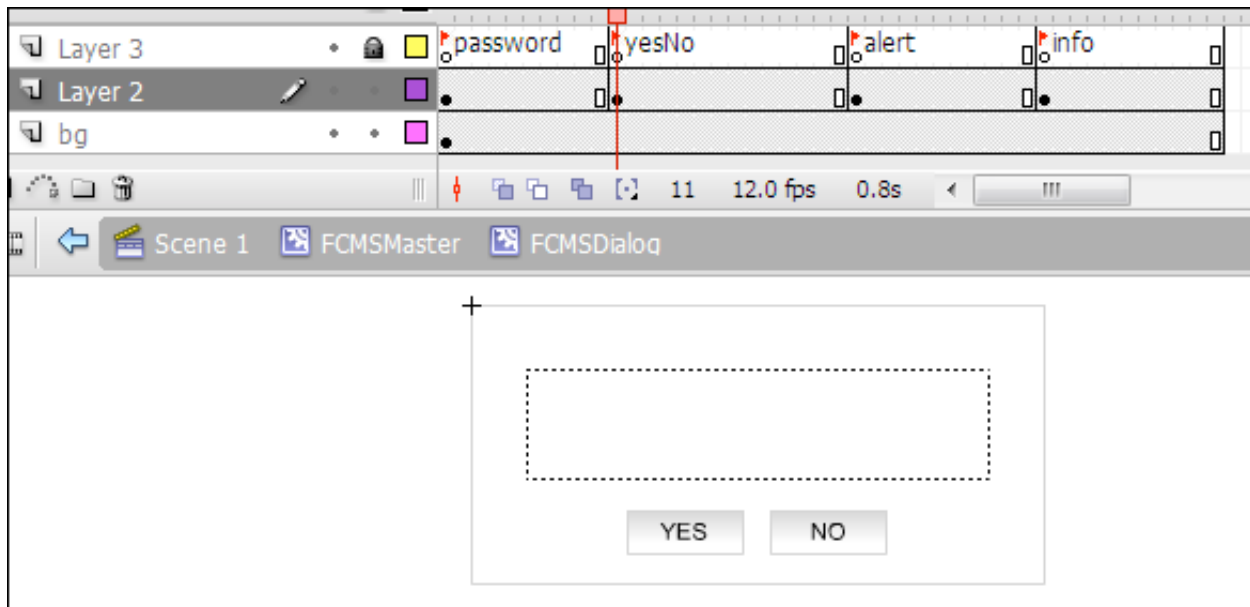
1. Double click the fCMS2Image component to access its timeline.
2. Make any changes you wish to the title text field.
3. Make any changes you like to the background clip. You can access its tint in the properties panel or you can turn off the tint and double click it to access its timeline.
4. Double click on the preloader graphic to access its timeline.

*Note that if you edit the Actionsript on the 'as' layer the onPercentage function must exist.*

## Skinning the fCMS2Master component

1. Double click on the fCMS2Master component to access its timeline.
2. Double click on the FCMSDialog clip to access the different dialog windows used by the component.
3. Edit the password, yesNo, alert and info keyframes to change the appearance of the dialog windows used by the fCMS2.

*Note that you should not change any instance names.*



## Skinning the fCMS2Pager component

1. Double click on the fCMS2Pager to access its timeline.
2. Double click on any of the graphics to access their timelines.
3. Each graphic contains a graphics layer (gfx) and a background layer (bg). Edit either to match your design.

*Note that you must not change their instance names or remove them altogether.*

## Using other components with fCMS2

fCMS2 can also be used to edit the content of some of Flashloaded's ActionScript 3.0 components, such as the MediaPlayer and FlashTextFX components.

Compatibility with more components will be added in the future.

For instructions on using fCMS2 with these components, please download the [fCMS2 Additional Components](#) userguide.

# FCMS API

## fcms.FCMS

The FCMS class contains static properties that help define the fCMS2 system.

## FCMS Properties

**admin**:Boolean

### Availability

Flash Player 9

### Description

Property; Returns whether the fCMS2 is in admin mode (true) or not (false).

### Example

```
import fcms.*;
trace(FCMS.admin);
```

**master**:FCMSMaster

### Availability

Flash Player 9

### Description

Property; returns the instance of the fCMS2Master giving access to its properties.

### Example

```
import fcms.*;
import fcms.controls.*;
var myMasterInstance:FCMSMaster = FCMS.master;
```

**version**:String

### Availability

Flash Player 9

### Description

Property; returns the version number of the fCMS2.

### Example

```
import fcms.*;
trace(FCMS.version);
```

# FCMSMaster API

## fcms.controls.FCMSMaster

The FCMSMaster class contains the core functionality for accessing the fCMS2's features with Actionscript.

## FCMSMaster Events

### complete

#### Availability

Flash Player 9

#### Description

Event; dispatched when a response from the backend is received.

#### Example

```
import fcms.events.*;

myInstance.addEventListener(FCMSEvent.COMPLETE, completeHandler);

function completeHandler(e:Event):void
{
    // do something
}
```

### ioError

#### Availability

Flash Player 9

#### Description

Event; dispatched when an input or output error occurs.

#### Example

```
import flash.events.*;

myInstance.addEventListener(IOErrorEvent.IO_ERROR, ioErrorHandler);

function ioErrorHandler(e:Event):void
{
    // do something
}
```

## net\_error

### Availability

Flash Player 9

### Description

Event; dispatched when an error from the backend is received.

### Example

```
import fcms.events.*;

myInstance.addEventListener(FCMSFetchEvent.NET_ERROR, netErrorHandler);

function netErrorHandler(e:Event):void
{
    // do something
}
```

## open

### Availability

Flash Player 9

### Description

Event; dispatched when a backend request is is sent.

### Example

```
import fcms.events.*;

myInstance.addEventListener(FCMSEvent.OPEN, openHandler);

function openHandler(e:Event):void
{
    // do something
}
```

## reset

### Availability

Flash Player 9

### Description

Event; dispatched when changes are reset.

### Example

```
import fcms.events.*;

myInstance.addEventListener(FCMSEvent.RESET, resetHandler);
```

```
function resetHandler(e:Event):void
{
    // do something
}
```

## securityError

### Availability

Flash Player 9

### Description

Event; dispatched is a security error is encountered whilst content is loading.

### Example

```
import flash.events.*;

myInstance.addEventListener(SecurityErrorEvent.SECURITY_ERROR, errorHandler);

function errorHandler(e:Event):void
{
    // do something
}
```

## status

### Availability

Flash Player 9

### Description

Event; dispatched on entering and exiting admin mode.

### Example

```
import flash.events.*;

myInstance.addEventListener(StatusEvent.STATUS, statusHandler);

function statusHandler(e:Event):void
{
    // do something
}
```

## FCMSMaster Properties

**backendPath**:String

**Availability**

Flash Player 9

**Description**

Property; URL of the fcms

**Example**

```
myInstance.backendPath = "http://example.com/fcms/";
```

**filePath**:String

**Availability**

Flash Player 9

**Description**

Property; the path to files relative to the backendPath.

**Example**

```
myInstance.filePath = "myfiles/";
```

**shellPath**:String

**Availability**

Flash Player 9

**Description**

Property; relative path of the fCMS2 module.

**Example**

```
trace(myInstance.shellPath);
```

## FCMSMaster Methods

**login**(p:Boolean = false):void

### Availability

Flash Player 9

### Description

Method; displays fCMS2 login dialog.

### Parameter

keepName. If set to true the last entered username will be kept.

### Example

```
myInstance.login();
```

**logout**():void

### Availability

Flash Player 9

### Description

Method; logs out of the fCMS2 admin mode.

### Example

```
myInstance.logout();
```

**save**(e:FCMSEvent = null):void

### Availability

Flash Player 9

### Description

Method; saves any changes to the backend.

### Example

```
myInstance.save();
```

# FCMSText API

## fcms.controls.FCMSText

The FCMSText class contains events, properties and methods for interacting with the fCMS2Text component.

## FCMSText Events

**change**:Event

### Availability

Flash Player 9

### Description

Event; Dispatched when the content changes

### Example

```
import fcms.controls.*;
import flash.events.*;

myInstance.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(e:Event):void
{
    // do something
}
```

## FCMSText Properties

**basicPreset**:Array

### Availability

Flash Player 9

### Description

Property; And array of strings that represent the controls for rich text formatting. The default is ["bold", "italic", "underline", "color", 4, "align\_left", "align\_center", "align\_right", "align\_justify", 4, "url"].

### Example

```
myInstance.basicPreset = ["bold", "italic", "url"];
```

**colors**:Array

**Availability**

Flash Player 9

**Description**

Property; Array containing the colors for the color palette of this instance of fCMS2Text.

**Example**

```
fcms2textinstance.colors=[0xff0000, 0xffffffff];
```

**data**:\*

**Availability**

Flash Player 9

**Description**

Property; Array containing page data.

**Example**

```
trace(myInstance.data);
```

**fieldName**:String

**Availability**

Flash Player 9

**Description**

Property; name of the field.

**Example**

```
trace(myInstance.fieldName);
```

**fontList**:Array

**Availability**

Flash Player 9

**Description**

Property; the list of fonts to use for editing.

**Example**

```
trace(myInstance.fontList);
```

**page**:uint

**Availability**

Flash Player 9

**Description**

Property; the current page of the field.

**Example**

```
trace(myInstance.page);
```

**sizes**:Array

**Availability**

Flash Player 9

**Description**

Property; Array containing the font sizes for this instance of fCMS2Text.

**Example**

```
fcms2textinstance.sizes=[10,12,14];
```

**target**:TextField

**Availability**

Flash Player 9

**Description**

Property; the textfield instance associated with this component.

**Example**

```
trace(myInstance.target);
```

**type**:String

**Availability**

Flash Player 9

**Description**

Property; data type.

**Example**

```
trace(myInstance.type);
```

## FCMSText Methods

`drawNow():void`

### Availability

Flash Player 9

### Description

Method; forces the component to draw content.

### Example

```
myInstance.drawNow( );
```

# FCMSImage API

## fcms.controls.FCMSImage

The FCMSImage class contains events, properties and methods for working with the fCMS2Image component.

## FCMSImage Events

**change**:Event

### Availability

Flash Player 9

### Description

Event; Dispatched when the content changes

### Example

```
import fcms.controls.*;
import flash.events.*;

myInstance.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(e:Event):void
{
    // do something
}
```

**open**:FCMSEvent

### Availability

Flash Player 9

### Description

Event; Dispatched when the 'on click' parameter is set to 'event' and the file is clicked.

### Example

```
import fcms.controls.*;
import fcms.events.*;

myInstance.addEventListener(FCMSEvent.OPEN, openHandler);

function openHandler(e:Event):void
{
    // do something
}
```

## FCMSImage Properties

**data**:\*

**Availability**

Flash Player 9

**Description**

Property; Array containing page data.

**Example**

```
trace(myInstance.data);
```

**delay**:Number

**Availability**

Flash Player 9

**Description**

Property; slideshow delay in seconds

**Example**

```
trace(myInstance.delay);
```

**fieldName**:String

**Availability**

Flash Player 9

**Description**

Property; name of the field.

**Example**

```
trace(myInstance.fieldName);
```

**page**:uint

**Availability**

Flash Player 9

**Description**

Property; the current page of the field.

**Example**

```
trace(myInstance.page);
```

**slideshow**:SlideShow

**Availability**

Flash Player 9

**Description**

Property; returns a reference to the SlideShow class instance.

**Example**

```
trace(myInstance.slideshow);
```

**transitionDuration**:Number

**Availability**

Flash Player 9

**Description**

Property; transition duration in seconds.

**Example**

```
trace(myInstance.transitionDuration);
```

**type**:String

**Availability**

Flash Player 9

**Description**

Property; the data type.

**Example**

```
trace(myInstance.type);
```

## FCMSImage Methods

`drawNow():void`

### Availability

Flash Player 9

### Description

Method; forces the component to draw content.

### Example

```
myInstance.drawNow( );
```

# FCMSFile API

## fcms.controls.FCMSFile

The FCMSFile class contains methods and properties for uploading files to the backend.

## FCMSFile Events

**change**:Event

### Availability

Flash Player 9

### Description

Event; Dispatched when the content changes

### Example

```
import fcms.controls.*;
import flash.events.*;

myInstance.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(e:Event):void
{
    // do something
}
```

**open**:FCMSEvent

### Availability

Flash Player 9

### Description

Event; Dispatched when the 'on click' parameter is set to 'event' and the file is clicked.

### Example

```
import fcms.controls.*;
import fcms.events.*;

myInstance.addEventListener(FCMSEvent.OPEN, openHandler);

function openHandler(e:Event):void
{
    // do something
}
```

## FCMSFile Properties

**data:**\*

### Availability

Flash Player 9

### Description

Property; Array containing page data.

### Example

```
trace(myInstance.data);
```

**fieldName:**String

### Availability

Flash Player 9

### Description

Property; name of the field.

### Example

```
trace(myInstance.fieldName);
```

**fileRef:**FileReference

### Availability

Flash Player 9

### Description

Property; instance of the FileReference class used for file download.

### Example

```
trace(myInstance.fileRef);
```

**page**:uint

**Availability**

Flash Player 9

**Description**

Property; the current page of the field.

**Example**

```
trace(myInstance.page);
```

**type**:String

**Availability**

Flash Player 9

**Description**

Property; the data type.

**Example**

```
trace(myInstance.type);
```

## FCMSFile Methods

**drawNow**():void

**Availability**

Flash Player 9

**Description**

Method; forces the component to draw content.

**Example**

```
myInstance.drawNow();
```

# fCMS2Pager API

## fcms.controls.fCMS2Pager

The fCMS2Pager class contains events and properties for interacting with the fCMS2Pager class.

## fCMS2Pager Events

**change**:Event

### Availability

Flash Player 9

### Description

Event; Dispatched when the page changes

### Example

```
import fcms.controls.*;
import flash.events.*;

myInstance.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(e:Event):void
{
    // do something
}
```

## fCMS2Pager Properties

**page**:uint

### Availability

Flash Player 9

### Description

Property; gets or sets the current page for the component.

### Example

```
myInstance.page = 2;
```

# FCMSURL API

## fcms.utils.FCMSURL

The FCMSURL class is a utility class to assist in your actionscript programming.

## FCMSURL Methods

**getThumb**(url:String, size:int, extension:String):String

### Availability

Flash Player 9

### Description

Method; returns the path to the specified images thumbnail. The method takes three parameters: url, which is the URL of the image; size, which is the size of the thumbnail as defined in thumbs.xml; extension, which is the file extension of the thumbnail.

### Example

```
import fcms.utils.*;
trace(getThumb("myImage.jpg", 0, "jpg"));
```

# 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: [fCMS2 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.*