

MediaPlayer

User Guide revision 1.2
www.flashloaded.com

Table of Contents

Installation	3
Getting started	4
XML	6
Component Inspector Parameters	8
MediaPlayer Component	8
MediaList Component	9
SoundSpectrum Component	9
Skinning	10
ActionScript Events	14
ActionScript Properties	16
ActionScript Methods	18
Help	21

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 MediaPlayer component.
2. Unzip/extract the MediaPlayer.zip file that you downloaded. You will find a file called MediaPlayer.mxp. Double click on this file in order to install the component using Extension Manager.

MediaPlayer should now be installed in your Flash Components Panel.

Getting started

1. Having installed MediaPlayer using the Adobe Extension Manager, start a new Flash ActionScript 3 file and save it.
2. Locate the MediaPlayer folder in the Components panel and double click on it to expand it. You will find the following 3 components inside this folder:

MediaList - used to display a playlist of songs for the MediaPlayer.

MediaPlayer - the main MediaPlayer component.

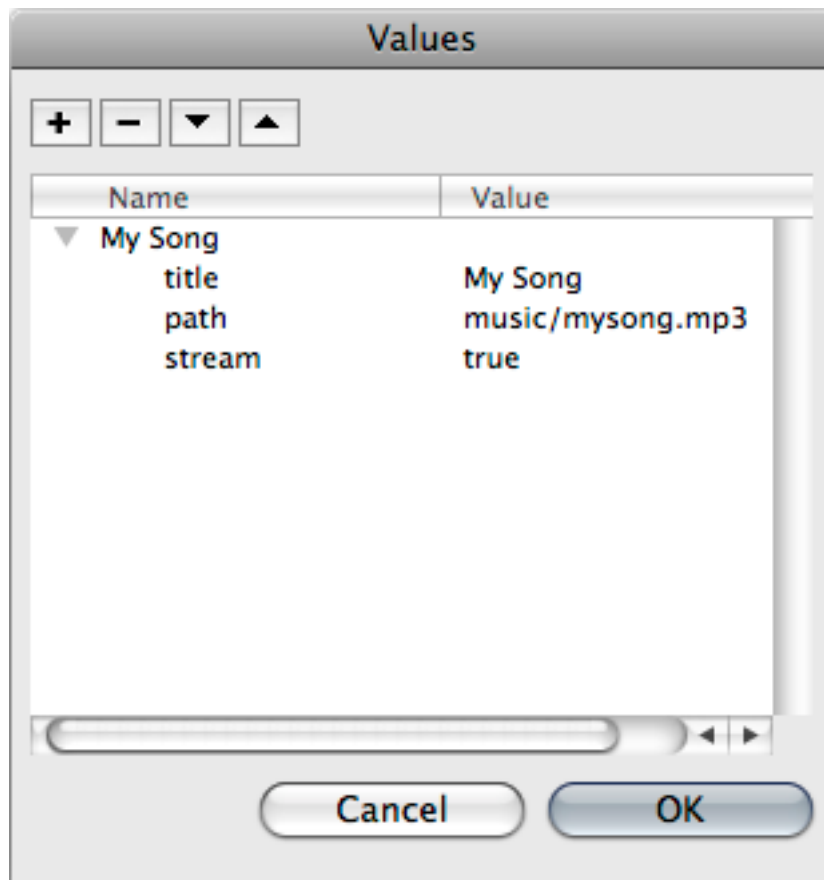
SoundSpectrum - displays a graphical representation of the music playing.

3. Drag and drop the **MediaPlayer** component onto the stage. This is the main player which contains the controls and information about the current sound file that is playing. The default player has a basic default skin. Please refer to the [skinning](#) section for instructions on skinning the player and how to remove unwanted controls.
4. Click on the component and open the Component Inspector panel (shift +F7).
5. The playlist of music files can be entered either directly into to the Component Inspector, or through an external XML file. An advantage in using an external XML file is that the playlist can be changed without the need to republish the SWF.

Please refer to the [XML](#) section for instructions on creating an XML playlist. If you are using the XML option, enter the name and path to the XML file in the **xml source** parameter.

To enter the playlist directly in the Component Inspector:

- a. Double click on the empty square brackets in the **playlist** parameter. This will open the values window.
- b. Press the + button to define a sound file.
- c. Enter a title for the sound file (which will appear when this track is playing), the path and filename of the sound file and select whether the file should play while streaming the download or download fully before playing.
- d. Repeat steps b and c to add more file to the playlist. Press OK when done.



6. At this stage, you can already test MediaPlayer with the default parameters, to ensure that you have set it up correctly. Press Ctrl+Enter (win) or Cmnd+Enter (mac) to test your movie.
7. To add a playlist, drag and drop the **MediaList** component onto the stage at the desired location. Click on the *MediaPlayer* that's on the stage and give it an instance name, then click on the *MediaList* component, open the Component Inspector panel (shift +F7) and enter the instance name that you just gave to the *MediaList* component in the **player instance name** parameter of the *MediaList* component. Please refer to the [skinning](#) section for instructions on skinning the playlist.
8. To add a graphical representation of the sound, drag and drop the SoundSpectrum onto the stage. Please refer to the [skinning](#) section for instructions on skinning this component.
9. You can change the various parameter settings in the Component Inspector for each of these three components to obtain the desired look and feel. Please see the [Component Inspector parameters](#) section for a description on each setting.

XML

The playlist of sound files for the MediaPlayer can be specified using an external XML file. You can also add custom fields within the XML file.

1. Open your favorite plain text editor (for example Notepad on Windows or TextEdit on Mac) and start a new file. *Note: If you are using TextEdit on Mac, choose Format > Make Plain Text*
2. Begin your file with the following line:

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

This is the standard xml declaration.

3. Add the following lines to your xml file (the bold lines are the new additions)

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

4. Add the following lines between the gallery tags (the bold lines are the new additions)

```
<?xml version="1.0" encoding="utf-8"?>  
<playlist>  
  <track>  
    <title>Title of first song</title>  
    <stream>true</stream>  
    <path>music/song1.mp3</path>  
    <cover>covers/song1.jpg</cover>  
    <screenshot>screenshots/song1.jpg</screenshot>  
  </track>  
  <track>  
    <title>Title of second song</title>  
    <stream>>false</stream>  
    <path>music/song2.mp3</path>  
    <cover>covers/song2.jpg</cover>  
    <screenshot>screenshots/song1.jpg</screenshot>  
  </track>  
</playlist>
```

Here's an explanation of the tags used in the above examples:

title (optional): defines a title for the track which will be displayed in the playlist. If this parameter is omitted, the name of the file or ID3 tag details will be used (artist - songname) instead.

stream (optional): sets whether the file should stream download or download fully before playing. If this parameter is omitted, the file will be streamed by default.

- path:** the path and filename of the mp3 or flv file.
- cover (optional):** the path and filename of the small thumbnail image to display in the MediaList component for this file.
- screenshot (optional):** the path and filename of the optional image to display in the video display while the music is playing.

Note: You can add any custom fields to the XML file. If you wish to add text, you can do this by adding a textfield to your Media Player and giving it an instance name. You would assign the data to the textfield by using the same name in the XML file. For example, if you add a textfield to your player with instance name *description*, the XML entry would look something like this:

```
<track>
  <title>Title of first song</title>
  <stream>true</stream>
  <path>music/song1.mp3</path>
  <description>Description of this song</description>
</track>
```

5. Save the XML file to the same folder as your Flash file. In this example, we have given the XML file the name: *music.xml*
6. Return to your Flash file. Enter the name and path to the XML file that you just created in the **xml source** parameter of the MediaPlayer that's on the stage.

Note: If your .swf file will be in a different folder to the HTML file in which it is embedded, you should enter the path to the XML file, relative to the location of the .html file.

7. Press Ctrl+Enter (Win) or Cmnd+Enter (Mac) to test your MediaPlayer.

Component Inspector Parameters

MediaPlayer Component

Parameter	Description	Example
playlist	Use this parameter to define the playlist directly in the Component Inspector, instead of using the XML option.	
- xml source	The path and name of the XML file containing the playlist data (if the playlist is not defined in the playlist parameter).	music.xml
- autoplay	Defines whether the music will start playing automatically upon initialization or not.	true
- continuous	Defines whether the next song in the playlist will play automatically when the previous song finishes or not.	true
- loop	Defines whether the playlist loops continuously after the last track has player.	true
- shuffle	If set to true, the playlist will be randomized before it is first played.	false
Buffer time	The number of streaming seconds to buffer the sound or video files when streaming the download.	10
Show remaining time	If set to true, the remaining time will be displayed as opposed to the elapsed time.	false
scroll speed	The speed at which the song title scrolls.	100
single play	This parameter is used if there are multiple instances of the MediaPlayer component that are either on the same stage, or embedded in the same html page. If set to true, only one instance will play at a time, while others will stop playing automatically.	true
autoHide	Sets whether to auto hide the controls and if they should only be hidden in fullscreen mode. Options available are: <i>never, always, fullscreen</i>	fullscreen
scrubbing	Sets whether to scrub the sound or video files while dragging the progress bar left or right.	true

MediaList Component

Parameter	Description	Example
player instance name	The instance name of the MediaPlayer component that this should connect to.	myplayer
auto mask	Sets whether to mask the playlist automatically or not.	true
- visible tracks	The number of tracks to display in the list.	6
- acceleration	Autoscroll sensitivity setting.	10

SoundSpectrum Component

Parameter	Description	Example
orientation	Sets whether to display at vertically or horizontally.	vertical
mode	<i>frequency</i> : Displays frequency ranges <i>volume</i> : Displays according to the sound volume	frequency
- bands	The number of graphic bands to display.	10
distance	Distance between the graphic bands.	1
autosize	If set to true, the bands will be automatically scaled to fit the components area.	true
laziness	The amount of time that bands require to return to a zero value.	2
gain	Signal gain.	2
attenuation	Attenuates higher frequency ranges for more linear distribution. Options: <i>none</i> , <i>linear</i> , <i>cubic</i>	linear

Skinning

The MediaPlayer, MediaList and SoundSpectrum components can all be skinned to match your desired look and feel.

Skinning the MediaPlayer component

Double click anywhere on the MediaPlayer component that's on the stage in order to skin the elements.

You should now see two movie clips: **Mediaplayer_Video** and **MediaPlayer_Controls_Full**



Skinning the MediaPlayer controls

Double click on the *MediaPlayer_Controls_Full* movie clip in order to edit the controls. Double click on the movie clip of each element that you wish to skin. You can also rearrange or remove any elements or buttons. When you double click on a button, you will see that each button has frames labeled according to button state (*_up*, *_over*, *_down*). Move the playhead along the timeline to change the look of the button in the frames for each of these states.

The *MediaPlayer_Controls_Full* skin can have any number of additional textfields that will be populated automatically according to the field of the same name that you add to the XML file, or according to the

value of the ID3 tag when an mp3 is played. Click [here](#) to see the possible ID3 tag names.

There are two special textfields:

title: This is the only field that will scroll endlessly if the content is longer than the size of the textfield. The value of this field is taken from the *title* value defined in the Component Inspector or inside the XML file. If no value is defined, this field will be populated by the ID3 tags (artist and song name). If the mp3 does not contain the ID3 tag, the filename will be used instead.

ordinal: This field contains the order value of the song in the playlist.

MediaPlayer controls behavior in fullscreen mode

The *bar* and *position_slider* skin elements are the only movie clips that change width in fullscreen mode. All movie clips inside *position_slider* will be resized except *slider*.

The size of *position_slider* in fullscreen mode depends on its x position and the closest control movie clip on the right side of *position_slider*. All control movie clips on the right of *position_slider* will be moved to the right edge of the full screen.

The entire *MediaPlayer_Controls_** movieclip will be aligned to the bottom of the fullscreen

Skinning the MediaPlayer video

The *MediaPlayer_Video* movie clip contains the video controls that appear on the screen while a video is playing, the buffering movie clip, the logo and the video display screen. Double click on the *MediaPlayer_Video* movie clip in order to skin these elements. The *MediaPlayer_VIDEO_PLAY* movie clip contains frames for the play and pause buttons in their up and over states.



MediaPlayer video controls behavior in fullscreen mode

When viewing a video in fullscreen mode, the *video_display* movie clip will be resized accordingly. The *video_play* and *buffering* movie clips will be center aligned. The *logo* movie clip will move depending on its initial position (aligned right, bottom-right or bottom) from the *video_display* movie clip.

Skinning the SoundSpectrum component

Double click anywhere on the SoundSpectrum component that's on the stage in order to edit the color of the bar. You may need to zoom in to see the bar clearly.

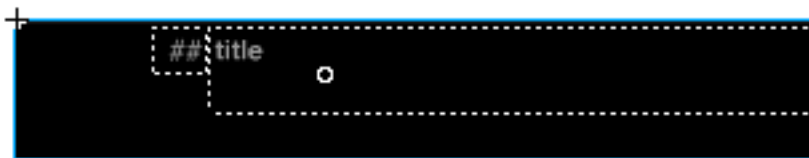


You can now edit this movie clip. This movie clip must have 100 frames. The component will send the playhead to the frame according to the value of the band.

Skinning the MediaList component

Double click anywhere on the MediaList component that's on the stage. This will get to the MediaPlayer_TRACK movie clip. Double click on the MediaPlayer_TRACK movie clip in order to skin these elements.

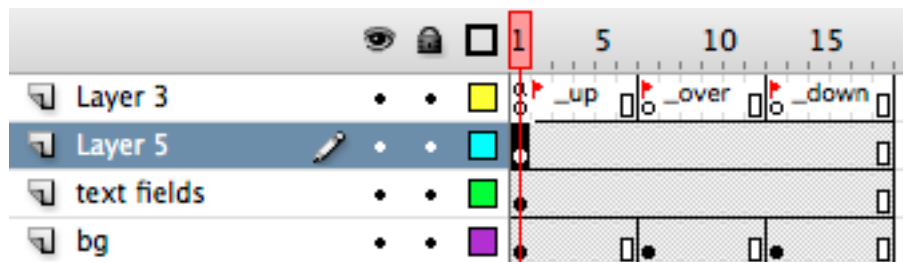
You should now see two textfields and an empty movie clip in the top left corner with the instance name *cover_mc*.



cover_mc is the holder for the cover thumbnail image. This can be moved to change the position of the image. If the *cover_mc* movie clip is deleted, the cover will be positioned at the 0,0 coordinates of the MediaPlayer_TRACK movie clip.

Click on the textfield on the right to change the style of the text for the track number. Click on the textfield on the right to change the style of the text for the track title.

The background that appears behind the titles on mouse up, over and down can be edited by moving the playhead along the timeline and editing the background in the *bg* layer for each of these frame labels.



The *MediaPlayer_TRACK* skin can have any number of additional textfields that will be populated automatically according to the field of the same name that you add to the XML file, or according to the value of the ID3 tag when an mp3 is played. Click [here](#) to see the possible ID3 tag names.

There are two special textfields:

title: The value of this field is taken from the *title* value defined in the Component Inspector or inside the XML file. If no value is defined, this field will be populated by the ID3 tags (artist and song name). If the mp3 does not contain the ID3 tag, the filename will be used instead.

ordinal: This field contains the order value of the song in the playlist.

ActionScript Events

Events are called whenever the MediaPlayer or MediaList perform the specified action. The component includes an event class called MediaEvent in the import `com.flashloaded.media.events.MediaEvent` package.

The event has an `item` property which holds the following element properties:

trackData: all of the data contained within the *track* tags in the XML file. If, for example, you add a custom tag called *cover*, *trackData.cover* will return the value specified in the *cover* tag.

MediaPlayer events:

MediaEvent.POSITION

Broadcasted whenever the playhead position is changed.

MediaEvent.VOLUME

Broadcasted whenever the volume is changed.

MediaEvent.PAN

Broadcasted whenever the pan bar is dragged.

MediaEvent.PLAY

Broadcasted whenever the track starts playing.

MediaEvent.PLAY_END

Broadcasted whenever a track ends.

MediaEvent.STOP

Broadcasted whenever the stop button is pressed.

MediaEvent.PAUSE

Broadcasted whenever the pause button is pressed.

MediaList events:

MediaEvent.CHANGE

Broadcasted when currently playing track is changed.

MediaEvent.DISPLAY_CHANGE

Broadcasted after display is changed

MediaEvent.PLAYLIST_CHANGE

Broadcasted after playlist is changed

MediaEvent.PLAYLIST_TRACK

Broadcasted after track skin is populated

Example using events:

```
import com.flashloaded.media.events.MediaEvent;
mp.addEventListener( MediaEvent.CHANGE, onChange );

function onChange( e:* )
{
    var trackInfo:Object = e.target.playlist.current;
    for ( var name:String in trackInfo ) trace( name + ":" +
        trackInfo[ name ] );
}
```

ActionScript Properties

MediaPlayer

length:int

Availability

Flash Player 9

Description

Property; returns the length (in milliseconds) of the current track. This is a read only property.

Example

```
trace(myPlayer.length);
```

pan:Number

Availability

Flash Player 9

Description

Property; returns the sets the panorama. The left-to-right panning of the sound, ranging from -1 (full pan left) to 1 (full pan right).

Example

```
myPlayer.pan = 0;
```

playlist

Availability

Flash Player 9

Description

Property; a reference to the playlist object inside the MediaPlayer which holds the playlist data.

Example

Outputs the title of the currently playing song:

```
trace(myPlayer.playlist.current.title);
```

- **data**

An array with information about playlist (path, title, cover etc) that is set either through the data parameter in the Components Inspector or through XML.

```
MediaPlayer.playlist.data
```

- **current**

```
MediaPlayer.playlist.current
```

position:Number

Availability

Flash Player 9

Description

Property; gets and sets the position of the current track in milliseconds.

Example

```
myPlayer.position = 1550;
```

relativePosition:Number

Availability

Flash Player 9

Description

Property; gets and sets the relative position of the current track (0-1)

Example

Sets the song/video position to half of the total running time:

```
myPlayer.relativePosition = 0.5;
```

volume:Number

Availability

Flash Player 9

Description

Property; returns the sets the volume.

Example

Set the volume on both channels to 50% of the maximum:

```
myPlayer.volume = 0.5;
```

ActionScript Methods

MediaPlayer

doPlay

Availability

Flash Player 9

Description

Method; prepares and starts playback of current track.

Syntax

```
MediaPlayerInstance.doPlay();
```

Example

```
myPlayer.doPlay();
```

doStop

Availability

Flash Player 9

Description

Method; stops playback and closes the stream. If the close parameter is set to true, the mp3 will be unloaded completely and streaming will be halted.

Syntax

```
MediaPlayerInstance.doStop(close:Boolean = false);
```

Example

```
myPlayer.doStop(close:Boolean = false);
```

doPause

Availability

Flash Player 9

Description

Method; pauses playing.

Syntax

```
MediaPlayerInstance.doPause();
```

Example

```
myPlayer.doPause;
```

doNext

Availability

Flash Player 9

Description

Method; jump to the next track.

Syntax

```
MediaPlayerInstance.doNext();
```

Example

```
myPlayer.doNext();
```

doPrev

Availability

Flash Player 9

Description

Method; jump to the previous track.

Syntax

```
MediaPlayerInstance.doPrev();
```

Example

```
myPlayer.doPrev();
```

loadXML

Availability

Flash Player 9

Description

Method; loads an external XML.

Syntax

```
MediaPlayerInstance.loadXML(path:String);
```

Example

```
myPlayer.loadXML("filename.xml");
```

playlist

Availability

Flash Player 9

Description

Method; a reference to the playlist object inside the MediaPlayer which holds the playlist data.

- *clear*

```
MediaPlayer.playlist.clear(); //clears the playlist.
```

- *add*

```
MediaPlayer.playlist.add(); //adds track to playlist.
```

Syntax

```
MediaPlayerInstance.playlist.add(path:String, stream:Boolean,  
title:String);
```

Example

```
myPlayer.playlist.add("song2.mp3", true, "Another song");
```

- *addItem*

```
MediaPlayer.playlist.addItem(); //adds track to playlist.
```

Syntax

```
MediaPlayerInstance.playlist.addItem(obj);
```

Example

```
myPlayer.playlist.addItem({path:"song2.mp3", stream:true,  
title:"Another song", cover:"path.img.jpg", whatever:123});
```

- *doShuffle*

```
MediaPlayer.playlist.doShuffle(); //shuffles the playlist.
```

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:

[MediaPlayer 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.