PLYmedia Inc.
Flow Player Integration Document

Description

PLYmedia add-on is a transparent layer, loaded on top of a video, allowing you to create and display custom content synchronized with the stream, without altering the latter in any way.

Without the headache of video editing, you will be able to create and synchronize to your FLV stream dynamic content such as subtitles, images, SWF clip arts and more, all streaming from our servers in real-time.

This tutorial will accommodate the easiest way to integrate PLYmedia add-on with FLV FlowPlayer FLV.

You can download the latest version of the player here:

http://flowplayer.org/download

Further information and FAQ can be found here:

http://flowplayer.org/documentation

PLYmedia add-on can be loaded from here:

http://www.bubbleply.com/plyPlayer.swf

PLYController class can be downloaded here:

http://www.bubbleply.com/PLYController.zip

Installation

PLYmedia add-on integration with FlowPlayer includes 7 main steps:

1. Adding the PLYController.as class into the org/flowplayer library.
2. Defining a switch that will allow you to turn on/off the add-on at your convenience.
3. Defining an empty movie clip and PLYcontroller class instance.
4. Defining a dispatcher to inform the PLYcontroller class of the current played flv path.
5. Loading the plyPlayer into the created movieclip.
6. Implementing a resize function when completing loading and going to full screen and back, to fit the video proportions at all time.
7. Sending the time parameter to the add-on in order synchronize it to the video.

First, unzip and copy the downloaded class:

To avoid scope issues we have written a class that has all the needed functions to load, resize, send variables and update plyPlayer with the video stream time. We will address it throughout the different stages of the player.

Copy the file PLYController.as into org/flowplayer library.

You will need to make changes in seven classes to implement those steps:

In order to implement use of showPLYmedia parameter from html to turn on/off the PLYmedia
we will take the following three steps:

1. Open Config.as in folder org/flowplayer/config

In the functions section we will define showPLYmedia function.

```
public function showPLYmedia():Boolean;
```

2. Open DefaultsConfig.as in folder org/flowplayer/config

Here we will name the desired function also showPLYmedia, it will return true assuming we want PLYmedia to appear, later on we will take care of the negative case as well.

```
public function showPLYmedia():Boolean {
        return false;
}
```

3. Open HierarchicalConfig.as in folder org/flowplayer/config

Finaly we will define the function that will return true/false according to the parameter PLYmedia in the html

```
public function showPLYmedia():Boolean {
        if (this.config.showPLY == undefined) return getChild().showPLYmedia();
        else if (this.config.showPLY =='true') return true;
        else if (this.config.showPLY =='false') return false;
}
```

Most action will take place in the FLVDisplay, here we will define, load, resize and pass all the needed parameters to PLYmedia:

4. Open FLVDisplay.as in folder org/flowplayer

Before anything else, we will import the PLYController class.

```
import org.flowplayer.PLYController;
```

Next lets define two new parameters, PLYController class instance and the host movieclip.

```
private var PLYCtrl:PLYController;
private var mcPLYmedia:MovieClip;
```

In the functions section, lets define the function that will manage various loading and defining actions.

```
private function managePLYmenu(toDo:String) {
        logger.debug("FLVDisplay >> createPLYmenu( "+toDo+" )");
        switch(toDo) {
                case 'init':
                        createEmptyMovieClip("mcPLYmedia", getNextHighestDepth());
                        PLYCtrl = new PLYController(mcPLYmedia,this.control);
                break;
                case 'load':
                        PLYCtrl.LoadPLY(getVideoAreaWidth(),getVideoAreaHeight(),this.control);

                break;
        }
}
```

Now lets define another function that will be called from NewAbstractMovieClip class when the flowplayer is reached its final, wanted, stage size. The function will initiate plyPlayer loading process.

```
private function onSetSizeDoPLY():Void {
        logger.debug("FLVDisplay >> onSetSizeDoPLY()");
        managePLYmenu('load');
}
```

In function that initiates all the main player controls will make a call to initiate the scope and the movieclip.

```
public function init(controller:MediaControl) {
        ...
        if (getConfig().showPLYmedia()) {
                managePLYmenu('init');
        }
        ...
}
```

NewAbstractMovieClip, as the name implies, is responsible for movieclip structuring. We will figure out the last stage where the stage reaches the predefined size and call a function to start loading plyPlayer.

5. Open NewAbstractMovieClip.as in folder org/flowlib

Please copy the changes of this function in the correct order as in the following example.

```
public function setSize(nW:Number, nH:Number, forceArrange:Boolean):Void {
        if (! forceArrange) {
                if (nW == __width && nH == __height) {
                        return;
                }
                if (! (nW > 0 || nH > 0)) {
                        return;
                }
        }
        var gotoSetPLY:Boolean = (getConfig().showPLYmedia() && this._name=='display' &&
__width!=0 && __height!=0 && nW!=0 && nH!=0);
        _xscale = 100;
        _yscale = 100;
        if (nW != null && nW != undefined) {
                __width = nW;
        }
        if (nH != null && nH != undefined) {
                __height = nH;
        }
        if (gotoSetPLY) logger.debug("setSize -> RIGHT CONDITIONS TO SET PLY");
        if (gotoSetPLY) onSetSizeDoPLY();
        ...
}
```

For inheritance reasons here we will define again the onSetSizeDoPLY function that we use in FLVDisplay.as

```
private function onSetSizeDoPLY():Void {
        // overwritten in FLVDsiplay
}
```

An important part of the integration is passing the current played flv path to plyPlayer, this is how we will load all the appropriate data.

6. Open FLVController.as in folder org/flowplayer

First lets define a parameter, we will use it to store the current played path.

var nowPlaying:String;

The following are two functions that resolve and load the flv path (one for rtmp the other for http). On resolving the desired flv, it stores it in the new parameter.

```
public function doLoadClips(clips:Array, protectionCode:String):Void {
        logger.info("FLV Controller -> doLoadClips " + clips);
        for (var i : Number = 0; i < clips.length; i++) {
                var clip:Clip = clips[i];
                connect();
                if (clip.isLive()) {
                        logger.debug("starting a live stream");
                        ns.play(clip.getURL(protectionCode));
                        nowPlaying = clip.getURL(protectionCode);
                } else if (config.getStreamingServerURL() != undefined) {
                        startStreamingClip(clip, protectionCode, i == 0);
                } else {
                        logger.debug("loading video from " + clip.getURL(protectionCode));
                        ns.play(clip.getURL(protectionCode));
                        nowPlaying = clip.getURL(protectionCode);
                }
        }
}
private function startStreamingClip(clip:Clip, protectionCode:String,resetPlayList:Boolean):Void
{
   ...
        nowPlaying = url;
   ...
}
```

Lastly, on receiving flv metadata, we will dispatch an event with the path to PLYController class. Now that we also have movie duration, we are free to complete the plyPlayer initiation, passing it the flv path and duration.

```
function updateNetStreamMetaData(obj:Object):Void {
   ...
   dispatchEventWithObj("onStartPlayingVideo", nowPlaying);
}
```

Last but not least is an additional function in Util class, if called, it centers the plyPlayer movieclip on the stage for optimal visual appearance.

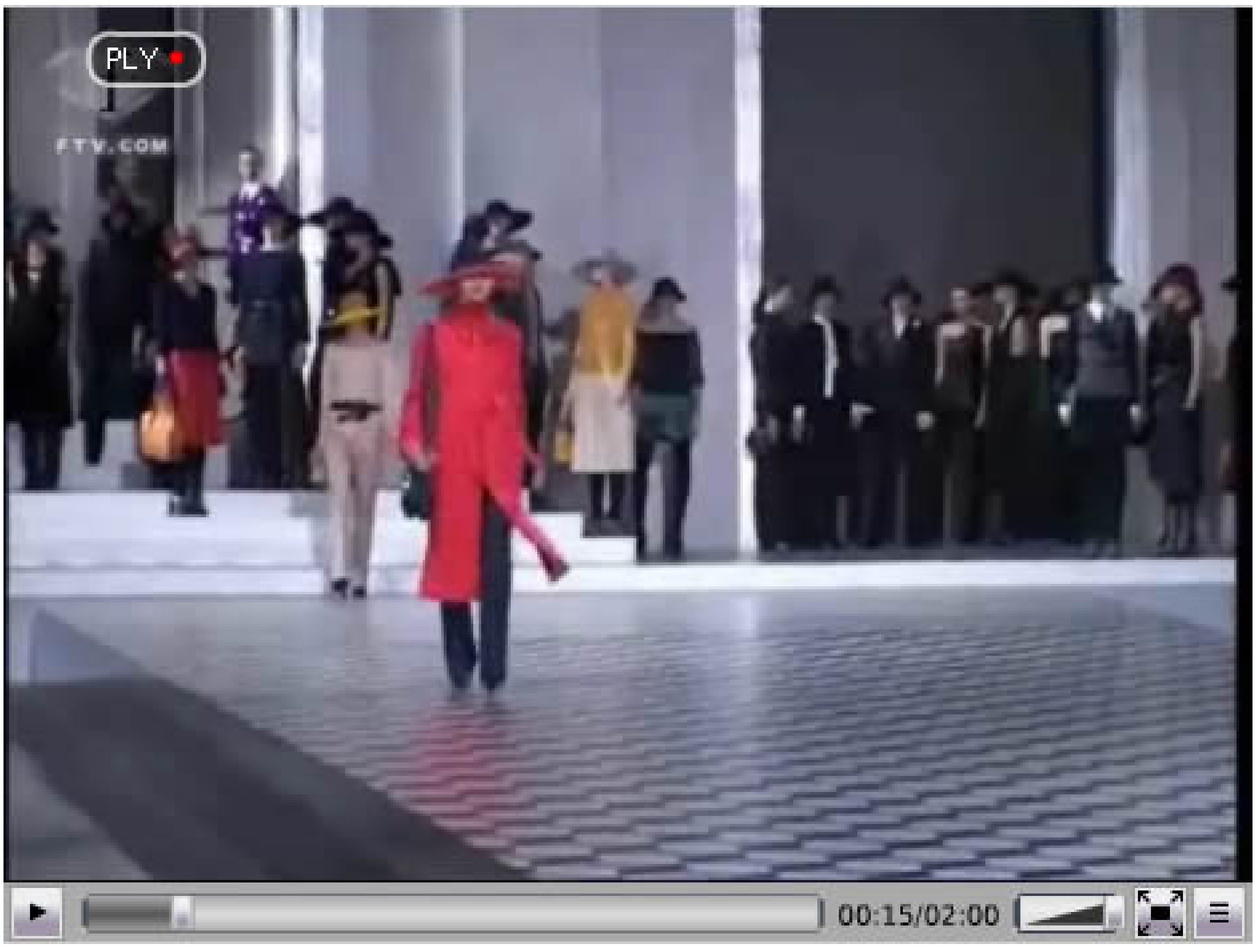
7. Open Util.as in folder org/flowplayer

Copy the following function.

```
public static function centerPLY(clip:MovieClip, areaWidth:Number, areaHeight:Number):Void {
        logger.debug("utils >> centerPLY");
        var clipWidth:Number = clip._width;
        var clipHeight:Number = clip._height;
        if (areaWidth)
                clip._x = (areaWidth / 2) - (clipWidth / 2);
        if (areaHeight)
                clip._y = (areaHeight / 2) - (clipHeight / 2);
}
```

Notes

If the integration was successful you should be able to view the video with PLYmedia's menu.



Sample FlowPlayer with PLYmedia add-on integration:

The following link is to a sample FlowPlayer with the described PLYmedia integration:

http://www.bubbleply.com/sample_FlowPlayer_PLY.zip

PLYmedia information:

In order to add the ability to create, save & modify PLYs or professional subtitles to your site please contact our technical department:

PLYmedia Inc.
Support@PLYmedia.com