Flash OSMF How To Guide

Original Document: Flash_IDE_OSMF_LLNW_Sample.doc by Bob Wohl
Purpose: This document helps developers get started using OSMF with the Limelight Streaming Plugin in Flash CS5 IDE.

Steps to Follow

1. Create a new Flash AS3 .FLA and save it.
2. In the same directory, create a new AS3 class and name it "HelloWorld.as".
3. In the .FLA, open the *Publish* settings by pressing CTRL+Shift+F12.
    a. Select the *Flash* tab and click on the *Settings* button.
    b. In the *Document Class* field, type "HelloWorld".
    c. Click the green arrow to confirm the class's location.
4. In the same location, select the *Library* path.
    a. Include the "OSMF.swc" and the "Controls.swc".
    b. The current version of "OSMF.swc" is located in http://www.opensourcemediaframework.com/.
5. Copy the example code found in this document into the "HelloWorld.as" file and save it.
6. Publish the Flash file by clicking CTRL+Enter.

You now should have a working player that looks like this:

Loading the Plugin with MediaFactory

This section describes the code needed to load the the Limelight Streaming plugin. This is the first step to display Flash media on the Limelight Content Delivery Network.

1. Create a new *DefaultMediaFactory* instance to load the Limelight Streaming plugin.
2. Add two listeners to wait for either a PLUGIN_LOAD or a PLUGIN_LOAD_ERROR event to determine the next step.
    a. If the plugin does not load, check the path to the plugin.
3. Create a new *URLInstance*.
    a. If testing locally, use "file://path/to/plugin.swf" format.

On a successful load of the plugin, the next step is to create the *MediaContainer* and *MediaElement*.

Creating the MediaContainer and MediaElement

Creating the *MediaElement* adds a display to the stage that plays the *MediaFactory*.

1. Create a new *MediaContainer* and add it to the stage.
2. Create a new *StreamingURLResource* to associate the path to the desired stream.
3. For Limelight streams that include an instance name in the URI, set the resource's *urlIncludesFMSApplicationInstance* to true.
4. The Limelight plugin must know the difference between stream types in order to handle the stream. Stream types include: Progressive, LiveStream, MBR, LiveMBR, FLVSeek or MOOVSeek. Refer to OSMF Release Notes for the full list of supported stream types.
5. Create a new *Metadata*.
6. Add the values "llnwStreamType" and the stream type, "Streaming".
7. Add it to the *StreamingURLResource*.
8. Create a new *MediaElement* using *MediaFactory* and assign it to the *StreamingURLResource.*
9. Add the *MediaElement* to the *MediaContainer*.
10. Create a new *MediaPlayer* to play the video.
11. Add a listener to watch for a MEDIA_ERROR event.
12. Assign the *MediaElement* to the *MediaPlayer*.

Controlling the MediaPlayer

This section describes the custom controls for the OSMF streaming media. For complete documentation on available *MediaPlayer* controls, refer to the Adobe MediaPlayer Documentation

Time Display

To update the time display, create a listener for TimeEvent.CURRENT_TIME_CHANGE on the *MediaPlayer*.

The *MediaPlayer* broadcasts the current time change event. The first two values from OSMF are the *currentTime* and *duration* which are used to display time. The second pair of values, *bytesLoaded* and *bytesTotal*, are used to display the buffer.

Seek

To seek with an OSMF *MediaPlayer*, pass the desired time in seconds using the *seek* method.

Example: Seek to 1 minute and 30 seconds:

Volume

To set the volume, pass a value between 0.1 and 1.0 to the *MediaPlayer* instance.

Example: Set volume to 50 percent:

Play, Pause and Stop

To play, pause or stop a stream, use the following *MediaPlayer* methods:

Example source

```
0) {
      volumeConstant = controlBar.volumeValue;
      mutedValue = 0;
    } else {
      mutedValue = volumeConstant;
    }

    changeVolume(mutedValue);
  }

  // full screen button click handler
  private function handleFullScreenClick(event:Event):void
  {
    try {
      var isFullScreen:Boolean = (stage.displayState == StageDisplayState.FULL_SCREEN);
      stage.displayState = isFullScreen ? StageDisplayState.NORMAL :
StageDisplayState.FULL_SCREEN;
    } catch(e:Error) {
      // catch error
    }
  }

  private function restartMedia(event:Event):void
  {
    mediaPlayer.seek(0);
  }

  private function handlePlayClick(event:Event):void
  {
    // this handles the play/pause toggle
    if (mediaPlayer.playing) {
      mediaPlayer.pause();
    } else {
      mediaPlayer.play();
    }
  }

  // Full screen example
  private function handleFullScreen(event:FullScreenEvent):void
  {
    var currentPos:Number = 0;

    if (event.fullScreen) {
      // save values when going to fullScreen
      savedWidth = mediaPlayer.mediaWidth;
      savedHeight = mediaPlayer.mediaHeight;

      // set the size of the video object to the size of the stage
      container.width = stage.stageWidth;
      container.height = stage.stageHeight;
```

```
      //capture width for control bar positioning.
      currentPos = stage.stageWidth;

      controlBar.y = Math.floor(stage.stageHeight - controlBar.height)+1;
      controlBar.x = stage.x;
    } else {
      container.width = savedWidth;
      container.height = savedHeight;
      currentPos = Math.floor(savedWidth);
      controlBar.y = Math.floor(savedHeight - (controlBar.height)+1);
    }
    controlBar.controlBarPlacement(currentPos);
  }
}
]]>
```