MediaProvider Development Guide

Tools

Required

- MediaProvider SDK - MediaProvider SDK
- Flash Player - Download Flash Player
- Flex SDK - Download Flex SDK 3.3
- Ant - Download Ant

Recommended

- Flex Builder - Download Flex Builder 3 **or**
- Flash Builder - Download Flash Builder 4
- Flash Player Debugger version - Download Flash Player Debugger version
- A local web server (http) for testing

Introduction

MediaProviders define a method for playing back media within the JW Player. Prior to JW5, MediaProviders were known as models, and were always compiled directly into the player. While this is still possible, JW5 introduced loadable MediaProviders, which allow the player to add additional playback capabilities without modifying the source of the player itself.

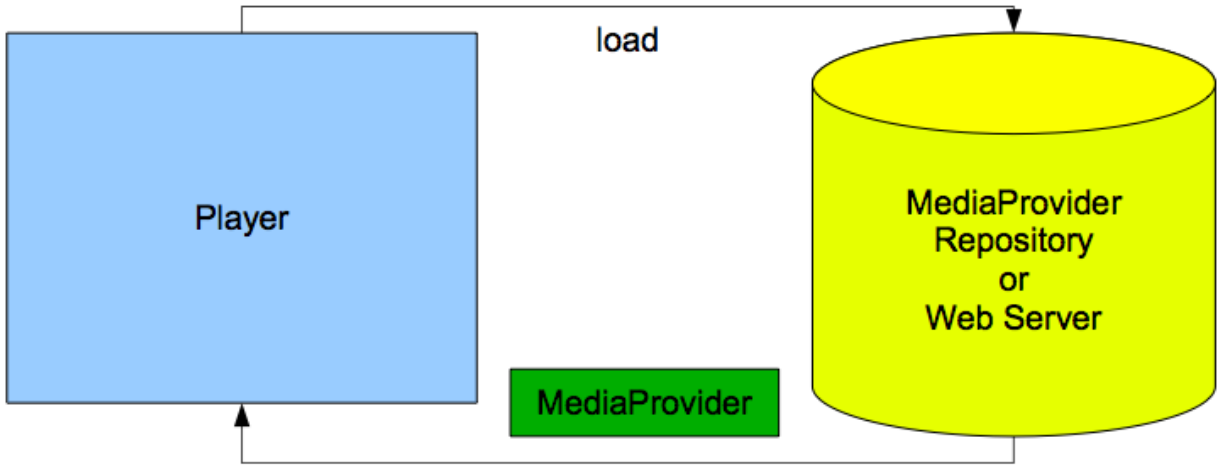

Figure 1 - Player loading a remote MediaProvider

When the player detects a MediaProvider it does not have, it will attempt to load that MediaProvider either from the MediaProvider repository or from any standard Web Server (Figure 1).

Building your MediaProvider

Getting Started

We have made an excellent implementation of a MediaProvider available to you in src/MyMediaProvider.as. Although it's just a skeleton, it has all the methods that are required.

You should rename this file to something a bit more meaningful, like your company's name. Feel free to simply rename the file, or create a new file and copy in the contents of MyMediaProvider.

Be sure to update the name of the class, the name of the constructor function, and the ''MediaProvider'' name, bolded below.

```
public class MyMediaProvider extends MediaProvider {
...
    public function MyMediaProvider() {
        super("myMediaProvider");
        // Your code here
    }
...
}
```

After you rename the file, you'll still have to make a few changes in build/build.properties order to ensure that everything works smoothly. Specifically

- application.package - References the path from the src directory to the MediaProvider, e.g. com/longtailvideo/mediaproviders/
- application.class - The filename of the actionscript file containing your MediaProvider, e.g. MyMediaProvider
- application.name - The filename of the output SWF, e.g. mymediaprovider

Available functions, events, and objects

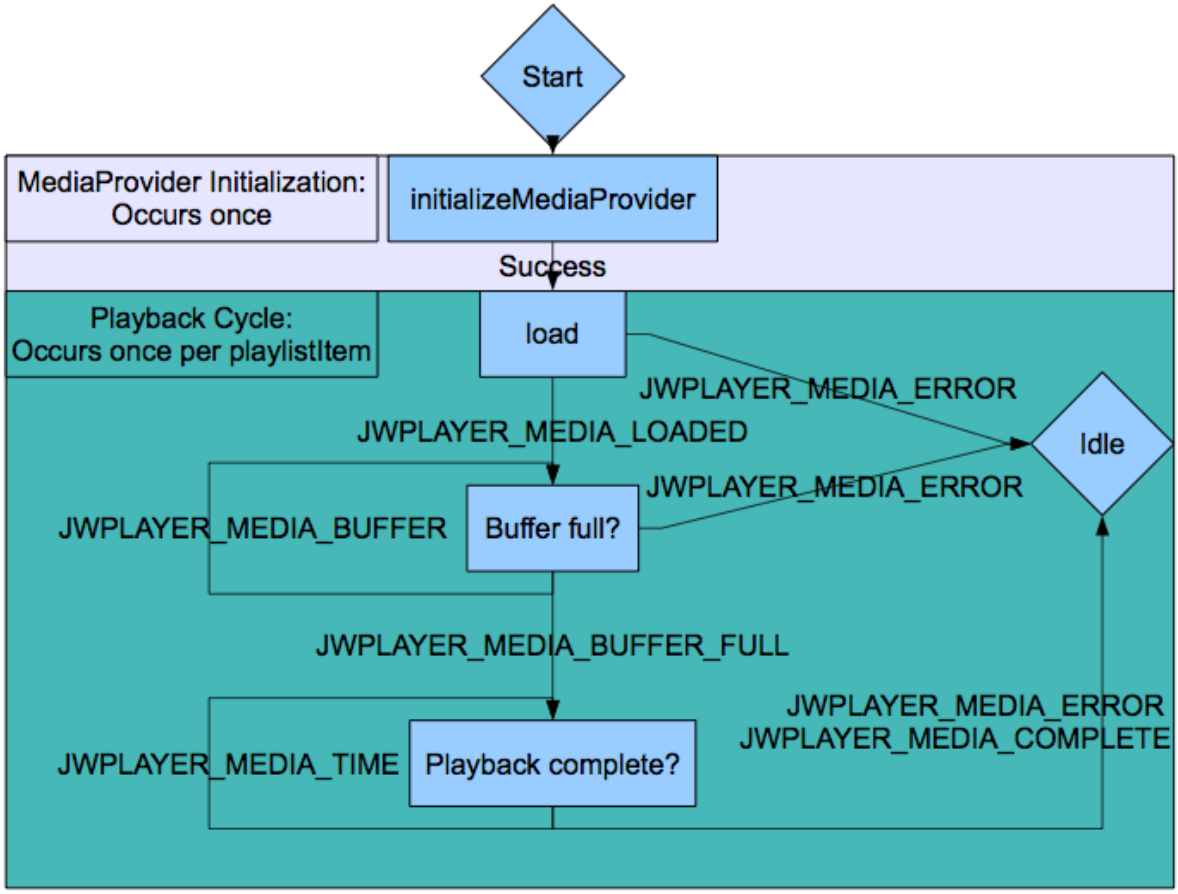

Figure 2 - MediaProvider workflow.

Your MediaProvider cannot access the player itself. Instead, the player will call methods on your MediaProvider, and it will need to appropriately respond by dispatch events. Figure 2 demonstrates the required workflow.

Functions

The functions in MyMediaProvider.as are broken up into three categories: required, recommended, and utility functions.

Required functions are public methods called by the player on the MediaProvider. While required functions are partially implemented by the MediaProvider base class, you will need to add additional logic to make your MediaProvider work, and to distinguish it from other MediaProviders.

Recommended functions are frequently used internally by MediaProviders, but are completely optional.

Utility functions exist for your convenience - they're well implemented by the base class, but feel free to ignore them if they're not helpful.

Required

- initializeMediaProvider - Initializes a MediaProvider after loading. If there is an error, throws an error.
- load - Load a new playlist item.
- play - Resume playback of the item.
- pause - Pause playback of the item.
- seek - Seek to a certain position in the item.
- stop - Stop playing and loading the item.
- setVolume - Change the playback volume of the item.
- mute - Changes the mute state of the item.
- resize - Resizes the display.

Recommended

- buffer - Puts the video into a buffer state.
- complete - Completes video playback.
- error - Dispatches error notifications.

Utility

- setState - Sets the current state to a new state and sends a PlayerStateEvent.
- sendMediaEvent - Sends a MediaEvent, simultaneously setting a property.
- sendBufferEvent - Dispatches buffer change notifications.
- get config - The current config.
- getConfigProperty - Gets a property from the player configuration.
- get media - Gets the graphical representation of the media.
- set media - Sets the graphical representation of the media.

Events

MediaEvents

Your MediaProvider is responsible for dispatching the following events, when appropriate. The sample implementation offers functions that appropriately dispatch these events. **Note** - You **must** dispatch a MediaEvent.JWPLAYER_MEDIA_BUFFER_FULL rather than calling the synchronous method play(). This is to ensure proper functioning of the locking mechanism.

Event	**Description**
MediaEvent.JWPLAYER_MEDIA_BUFFER	Fired when a portion of the current media has been loaded into the buffer.
MediaEvent.JWPLAYER_MEDIA_BUFFER_FULL	Fired when the buffer is full.
MediaEvent.JWPLAYER_MEDIA_ERROR	Fired if an error occurs in the course of media playback.

MediaEvent.JWPLAYER_MEDIA_LOADED	Fired after the MediaProvider has successfully set up a connection to the media.
MediaEvent.JWPLAYER_MEDIA_TIME	Sends the position and duration of the currently playing media.
MediaEvent.JWPLAYER_MEDIA_VOLUME	Fired after a volume change.
MediaEvent.JWPLAYER_MEDIA_COMPLETE	Fired when the currently playing media has completed its playback.

PlayerState Events

Event	Description
PlayerStateEvent.JWPLAYER_PLAYER_STATE	Sent when the playback state has changed.

Your MediaProvider should also notify the player about changes in the media state. The states are enumerated below.

- PlayerState.IDLE - Nothing happening. No playback and no file in memory.
- PlayerState.BUFFERING - Buffering; will start to play when the buffer is full.
- PlayerState.PLAYING - The file is being played back.
- PlayerState.PAUSED - Playback is paused.

Objects

The following objects are defined by MediaProviderBase:

- _item:PlaylistItem - Reference to the currently active PlaylistItem.
- _position:Number - The current position of the stream.
- _width:Number - Width of the display object.
- _height:Number - Height of the display object.

MediaProvider configuration: Passing in your own variables and parameters

If your MediaProvider requires additional configuration parameters, you may pass them in via flashvars in the format 'mymediaprovider.parameter=value'. We recommend that you access these using getConfigProperty(). See Figure 3 for an example.

```
<embed
    src='player.swf'
    width='470'
    height='320'
    bgcolor='#ffffff'
    allowscriptaccess='always'
    allowfullscreen='true'
```

```
 flashvars='file=video.flv&provider=mymediaprovider&mymediaprovider.application=http:
application/' />
```

Playing your Media

Your MediaProvider is responsible for properly rendering its own media. This includes

- loading the visual assets
- displaying the visual assets at the correct time
- sizing / resizing the visual assets appropriately
- deciding when to play and buffer
- handling seek, stop, pause, mute and volume requests correctly

Testing your MediaProvider

Be sure to test your MediaProvider thoroughly before submitting it to LongTail! A thorough test would include testing your MediaProvider with both known good and known bad configurations.

Building the tester

Anytime you want to test your MediaProvider, you should first run the build-debug Ant task. This will compile your SWF (in debug mode) and copy it in our testing framework.

Using the tester

To test your MediaProvider, simply open bin-debug/index.html in any web browser. We recommend that you access this via a local web server because of restrictions in the Flash security model.

Modifying the tester

If you'd like to set up a series of tests configurations, simply add additional examples to debug-template/files/settings.js.

Preparing your MediaProvider for distribution

To build a SWF that is ready for distribution, simply run the build-release Ant task. The SWF will appear in bin-release.

Submitting your MediaProvider

Before submitting your MediaProvider, please be sure that

- you dispatch a MediaEvent.JWPLAYER_MEDIA_BUFFER_FULL rather than calling the synchronous method play().

- there are no references to

    - RootReference
    - root
    - stage
    - parent
    - ExternalInterface

- your MediaProvider is tested and working as expected using the provided tester.

- all configuration parameters are properly reflected in debug-template/files/settings.js.

Submission contents

In your submission, you should include:

- Everything necessary to compile your MediaProvider, including
    - Your source code
    - Libraries
    - Build scripts

- A working test example with all necessary configuration parameters.

We recommend that you simply zip up the your modified SDK and submit that, as it will include all of the necessary elements.

Approval process (QA)

The MediaProvider approval process is quite straight-forward.

1. We will recompile your MediaProvider and test it using your supplied testing page.
2. We put the MediaProvider into our development environment and test it from within our testing framework.
3. We will release your MediaProvider and make it available to all sites.

Modification

Should you need to make an update to your MediaProvider, it will generally need to go through the same QA process, so please be sure to include all files listed above.