Azure Media Player Video Editor (AMVE) Implementation Guide

Prerequisites

Azure Media Player

The Azure Media Video Editor (AMVE) is a plugin for The Azure Media Player (AMP) and as such requires AMP to work. At the time of this release, AMVE has been tested with AMP version 1.7.4. It is quite possible that AMVE will work with later versions of AMP however, there is no guarantee.

For more information on how to add Azure Media Player to your project, please see the Azure Media Player documentation, Azure Media Player demo and Azure Media Player samples.

How to Implement

1. Add Necessary Files

To use the Azure Media Video Editor(AMVE) you'll first need to add the necessary CSS and JavaScript files to your application. AMVE expects the following files to operate:

1.1 CSS Files

- amve.min.css

1.2 Font Files

All font files must be placed in fonts directory relative to amve.min.css

- amve.eot
- amve.svg
- amve.ttf
- amve.woff

1.3 JavaScript Files

- amve.min.js

2. Add CSS and JS References to your Page

Next we'll need to add the references to the JavaScript and CSS files to your page.

2.1 JavaScript References

Add the following JavaScript reference in your page:

```
<script src="amve.min.js"></script>
```

2.2 CSS References

Add the following CSS reference to your page after the AMP CSS reference(s) like so:

```
<link rel="stylesheet" type="text/css" href="//amp.azure.net/libs/amp/1.7.4/skins/amp-
default/azuremediaplayer.min.css" />
```

```
<link rel="stylesheet" type="text/css" href="amve.min.css" />
```

3. Add the AMVE Plugin Registration to your AMP Configuration.

With the requisite references in place, we can now add the plugin registration to our AMP configuration like so:

3.1 AMVE Plugin Registration

```
    var myOptions = {
        'nativeControlsForTouch': false,
        autoplay: true,
        controls: true,
        poster: '',
        flashSS: {
            swf: 'tech-wrappers/Players/osmf/StrobeMediaPlayback.2.0.swf',
            plugin: 'tech-wrappers/Players/osmf/MSAdaptiveStreamingPlugin-osmf2.0.swf'
        },
        silverlightSS: {
            xap: 'tech-wrappers/Players/smf/SmoothStreamingPlayer.xap'
        },
        plugins: {
            AMVE: { containerId: 'amve', customMetadataContainerId: 'custommetadata',
clipdataCallback: onClipdataCallback, keyboardShortcutConfig: keyboadShortcutConfig }
        }
    };

    var myPlayer = amp('vid1', myOptions);
```

4. Add an Element to your Page hold the Editor

AMVE needs a container element to hold the editor UI DOM that wraps AMP. This element should wrap around the element used for AMP like so:

4.1 AMVE Container Example

```
<div id="amve">
    <video id="vid1" class="azuremediaplayer amp-default-skin amp-big-play-centered">
        <p class="amp-no-js">To view this video please enable JavaScript, and consider
upgrading to a web browser that supports HTML5 video</p>
    </video>
</div>
```

5. Optional: Custom Submission Dialog UI - Add a DOM Fragment to Your Page

You can add a DOM fragment containing custom UI that will be pulled into the submission dialog. To do so, add the DOM fragment as a child of the container element described in section 4. The fragment will be placed into the proper DOM hierarchy by AMVE after AMVE loads. To that end, it is best to set CSS visibility of the fragment to hidden to start. The id for the element is passed to AMVE as the customMetadataContainerId parameter which is further explained in section 8. An example of a custom submission UI dialog fragment is as follows:

5.1 Custom Submission Dialog UI Example

```
<div id="amve">
    <video id="vid1" class="azuremediaplayer amp-default-skin amp-big-play-centered">
        <p class="amp-no-js">To view this video please enable JavaScript, and consider
upgrading to a web browser that supports HTML5 video</p>
    </video>
    <div id="custommetadata">
        <hr />
        <label>Test1: </label>&nbsp;<input id="test1" type="text" value="Test 1" /><br>
        <label>Test2: </label>&nbsp;<input id="test2" type="text" value="Test 2" /><br>
        <label>Test3: </label>&nbsp;<input id="test3" type="text" value="Test 3" /><br>
        <label>Test4: </label>&nbsp;<input id="test4" type="text" value="Test 4" />
        <hr/>
        <label>Color: </label><br/>
        <input type="radio" name="colorchoice" value="red" checked> Red<br>
        <input type="radio" name="colorchoice" value="blue"> Blue<br>
        <input type="radio" name="colorchoice" value="green"> Green
        <hr />
        <label>Food: </label><br />
        <input type="checkbox" name="foodchoices" value="breakfast"> Breakfast<br>
        <input type="checkbox" name="foodchoices" value="lunch"> Lunch
    </div>
</div>
```

6. Clip Submission Callback Function

Create a function with the signature onClipdataCallback(clipData) that will be called when a clip has been created. A reference to the function will be passed to AMVE as the clipDataCallback parameter described in section 8. The clipData object passed to the callback has all of the data necessary to create a clip using AMS. The properties of the clipData object are as follows:

6.1 clipData Object Properties

Property	Description
src	The src uri of the stream being clipped.
markIn	The time in seconds for the beginning of the clip.
markOut	The time in seconds for the end of the clip. If the markOut is zero it is assumed that the stream should be trimmed from the start of the stream to the markIn point.
title	A title for the clip.
description	A Description for the clip.
thumbnail	The base64 encoded image URI for the chosen thumbnail.

6.2 clipDataCallback Function Example

```
function onClipdataCallback(clipData) {
    if (clipData) {
        var dataTxt = 'data.src: ' + clipData.src +
            '\ndata.markin: ' + formatTime(clipData.markIn) +
            '\ndata.markout: ' + formatTime(clipData.markOut) +
            '\ndata.title: ' + clipData.title +
            '\ndata.description: ' + clipData.description +
            '\ndata.thumbnail: ' + (clipData.thumbnail ? clipData.thumbnail.dataUrl :
null);
        alert(dataTxt);
    }
}
```

6.3 Formatting clipData markIn and markOut times for AMS

For AMS, you should format time per the AMS clip job specification. At the time of this documentation the following function will format time appropriately:

```
function formatTime(time) {
    var d = Math.floor(time / 86400);
    time -= (d * 86400);
    var h = Math.floor(time / 3600) % 24;
    time -= (h * 3600);
    var m = Math.floor(time / 60) % 60;
    time -= (m * 60);
    var s = Math.floor(time);
    time -= s;
    var hs = Math.floor(time * 100);

    var returnVal = '';

    returnVal = returnVal + (d > 0) ? d + '.' : '';
    returnVal = returnVal + (h < 10 ? ('0' + h) : ('' + h));
    returnVal = returnVal + ':' + (m < 10 ? ('0' + m) : ('' + m));
    returnVal = returnVal + ':' + (s < 10 ? ('0' + s) : ('' + s));
    if (hs > 0) {
        returnVal = returnVal + '.' + hs;
    }

    return returnVal;
}
```

7. Keyboard Shortcuts

AMVE has pre-configured shortcut configurations that map to Adobe Premier Pro or Avid. You can also create your own by creating an instance of AMVE.KeyboardShortcutConfig and configuring the shortcut mappings like below. The completed KeyboardShortcutConfig instance would then be passed as a parameter to AMVE as described in section 8. An example of using the Adobe Premier Pro, Avid or custom KeyboardShortcutConfig is as follows:

7.1 KeyboardShortcutConfig Example

```
//Adobe Premier Pro keyboard shorcut config
var keyboadShortcutConfig = new AMVE.AdobePremierProShortcutConfig();

//Avid keyboard shorcut config
keyboadShortcutConfig = new AMVE.AvidShortcutConfig();

// Custom keyboard shorcut config
keyboadShortcutConfig = new AMVE.KeyboardShortcutConfig();
keyboadShortcutConfig.playPauseShortcut = 'space';
keyboadShortcutConfig.markInShortcut = 'i';
keyboadShortcutConfig.markOutShortcut = 'o';
keyboadShortcutConfig.goToInShortcut = 'shift+i';
keyboadShortcutConfig.goToOutShortcut = 'shift+o';
keyboadShortcutConfig.clearInShortcut = 'ctrl+shift+i';
keyboadShortcutConfig.clearOutShortcut = 'ctrl+shift+o';
keyboadShortcutConfig.clearBothShortcut = 'ctrl+shift+x';
keyboadShortcutConfig.playInToOutShortcut = 'ctrl+shift+space';
keyboadShortcutConfig.playToOutShortcut = 'ctrl+space';
keyboadShortcutConfig.ffwdShortcut = 'ctrl+right';
keyboadShortcutConfig.rwdShortcut = 'ctrl+left';
keyboadShortcutConfig.backOneFrameShortcut = 'left';
keyboadShortcutConfig.backFiveFramesShortcut = 'shift+left';
keyboadShortcutConfig.backOneSecondShortcut = 'alt+shift+left';
keyboadShortcutConfig.backOneGopShortcut = 'ctrl+shift+left';
keyboadShortcutConfig.fwdOneFrameShortcut = 'right';
keyboadShortcutConfig.fwdFiveFramesShortcut = 'shift+right';
keyboadShortcutConfig.fwdOneSecondShortcut = 'alt+shift+right';
keyboadShortcutConfig.fwdOneGopShortcut = 'ctrl+shift+right';
keyboadShortcutConfig.markInOneFrameShortcut = 'alt+,';
keyboadShortcutConfig.markInFiveFramesShortcut = 'alt+shift+,';
keyboadShortcutConfig.markOutOneFrameShortcut = 'alt+.';
keyboadShortcutConfig.markOutFiveFramesShortcut = 'alt+shift+.';
keyboadShortcutConfig.undoShortcut = 'ctrl+z';
keyboadShortcutConfig.redoShortcut = 'ctrl+y';
keyboadShortcutConfig.exportShortcut = 's';
keyboadShortcutConfig.changeModeVirtualShortcut = 'shift+v';
keyboadShortcutConfig.changeModeRenderedShortcut = 'shift+r';
keyboadShortcutConfig.changeModeTrimShortcut = 'shift+t';
```

8. AMVE Plugin Parameters

There are a number of parameters that can be passed to AMVE via the plugin registration. They are all passed via the plugin options parameter in the plugin registration like so:

8.1 AMVE Plugin Parameters – Passing through AMP Registration

```
plugins: {
    AMVE: { containerId: 'amve', customMetadataContainerId: 'custommetadata',
clipdataCallback: onClipdataCallback, keyboardShortcutConfig: keyboadShortcutConfig }
}
```

8.2 AMVE Parameters

The possible configuration parameters, some required some optional, are as follows:

Parameter	Description	Required or Optional
containerId	The id of the container element described in section 4.	Required
customMetadataContainerId	The id for the custom submission dialog UI fragment explained in section 5.	Optional
clipdataCallback	The callback function called upon clip submission as described in section 6.	Required
keyboardShortcutConfig	The AMVE.KeyboardShorcutConfig instance described in section 7.	Optional

9 CSS Customization

The style of the editor, display and position of elements can be customized at will by overriding the applied CSS styles. With this said, it should be noted that manipulating the CSS properties of the editor elements could cause issues in the function of the editor.

As a guide the CSS class names and style rules generally follow the hierarchy of the components of the editor. The best way to customize the editor is by looking at the rendered and styled DOM using your favorite HTML dev tools like those built into your favorite browser.