Stand Alone Video Player

Last Modified on 06/16/2022 11:06 am EDT

Stand Alone Video Player

The SpringServe Stand Alone Player is an on-demand player that will place the video player within a predefined and created div provided by a publisher.

If there is a successful video fetched and played, the video will play within the created div. If there is not a successful video fetched an attempt will be made to place a banner ad within the div (this is an option so we can turn off display backfill if requested). If either of the actions fails there is an optional callback allowed when the newStandAlonePlayer method is invoked that will send a failure or error message. This can be actioned on to remove the div or call a different type of product in its place.

Publisher Actions Required

1. A publisher must add a div to their page with a **unique id** that only appears once per page view. The div will be provided to you with your ad tags text file.
    a. e.g.

```
<div id="video_placement_728x90"></div>
```

2. The div with the id video_placement_728x90 must also have a predetermined width which can be defined via a standard css style sheet or defined as an inline style. **Please note the minimum width for this product is 400 pixels.**

```
<div id="video_placement_728x90" style="width: 400px;"> </div>
```

```
#video_placement_728x90 {
  width: 400px;
}
```

3. In order to activate the stand alone player, the publisher must execute the freestar.newStandAlonePlayer method with an **object** as the first and only argument. This code will also be provided in your ad tags text file.
    a. e.g.

```
<script>
 // without optional callback
 window.freestar.queue.push(function(){
   freestar.newStandAlonePlayer({ placementName: 'video_placement_728x90' });
 });
 // with optional callback
 function standAloneCallback(err, res) {
  // do something  9  }
 window.freestar.queue.push(function(){
   freestar.newStandAlonePlayer({ placementName: 'video_placement_728x90', callback: standAloneCall
back});
 });
</script>
```

4. Object keys for the newStandAlonePlayer method object argument.

Key	Type	Example	Required	Note
placementName	String	'video_placement_728x90'	True	Must match created placement name within publishers fsdata

callback	Function	`function myCallBack(error, success) {`
		`  if (error) {`
		`  // do something`
		`  } else if (result) {`
		`  // do something`
		`  }`
		`}`
			False	The callback method will return either an error Boolean (first parameter) or a result Object (second parameter).
				
				The result Object may have multiple keys values that indicate the current place in the lifecycle for the Stand Alone Video Player.

*the result object is available in **Pubfig 4.10 +** prior to this the result parameter is a Boolean.

Stand Alone Video Player Callback Results Object Keys

Key	Value	Note
videoAdStarted	true	Video fetched and started correctly.
videoAdStarted	false	Video fetch failed, or was not started correctly. An attempt at filling the video with banner backfill display ad will begin.
displayAdServed	true	Banner ad has rendered, and the Process is terminated. The completed key will be sent with this key as well since the process will be terminated. *Keep in mind that the banner ad would have just been rendered at this point, and to allow the ad to stay in place for viewability.*
displayAdServed	false	Banner ad failed to render, and the Process is terminated. The completed key will be sent with this key as well since the process will be terminated.
videoAdSkipped	true	Video ad has been skipped, and the process is terminated.
completed	true	Stand Alone Video Ad process is terminated. This Key and value will be sent when either the video ad or banner ad has been completed.

- Please note that the same AdStopped event from SpringServe occurs when the user closes the video ad and when the video completes its lifecycle and closes itself. The AdStopped event is sent through the callback with the completed key with a value of true.

Stand Alone Video Player Callback Example

```
function myCallBack(error, result) {
  if (error) {
    // do something
  } else if (result) {
    if (result.videoAdStarted === true) {
      console.log('video ad is playing')
    } else if (result.videoAdStarted === false) {
      console.log('video did not fetch or play attempt at a banner ad will occur')
    } else if (result.displayAdServed) {
      console.log('banner ad has rendered process is terminated')
      // allow time for viewability measurements
    } else if (result.videoAdSkipped) {
      console.log('video ad was skipped process is terminated')
      // do something
    } else if (result.completed) {
      console.log('stand alone video process has been completed, and the process is terminated.')
      // do something
    }
  }
}
```