ADMARU VIDEO PLAYER

Ads Player Configuration


Ad
maru

Player support Values/Parameters.

Support Video format – HTML 5 Videos (.mp4,.mov,WebM,.ogv),youtube,dailymotion,vimeo-oembed iframe

Support Video Stream – hls,mpeg dash,webrtc, rtmp with rtmp java script plugin,p2p stream

Responsive - works with any screen size

Monetization - make money from your videos with any ads network –prebid,header bidding Up to VPAID 2 & Vast 4.2

Streaming - support for hls.js, Shaka and dash.js streaming playback

API - toggle playback, volume, seeking, and more through a standardized API

Events - no messing around with Vimeo and YouTube APIs, all events are standardized across formats

IMA Support Parameters:

Settings	Type	Description
adLabel	string	Replaces the "Advertisement" text in the ad label. Added for multilingual UI support.
adLabelNofN	string	Replaces the "of" text in the ad label (e.g. ... (1 of 2) ...). Added for multilingual UI support.
adTagUrl	string	A URL which returns a VAST, VMAP or ad rules response. This will override adsResponse.
adsRenderingSettings	object	JSON object with ads rendering settings as defined in the IMA SDK Docs(1).
adsResponse	string	The VAST, VMAP, or ad rules response to use in lieu of fetching one an ad tag. This is overridden if adTagUrl is set.

Settings	Type	Description
adsRequest	object	JSON object with ads request properties defined in the IMA SDK Docs(2). Properties set here that can also be provided elsewhere (e.g. adTagUrl) will override those other settings.
autoPlayAdBreaks	boolean	Whether or not to automatically play VMAP or ad rules ad breaks. Defaults to true.
deprecated adWillPlayMuted	boolean	Notifies the SDK whether the player intends to start ad while muted. Changing this setting will have no impact on ad playback. Defaults to false.
contribAdsSettings	object	Additional settings to be passed to the contrib-ads plugin(3) used by this IMA plugin.
debug	boolean	True to load the debug version of the plugin, false to load the non-debug version. Defaults to false.
disableAdControls	boolean	True to hide the ad controls(play/pause, volume, and fullscreen buttons) during ad playback. Defaults to false.
disableCustomPlaybackForIOS10Plus	boolean	Sets whether to disable custom playback on iOS 10+ browsers. If true, ads will play inline if the content video is inline. Defaults to false.
disableFlashAds	boolean	True to disable Flash ads - Flash ads will be considered an unsupported ad type. Defaults to false.
featureFlags	object	Sets IMA SDK feature flags.
forceNonLinearFullSlot	boolean	True to force non-linear AdSense ads to render as linear fullslot. If set, the content video will be paused and the non-linear text or image ad will be rendered as fullslot. The content video will resume once the ad has been skipped or closed.
id	string	**DEPRECATED** as of v.1.5.0, no longer used or required.
locale	string	Locale for ad localization. The supported locale codes can be found in Localizing for Language and Locale
nonLinearHeight	number	Desired height for non-linear ads. Defaults to 1/3 player height.

Settings	Type	Description
nonLinearWidth	number	Desired width of non-linear ads. Defaults to player width.
numRedirects	number	Maximum number of VAST redirects before the subsequent redirects will be denied and the ad load aborted. The number of redirects directly affects latency and thus user experience. This applies to all VAST wrapper ads.
omidVendorAccess	object	Sets and enables the Open Measurement SDK(4). Accepts an object with keys corresponding to OMID verification vendors(5). The value pair for each key should be the OMID access mode(6) associated with that vendor.
ppid	string	Sets the publisher provided ID
preventLateAdStart	boolean	Prevent ads from starting after the content has started if an adtimeout occurred (preroll, midroll, postroll). The default value is false
sessionId	string	Sets the session ID
showControlsForJSAds	boolean	Whether or not to show the control bar for VPAID JavaScript ads. Defaults to true.
showCountdown	boolean	Whether or not to show the ad countdown timer. Defaults to true.
vastLoadTimeout	number	Override for default VAST load timeout in milliseconds for a single wrapper. The default timeout is 5000ms.
vpaidAllowed	boolean	**DEPRECATED**, please use vpaidMode.
vpaidMode	VpaidMode(5)	VPAID Mode. Defaults to ENABLED. This setting,overrides vpaidAllowed.

(1) AdsRenderingSettings
(2) AdsRequest
(3) contrib-ads plugin
(4) Open Measurement SDK guide
(5) OmidVerificationVendor

(6) OmidAccessMode
(7) ImaSdkSettings.setVpaidMode

IMA Plugin Ad Events

The IMA Plugin fires events that can be listened for. Ad lifecycle events can be listened for by following our Advanced Example. Other events are emited from the videojs player. Please see the below example to set up listeners for these events.

```
this.player = videojs('content_video');

this.player.on('ads-manager', function(response){
  var adsManager = response.adsManager;
  // Your code in response to the `ads-manager` event.
})
```

Below are the events added by the videojs-ima plugin to the videojs player.

Event	Event String	Payload
Ad Started	'ads-ad-started'	none
Ads Manager	'ads-manager'	google.ima.AdsManager
Ads Loader	'ads-loader'	google.ima.AdsLoader
Ads Request	'ads-request'	google.ima.AdsRequest

Disable automatic ad break playback

In some circumstances you may want to prevent the SDK from playing ad breaks until you're ready for them. In this scenario, you can disable automatic playback of ad breaks in favor of letting the SDK know when you're ready for an ad break to play. To do so:

1. Set `autoPlayAdBreaks` to false in the initial options.
2. Provide an ad break ready listener via `setAdBreakReadyListener`.
3. Call `player.ima.playAdBreak()` in your ad break ready listener when you're ready to play the ads.

Logo Configuration

Property	Attributes	Type	Default value	Description
image	**Required**	**String**		**The URL to the logo image.**
url	Optional	String		A url to be linked to from the logo. If the user clicks the logo the link will open in a new window.
position	Optional	String	"top-right"	The location to place the logo (top-left, top-right, bottom-left, or bottom-right).
offsetH	Optional	Number	0	Horizontal offset (px) from the edge of the video.
offsetV	Optional	Number	0	Vertical offset (px) from the edge of the video.
width	Optional	Number		The width of the logo image (px). If not specified, it will be the width of the original image.
height	Optional	Number		The height of the logo image (px). If not specified, it will be the height of the original image.
padding	Optional	Number	5	Padding around the logo image (px).
fadeDelay	Optional	Number, Null	5000	Time until fade-out begins (msec). If `null` is specified, automatic fade-out is not performed.
hideOnReady	Optional	Boolean	false	Do not show the logo image when the player is ready.
opacity	Optional	Boolean	1	The opacity of the logo (from `[0, 1]`). If not specified, it will default to 1.

Methods

You can also manually show / hide the logo image at any time.

```
// To show the logo image on the player's play event:
player.on('play', () => {
  player.logo().show();
});
```

Method	Description
show()	Show the logo image
hide()	Hide the logo image

Prebid Out Stream Render Sample Code:

Demo 1:

Place this code in the page header.

```
<script>
    var pbjs = pbjs || {};
    pbjs.que = pbjs.que || [];

    function callANRenderer(bid, ad) {
      const adResponse = {
          ad: {
              video: {
                  content: ad,
                  player_width: 640,
                  player_height: 480,
              }
          }
      }

      bid.renderer.push(() => {
          window.ANOutstreamVideo.renderAd({
              targetId: bid.adUnitCode,
              adResponse,
          });
      });
    }

    function render(bid) {
        let ad = bid.ad || bid.vastXml;

        if (ad) {
            callANRenderer(bid, ad)
        } else {
            if (bid.vastUrl) {
              (async () => {
                  ad = await fetch(resp).then(resp => resp.text());

                  if (typeof ad === 'string') {
                      callANRenderer(bid, ad);
                  } else {
                      console.log('Invalid VAST');
                  }
              })();
            } else {
                console.log('Invalid ad');
            }
        }
    }

    const adUnits = [{
        code: 'admaru',
```

```
        mediaTypes: {
            video: {
                context: 'outstream',
                playerSize: [640, 480],
                mimes: ['video/mp4'],
                protocols: [1, 2, 3, 4, 5, 6, 7, 8],
                playbackmethod: [2],
                skip: 1,
                renderer: {
                    render,
                    url: " outstreamrender.js"
                },
            }
        },
        bids: [{
            bidder: 'appnexus',
            params: {
                placementId: 13232385
            }
        }]
    }];

    pbjs.que.push(function() {
        pbjs.addAdUnits(adUnits);
        pbjs.requestBids({
            timeout: 1000,
            bidsBackHandler: function(bids) {
                const highestCpmBids = pbjs.getHighestCpmBids('admaru');
                pbjs.renderAd(document, highestCpmBids[0].adId);
            }
        });
    });
</script>
```

Place this code in the page body.

```
<div id='admaru'>
  <p>Test Outstream Video Ad Demo</p>
</div>
```

playback_method:

```
    'unknown': 0,
    'auto_play_sound_on': 1,
    'auto_play_sound_off': 2,
    'click_to_play': 3,
    'mouse_over': 4,
    'auto_play_sound_unknown': 5
```

context:

```
    'unknown': 0,
    'pre_roll': 1,
    'mid_roll': 2,
    'post_roll': 3,
    'outstream': 4,
    'in-banner': 5
```

Demo 2:

```
<script>
  var pbjs = pbjs || {};
  pbjs.que = pbjs.que || [];

  function render(bid) {
    const videoContent = document.getElementById('contentElement');
    const adContainer = document.getElementById('adContainer');
    const adDisplayContainer = new google.ima.AdDisplayContainer(adContainer,
videoContent);

    const adsRequest = new google.ima.AdsRequest();
    adsRequest.adTagUrl = '';
    adsRequest.adsResponse = bid.vastXml;

    adsLoader = new google.ima.AdsLoader(adDisplayContainer);
    adsLoader.requestAds(adsRequest);


adsLoader.addEventListener(google.ima.AdsManagerLoadedEvent.Type.ADS_MANAGER_LOADED,
(adsManagerLoadedEvent) => {
      const adsRenderingSettings = new google.ima.AdsRenderingSettings();
      adsRenderingSettings.restoreCustomPlaybackStateOnAdBreakComplete = true;
      /* videoContent should be set to the video DOM element. */
      const adsManager = adsManagerLoadedEvent.getAdsManager(videoContent,
adsRenderingSettings);

      /* Ad event listeners */
      adsManager.addEventListener(google.ima.AdEvent.Type.LOADED, () =>
videoContent.play());
      adsManager.addEventListener(google.ima.AdEvent.Type.CONTENT_PAUSE_REQUESTED, ()
=> videoContent.pause());
```

```
      adsManager.addEventListener(google.ima.AdEvent.Type.CONTENT_RESUME_REQUESTED,
() => videoContent.play());

      /* Play the ad */
      videoContent.load();
      adDisplayContainer.initialize();
      adsManager.init(640, 360, google.ima.ViewMode.NORMAL);
      adsManager.start();
    }, false);
  }

  /* Prebid video ad unit */
  const videoAdUnit = {
    code: 'adContainer',
    renderer: {
      render,
      url: 'https://imasdk.googleapis.com/js/sdkloader/ima3.js'
    },
    mediaTypes: {
      video: {
        context: 'outstream',
        playerSize: [640, 480],
        mimes: ['video/mp4'],
        protocols: [1, 2, 3, 4, 5, 6, 7, 8],
        playbackmethod: [2],
        skip: 1,
      }
    },
    bids: [{
      bidder: 'appnexus',
      params: {
        placementId: 13232385
      }
    }]
  };

  pbjs.que.push(() => {
    pbjs.addAdUnits(videoAdUnit);

    pbjs.requestBids({
      bidsBackHandler: (bids) => {
        const highestCpmBids = pbjs.getHighestCpmBids('adContainer');
        pbjs.renderAd(document, highestCpmBids[0].adId);
      }
    });
  });
</script>
```