Video Integration with VAST

The Adhese VAST JS library is designed to make it easy to integrate VAST ads into video players. It includes cross-domain safe methods for requesting ads from your Adhese account and convenient methods for playing and tracking video ads. However, it is not a player on its own and does not insert anything into the DOM.

1. Load the JavaScript file:
   ```
   <script type="text/javascript" src="adhese-vast.js"></script>
   ```
2. Create an `AdheseVastWrapper` object:
   ```
   var wrapper = newAdheseVastWrapper();
   ```
3. Register a listener for the ADS_LOADED event fired by `AdheseVastWrapper` The first parameter is the event name, and the second is the name of your callback function. This function will be called when the wrapper is ready to handle your ad request.
   ```
   wrapper.addEventListener("ADS_LOADED", yourCallBackFunction);
   ```
4. Initiate an ad request by providing the host of your Adhese account and the slot path and target parameters you want to request.
   ```
   wrapper.requestAds("http://ads.demo.adhese.com", "_test_", ["preroll", "postroll"]);
   ```
5. Once the request is finished, `AdheseVastWrapper` will fire the ADS_LOADED event, and your callback function will be called. From then on, you can access several properties of the wrapper object to get info on the ads.

This is a complete example:

```
<html>
<head>
    <script type="text/javascript"src="dist/adhese-vast.min.js">
        </script>
</head>
<body>
    <!-- create a player and info pane container -->
    <h1 id="info">
        </h1>
        <div id="player">
            </div>
            <script type="text/javascript">
```

```
                // just for completing the example, the content that will be shown after the
example ad
                var actualContentSrc = "http://media.w3.org/2010/05/bunny/movie.mp4";

                // get reference to the container elements
                var playerContainer = document.getElementById("player");
                var infoContainer = document.getElementById("info");

                var a = new AdheseVastWrapper(true);
                a.init();
                a.addEventListener("ADS_LOADED", adsAreLoaded);
                a.requestAds("http://ads.demo.adhese.com", "_sdk_example_", ["preroll"]);

                function adsAreLoaded() {
                console.log("adsAreLoaded")

                // if has preroll, show it
                if (a.hasAd("preroll")) {
                // display duration
                infoContainer.innerHTML = "ad takes " + a.getDuration("preroll") + " time,
stay tuned";

                // create source element for video
                var adSrc = document.createElement("source");
                adSrc.src = a.getMediafile("preroll","video/mp4");
                adSrc.type = "video/mp4";

                // create desired video element
                var adPlayer = document.createElement("video");

                adPlayer.width = 640;
                adPlayer.height = 480;
                adPlayer.autoplay = "true";

                // if using a flash based player: make sure adPlayer is a reference to the
flash object and
                allowScripAccess is true
                // event names will be different in flash as well, depending on how video
playback is implemented

                // attach to timeupdate event for passing the currentTime, this allows adhese
to track the actual
                viewing of the ad
                adPlayer.addEventListener("timeupdate", function() { a.timeupdate("preroll",
adPlayer.currentTime); },
                true);
                // clicks on video player should be sent to adhese for handling and reporting
                adPlayer.addEventListener("click", function() { a.clicked("preroll",
adPlayer.currentTime); }, true);
                // when playing has ended, tell and adhese and than continue to showing
content
                adPlayer.addEventListener("ended", function() { a.ended("preroll",
adPlayer.currentTime);
                showActualContentAfterPreroll(); }, true);
                //add the source to the video element
                adPlayer.appendChild(adSrc);

                // ad the video element to the player container
                playerContainer.appendChild(adPlayer);
                }

                }

                function showActualContentAfterPreroll() {
                // here comes the code to start your content after the ad
                infoContainer.innerHTML = "Feature film starting. Enjoy!";
                playerContainer.innerHTML = '
                <video
id="video"controls=""autoplay=""width="640"height="480"preload="none"poster="http:
                    //media.w3.org/2010/05/bunny/poster.png">
                    <source id="mp4"src="http:
//media.w3.org/2010/05/bunny/movie.mp4"type="video/mp4">
                        <source id="ogv"src="http:
//media.w3.org/2010/05/bunny/movie.ogv"type="video/ogg"></video>';
                            }
            </script>
</body>
</html>
```

Ad Pods (VAST & JSON)

About Ad Pods

An Ad Pod is a sequenced group of ads that allows publishers to display multiple ads within a single ad placement.

In the case of VAST, the ads can play before, during, or after the video content. In this case, ad pods are very similar to TV commercial breaks.

Pods are populated by Advar templates. We support two formats: VAST XML and JSON.

Request

Fill in the necessary brackets in the following ad pod request URL and paste it into your VAST-supporting video player.

```
https://ads-[client].adhese.com/m/[adpod type]/sl[position
identification]/[targeting]/?max_ads=[max number of desired ads]&t=[timestamp]
```

- adpod type — **adpod** for VAST, **stack** for JSON.
- max_ads — the returned number of ads will depend on different parameters, like availability and targeting, but will never exceed max_ads.

Output

VAST

```
<VAST version="3.0>
  All returned VAST ads.
<VAST/>
```

JSON

```
{
  "ads": [
 #JSON objects, one per ad.
  ]
}
```

Sequencing

Sequence suggestions are optional, and the given sequence number will not end up in the final ad, but Adhese will do its best to place the ad in the desired position within the pod. If `sequence=0`, the ad is preferred to be placed first in a pod. If `sequence=-1`, the ad prefers to be placed last in a pod. If `sequence=1`, it prefers the second place, and so on. This ordering happens after the ads are selected, so there is no guarantee that an ad with `sequence=0` will always be first in the pod (unless it is the highest priority of all possible options).

You can enter any number as a sequence suggestion. For example `sequence=<ADHESE_LIB_ID>`(in the template) will sort the ads in a pod by creative ID.

Templates

VAST

```
<Ad sequence="0">
VAST ad implementation
</Ad>
```

JSON

```
{
  "sequence": 0,
  "example field": "some value"
}
```

The sequence field must be added to the **main structure** of the JSON template. The order logic will not work if the field is added to a nested object within the JSON template. The field will be **stripped from the response** when the Adpod or Stack response is rendered.

Revision #12
Created 6 December 2024 11:59:34 by Casper Steuperaert
Updated 17 April 2025 13:08:05 by Ron Van Maanen