radioplayer

Web Console v3.1 User Guide

Table of Contents

What's New

- Responsive layout optimised for mobile devices
- VAST and AdsWizz ad support
- Enhancements to HTML5 audio playback including AAC support
- Enhancements to Flash playback including HLS support
- Easily enabled debug mode
- Easily enabled test mode setup
- Improved ticker supports full Now and Next

Foreword

This revision of the Radioplayer console builds on the strengths which the 3.0 delivered to the industry and delivers a set of incremental improvements which reflect current UX and commercial requirements.

Key to these improvements is the move to a responsive layout. It is still the intention that in desktop browsers, the console will open in a popup. But when opened on a mobile device, the layout adapts to the screen. Usability tweaks have been introduced including a clearer search field, an onboarding/EU cookie compliance announcement and an enhanced favouriting process.

To meet the needs of commercial partners, out-of-the-box support has been added for VAST video adverts and AdsWizz pre-roll and in-stream advertising. It is possible for a developer to configure a stations' console to support these technologies and allow monetisation of listening sessions. Display of track-now-playing has been updated to include the next-track-playing too, with the optional ability to add a click-to-buy button. This offers stations a further route to monetise via clicks to sales at music download sites.

Finally a set of technology-focussed features have been added with improvements to the audio engine which delivers greater stability, HLS and AAC support, allowing more choice over streaming technologies. This, alongside a simplified method to activate developer features brings together the ability to more easily see how the console is behaving during times of improvement or troubleshooting.

First run and cookie announcement

The Radioplayer web console now features a first-run feature induction experience when the user experiences Radioplayer for the first time. It illustrates how favouriting and the console's main menu works.

Additionally, to comply with EU cookie compliance and the UK enactment which implies consent after an announcement is shown, this onboarding experience also advises users that cookies are used and provides a link to the Radioplayer cookie compliance page.

Stations that already have their own cookie compliance page, are encouraged to take the text from the Radioplayer page and integrate it into their own - then update the link accordingly. To do this, edit `/js/lang-en-gb.js` and edit the value of `cookie_link`

Responsive layout

The Radioplayer web console is now responsive. There are three main sizes - the existing 360px width, a narrower width for small mobile devices and a maximum 720px width for phones and tablets.

The console **will continue to open in a popup** at its native size but these additional breakpoints allow for a more app-like experience on non-desktop devices.

The plugin space has been adapted to allow for this. It is now possible to set the plugin space to be responsive too. However many stations have plugin spaces which cannot be changed so provision has been made to ensure that both options are available. The console defaults to a non-responsive plugin space for backwards compatibility (so that stations can lift their existing plugin space and paste it in) but if responsiveness is required in this area, configure your console's index.html file with the following

```
var isResponsive = true;
```

VAST and AdsWizz support

The console now supports ads from VAST and AdsWizz pre-roll and instream ads and a station can use one, either or both should they so wish.

Configuring VAST support

The console is built with libraries that support both VAST 2 and VAST 3. When a console is configured for VAST, it will open as normal, make a call and if there is a VAST response then the console will resize, play the video and revert to its native size when the video has finished. Of course, if the console has not opened in a popup, the resizing parts of this will not take place. The video will be displayed in a space which is up to 640px wide and 480px high. It should be noted that the VAST response must contain a reference to a playable video asset. Do not use SWF's as ads or ad containers. It's not possible to track or render them correctly.

Companion ads are supported and will be displayed below the video player when they are defined in the ad response. It should be noted that a limited amount of space exists below the video player and we recommend a maximum size of 700px x 100px to prevent layout breakages or unexpected scrolling.

To configure your console, you will require your vast ad tag. This looks something like

`http://my.adprovider.com/path/to/vasttag.xml`

Edit the console's index.html file and locate the code block below

```
var vastAds = {
      enabled: false,
      vjsSkin: 'vjs-default-skin',
      vastTag: 'VAST_TAG_URL'
};
```

Update this block with your VAST ad tag URL and ensure the `enabled` flag is set to true to give something like this:

```
var vastAds = {
      enabled: true,
      vjsSkin: 'vjs-default-skin',
      vastTag: 'http://my.adprovider.com/path/to/vasttag.xml'
};
```

It is possible to change the styles of the VST video player by changing the value of `vjsSkin` and specifying your own stylesheet. This is not recommended and you should leave as is. Ensure the value for `enabled` is not enclosed in quotation marks.

Once editing is complete, save the index.html file and replace your existing version with this one. A VAST ad will then be called whenever the console is loaded.

Configuring AdsWizz support

The Radioplayer console is also AdsWizz-enabled out of the box. AdsWizz, at the time of writing, offer two solutions - a client-side solution and the more popular server-side solution. In the case of a client-side solution, adverts are delivered via VAST and the framework explained above will allow a station to deploy that.

However for stations using the server-side solution (whereby ads are injected into a listeners' stream at the start of - and during the stream) the console can be configured to respond to this. Like VAST, a companion ad is available but only available when the console is using Flash as a playback engine since the HTML5 player cannot extract the metadata from the stream. In this solution the companion ad is permitted to occupy up to the full size of the plugin space. A station, during their setup with AdsWizz will determine what sizes of companion ads they wish to support. This allows everything from a simple MPU to a rich media experience in the plugin space.

To configure the console you will need:

a) Your **AdsWizz hostname**. This isn't the full Adswizz domain - just the first part. So, if AdsWizz give you a domain such as 'myadswizzdomain.adswizz.com', please use only 'myadswizzdomain'.
b) **AdsWizz Zone ID** for at least one of either the pre-roll or mid-roll ad streams. These are two numbers specific to the station, which AdsWizz will provide you. In this example, we will use 11111 for pre-roll and 22222 for mid-roll
c) Additionally you may have an **Adswizz Player ID**
d) An flv encapsulated stream, ideally AAC, from which the metadata can be extracted

Edit the console's index.html file and locate the following code block:

```
var adsWizz = {
      enabled: false,
      domain: 'ADSWIZZ_DOMAIN',
      prerollZoneId: PREROLL_ID,
      midrollZoneId: MIDROLL_ID,
      playerId: 'UKRP'
};
```

Replace the placeholders with your hostname, preroll ID and midroll ID so ending up with something like this:

```
var adsWizz = {
      enabled: true,
      domain: 'myadswizzdomain',
      prerollZoneId: 11111,
      midrollZoneId: 22222,
      playerId: 'UKRP'
};
```

Ensure the `enabled` flag is set to true and that only the `domain` and `playerId` value should be in quote marks. The rest are boolean and numeric, so should be stated as such.

The index file can then be saved and replaced on the server.

Enhancements to the audio engine

The audio engine now supports three new features

- Improved stream selection and failover recovery
- Extended support for AAC in the HTML5 audio engine
- New support for HLS in the Flash engine

Stream selection and failover recovery

The audio engine now inspects the browser to see whether or not it has Flash and/or HTML5 audio playback capabilities. Where a device supports Flash, it will always be selected as the preferred playback method. If Flash is not present, then HTML5 playback will be attempted. However this behaviour can be overridden for markets where Flash is less prevalent, by adjusting the following flag, located in the index.html file

```
var preferredPlaybackMethod = 'flash';
```

The acceptable values are `html` or `flash`

Because of these changes, the provision of a separate HTML5 stream value has been removed. A station can now set one array of streams, listed in the index.html file as `audioArray` and the console will recurse through them using whichever playback method works.

After all streams in the list have been exhausted, the console will attempt the list again using the alternative playback method (if available) ie. if it started with Flash, it'll try the streams in HTML5 and vice versa. If all options fail, the console will display the necessary warning.

Extended AAC support in HTML5

Following extended browser support for AAC, this is now supported in the HTML5 engine. Unlike AAC support in Flash, the stream must not be flv encapsulated. Please also note that few versions of Firefox support AAC playback and IE needs aac in an mp4 wrapper which is something that few streaming servers provide. Further, Chrome on Android has been known to take a significant time to buffer AAC streams.

We still recommend that if you wish to use AAC streams, you use the Icecast KH server which can flv encapsulate the stream through the simple use of `type=.flv` in the querystring. You would then create an audio stream array which had both the encapsulated version (using the `'httpmp4m4a'` audio type) and the non-encapsulated version (using the `'aac'` audio type), which will allow the engine to work in both Flash and non-Flash scenarios.

```
Example
var audioArray = [
      {
            audioType: 'httpmp4m4a',
            audioUrl:
'http://myradiostream.com/audio?type=.flv'
      },
      {
            audioType: 'aac',
            audioUrl: 'http://myradiostream/audio'
      }
]
```

HLS support in Flash

The Flash player now supports HLS (other upcoming formats such as HDS and MPEG-DASH are not formally supported at this time however the door has been left open to add them in the future). HLS servers should be configured so that the segment lengths conform with Apple's 10 second recommendation. Shorter segments can cause choppy playback.

To configure the console, set the audioType to hls and specify the URL of your M3U8 file.

```
Example
var audioArray = [
      {
            audioType: 'hls',
            audioUrl: 'http://myradiostream.com/hls.m3u8'
      }
]
```

Remember that Flash will require a crossdomain.xml file in most situations. Also in-stream metadata is not widely supported. Where it is available, it will be surfaced to the user.

Development and debug modes

Debug mode

At times it is useful to see what the console is doing, during moments when its behaviour is unexpected - or when using an unusual audio stream. The console has always had a diagnostic mode which would output information to the javascript console however enabling it has always been difficult.

Now it is possible to enable it through the use of a simple querystring added to your console URL *(not stream URL)*. Just use the flag `debug=true`.

So in the case of a console playing a live audio stream, your debug enabled URL might look like:

```
http://www.mystation.com/radioplayer/?debug=true
```

For an on-demand clip, where the clip URL is already passed in through the querystring, use an ampersand delimiter as usual to pass in this extra flag.

```
http://www.mystation.com/radioplayer/?rpAodUrl=http://downloa
d.shareradio.sharp-stream.com/shareradio_2015-10-20_1700-
1800_MarketWrap_50f0d899b73749fe916d1ea818394c34.mp3&rpSt=sha
re+radio&rpSrp=1.0/&debug=true
```

Development mode

Radioplayer has available a development server which can be set up to work like the live environment but allow a station to test metadata and ingest flow without needing to put the console into production.

The console fully supports interfacing with this environment. To switch the console from addressing the production APIs to using a development API, use the following querystring parameter in your URL

```
?devapi=RPDEV_DOMAIN_NAME
```

The development server is not an always-available environment. It is necessary to book a window of testing on it, via a form on the Station Support site. Once a testing window has been allocated, you will be assigned with a domain name to use for this value.

Improvements to Now and Next

The ticker area has been improved. Not only does the scrubbing control work better on touch devices as part of the responsiveness functionality, the console supports the ability to show not just the track now playing but the next upcoming track.

To enable this, a station needs to be using the full ingest process and should submit PI/PE events with the correct expected TX timestamp. So long as upcoming events are ingested, they will be communicated to the console for display at the correct time.

Did you know that the console also supports Click To Buy, whereby a track title shown in the ticker can have a buy button available which links to a third-party destination of your choice. There is more information about this here

Other changes

- In order to evolve the Radioplayer web console product, some changes to existing functionality have been made
- Internet Explorer 7 is no longer officially supported although it's likely that users with Flash will still be able to use the console without too many problems
- Internet Explorer 8 is offered with limited support - Flash playback only and video ads are not available
- Windows XP is not supported however the console has been tested on an IE8/WinXP configuration
- Interstitials are now deprecated. Stations needing an interstitial-like experience should consider building their own solution.
- The HTML5-only stream is now deprecated - stations should provide an array of playable streams.
- Overlays, managed from the Station Control Panel, are supported but are planned to be retired in due course. It will still be possible for developers to enable overlays manually.
- Live and on-demand streaming over RTMP works only when Flash is the playback method.