# AudioVisual Video Player



## Introduction

Sportradar Video Player is the only video player designed for bookmakers. It's implemented by several online bookmakers and physical bet shops all over the world.

We develop our product in cooperation with bookmakers and thus know their needs. The Sportradar Video Player is a lightweight, feature rich and in-house developed, HTML5-first solution that supports all standard streaming and video formats, ready to cover all key platforms. With a smooth integration, 3rd parties can enhance their products by offering a complete Live and VOD player experience. The Product is deeply integrated into Sportradar production workflows, data feeds and optimized for the main CDN vendors.

## Feature List

### Supported Formats

- HLS (Flash/HLS.js)
- HDS
- MPEG-DASH
- RTMP
- MP4 progressive
- WebRTC
- CMAF using dash.js

### Playback features LCO VOD / LIVE

- Seeking (VOD / LIVE: DVR)
- Autoplay on/off
- Multi-bitrate streaming
- Manual and automatic quality index switching
- GEO-Blocking
- Adjustable buffer length for all stream types
- Supports different video scaling options
- JS API allows access to low-level information for each supported stream type (fps, bufferLength, etc.)
- Comprehensive API for playback control and quality control
- Detachable player
- Cue points support for triggered actions or JS callbacks

### User Interface

- Can be run as chromeless version fully controlled via API
- Skinnable (CSS and configuration themes)
- Responsive
- Custom Controls (rewind, forward, play/pause, info, share, backup-stream, HD, settings, volume, fullscreen)
- Title
- Multiple languages
- Fullscreen
- Audio volume controls
- Visible Watermark
- Poster Image
- Countdown to streamstart
- Various templates if stream is ended (e.g. Related videos (+ countdown to play the next video)
- Social Sharing
- Custom error and info messages

# Player Integration

The Player is easy to embed and can be controlled both via internal and external configuration files.

Below you can find a sample HTML snippet of the minimum version of the player:

**index.html**

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">   <meta http-equiv="x-ua-compatible" content="
ie=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="https://avplayer-cdn.sportradar.com/dist/latest/avvpl-
player.js"></script>
  <link rel="stylesheet" type="text/css" href="https://avplayer-cdn.
sportradar.com/dist/latest/styles.css">
  <title>Sample Sportradar Player</title>
  <style>
  body {
    margin: 0;
    padding: 0;
  }
  #playercontainer {
    position: relative;
    width: 100%;
    height: 100%;
    border: 1px solid black;
  }
  </style>
</head>

<body>
  <div id="playercontainer">
    <video>
        <source
          src="http://dl5.webmfiles.org/big-buck-bunny_trailer.webm"
          type="video/webm">
        </source>
        <source
          src="http://ia800501.us.archive.org/10/items/BigBuckBunny_310
/big_buck_bunny_640_512kb.mp4"
          type="video/mp4">
        </source>
    </video>
  </div>

  <script>
    var config = {
      id: "playercontainer"
    };

    var avvplInstance = new avvpl.setupPlayer(config);

  </script>
</body>
</html>
```

- Load the Player script from our Cloud Service

```
<script src="https://avplayer-cdn.sportradar.com/dist/latest
/avvpl-player.js"></script>
```

- Load the Player Stylesheet

```
<link rel="stylesheet" type="text/css" href="https://avplayer-cdn.
sportradar.com/dist/latest/styles.css">
```

- Create the player object
- Set the Player configuration variables

This is the most basic implementation of Sportradar player.

If you decide to load the files from a different location, please make sure it gets loaded from the html file location.

The html snippet contains a div element with the id of "playercontainer" (line 24).

There is a "video" tag with two different video sources within this div element.

Sportradar player will playback the first source it can manage to.

To instantiate the player we call below on line 42:

```
var avvplInstance = new avvpl.setupPlayer(config);
```

The config is the only "mandatory" object that you need to pass to the player.

We have a wide range of configuration possibilities that can be provided to the player for playback options and UI look and feel.

The must parameter within config is the id of the div element specified on line 24.

Sample config on line 38:

```
var config = { id: "playercontainer" };
```

Other config parameters will use default values as specified within the player code.

Some of the important parameters include the following:

- **handlers**
  The order specified in this array is important as the player decides the handler to be used based on this order in case a stream can be played back using more than one handler. For example, in an android device, a hls stream (.m3u8 url) can be played back by both hls handler(using hls.js) and html5 handler(native html5 video).
- **bufferTime**
  The player by default uses buffer value as below:

    - For DASH:
      "useSuggestedPresentationDelay":
      Default value is true. If manifest contains suggestedPresentationDelay - This value is used. If set to false in config: Player uses the "bufferTime" set in config: Default used is 2, If set in config, this value is used.

    - For HLS:
      If bufferTime is set, this value is used as liveSyncDuration(edge of live delay), and maximum delay allowed from edge of live will be bufferTime * 2.5

      If not set in config, default value used for edge of live delay - 2 * segment length, and maximum delay allowed from edge of live will be 5 * segment length

      Note: "useSuggestedPresentationDelay" has no effect on hls handler

    - For FLASH/RTMP:
      If bufferTime is set, this value is used. Else, by default a value of 4.5 seconds is used.

      Note: "useSuggestedPresentationDelay" has no effect on flash handler

- **allowFullScreen**
  If fullscreen icon should not be shown on UI and not be allowed via API, then set this value to false. Default is true

- **autoplay**
  By default, player tries to autoplay the stream. If this is not desired, set this to false. Note that some browsers/devices do not allow autoplay. In these cases, player shows an autoplay overlay even though this value is set to true.

- **enableUI**
  By default, UI is enabled. If this is not desired, then set the parameter to false

- **loglevel**
  By default, this is set to 0 (Off) To show:

  Just Errors on console : 1,
  Errors and Warnings on console : 2,
  Errors, Warnings and Info : 3

- **mute & volume**
  By default, the player tries to start unmuted with volume 50 (Except in case of mobile devices with autoplay)

  mute: true, volume: 45 -> Player starts muted (mute takes precedence)
  mute: false, volume: 45 -> Player starts with volume 45

- **enableCmaf**
  Default: false

  Enable (set to true) or disable low latency streaming supported by dash.js using CMAF (Common Media Application Format) chunks. Works only with MPEG-DASH streams.

# Notes

- All Endpoints of the JS API can be found here: https://avplayer-cdn.sportradar.com/docs/classes/videoplayer.html

- Recommendation when using dash.js directly for playback MPEG-DASH streams:

**dashjs settings**

```
1- dashjs.updateSettings({
  streaming: {
    fastSwitchEnabled: true,
    bufferTimeAtTopQuality: 2,
    bufferTimeAtTopQualityLongForm: 2,
    stableBufferTime: 2,
    lowLatencyEnabled: true, // For CMAF-DASH only, otherwise false
    jumpGaps: true
  }
});

// If manifest contains suggestedPresentationDelay, then this can also
be used if desired. Once manifest is loaded, do the following:
2- dashjs.updateSettings({
     streaming: {
        bufferTimeAtTopQuality: suggestedPresentationDelay,
        bufferTimeAtTopQualityLongForm: suggestedPresentationDelay,
        stableBufferTime: suggestedPresentationDelay
    }
});
Only values changed with (2) is overwritten with the second update.
```

- If it is certain that only dash streams would be used, you can also use : "avvpl-dash-player.js" in place of "avvpl-player.js". The same goes for hls, flash, html5.
- All these files contain ui related code. If this is not needed, you can also use :
  "avvpl-player-without-ui.js" in place of "avvpl-player.js"
  "avvpl-dash-player-without-ui.js" in place of "avvpl-dash-player.js"
  The same goes for hls, flash, html5.

# Download

Index.html document (Please download)

Back to top