SDK Player API Guide

September 22, 2014

Revision 4.2

Barco, Inc.: 1287 Anvilwood Avenue, Sunnyvale, CA 94089 www.barco.com
Office: +1.408.400.4100 Fax: +1.408.400.4101

Table of Contents

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

Introduction

This guide explains how to integrate the Barco Media Player into an HTML webpage. Player operation details are explained in the *Collaboration Client User Guide*.

Assumptions

This document assumes you know how to obtained *mediaURL*s and supply login credentials via the REST API. See the *REST JSON Format API* (API-BARCO-Rest-JSON.pdf) for details. Knowledge of jQuery is also beneficial but not required.

System Requirements

This SDK player has been verified to be compatible with these platforms.

- Windows 7 with these web browsers
  - Microsoft Internet Explorer editions 9–11
    - with Flash Player Add-on 11 or later
  - Google Chrome version 24 or later
    - with Flash Player plug-in 11 or later
  - Mozilla Firefox version 30 or later
    - with Flash Player plug-in 11 or later
- iOS 7 with this web browser
  - Mobile Safari 7

Conventions Used in This Document

Source code is shown in `this fixed-width font`. Parameters and metadata within code snippets are shown in *this font*. Optional parameters are shown enclosed by [*square brackets*].

Terminology

Cut list: A feature that allows the user to cut and trim recorded videos down to a more concise video presentation.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

Flash Player: A lower-level, widely popular, add-on or plug-in for web browsers provided from Adobe Systems which can play videos from dynamically loaded custom Flash modules. Barco supplies such a custom module with this SDK Player which plays HLS videos.

HLS: Apple's *HTTP Live Streaming* video format. There are both Flash and HTML5 players capable of playing this format on most desktop browsers and Apple iOS devices such as the iPad. The SDK Player automatically chooses the best player for the task. Because HLS is delivered via standard HTTP, it can be delivered across the greater Internet. However, HLS pays the penalty of longer latency and lack of synchronization compared to RTP and V2D.

HTML5 video player: A lower-level video player built into most HTML5 web browsers. The SDK Player presently uses this built-in web browser player to play HLS videos on some mobile devices.

mediaURL: A secure mechanism for playing, recording and sharing video streams in Barco's collaborative, enterprise environment.

RTP Player: A lower-level video player as an ActiveX add-on from Barco that can play RTP and V2D video streams which only runs in Internet Explorer. It must be installed by an administrator. When installed, the SDK Player will select this add-on to play the RTP and V2D streams (unless otherwise instructed not to do so). This player provides a better video experience with lower latency then other streaming players.

SDK Player: A high-level, full session-based video player that can select and play multiple video formats with a uniform, controller-bar, user interface and feature set. The SDK Player adds features like bookmarks and recording which are not available in the lower-level players. The SDK Player does not play videos directly. Instead, it loads one of the lower-level players to play the appropriate video stream based on the client's platform, type of media and other factors. The API reference within this document refers to this higher-level SDK Player.

PVR: Abbreviation for *Personal Video Recorder*, also known as **DVR** (*Digital Video Recorder)*, which is a live video stream within a time-shift window that allows the user to play back other parts of the live stream within this time-shift window or to catch up to the current moment again.

RTP: Abbreviation for standard *Real-time Transport Protocol* video streams which are playable by the SDK player on some browsers with the RTP Player add-on. Because RTP does not use HTTP, it is generally usable only within an organization's intranet.

V2D: Barco's high-definition video format (Video-To-Data) with better fidelity and lower latency than either HLS or RTP. Because V2D does not use HTTP, it is

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

generally usable only within an organization's intranet. Multiple V2D streams can be fully synchronized which is useful for 3-D and multi-media.

SDK Player Controls

The SDK Player adds several features to the video player user interface. These features are summarized below.

The SDK player provides a session control bar with a timeline and buttons. The individual controls are numbered above and described below:

⓪ Session error alerts. See the **Troubleshooting** section for more detailed descriptions of possible errors and ways to rectify.

① iOS play icon. This icon only appears on the iPad. See **Technical Limitations on iPad Media Playback** section for more details on this.

② This tooltip indicates elapsed time on the scrubber bar when mouse cursor hovers over that location in the scrub bar. The tooltip does not appear on touch devices because there is no concept of hovering in the touch UI.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

③ Current time since start where video is playing and the current time location of the scrubber control.

④ Time duration of PVR timeline window (for live streams) or the time duration of recorded video streams.

⑤ Session controls. From left to right the controls are: Play/Stop, Pause/Resume, Skip Back, Skip Forward, and Catchup (if live). The last three are not available on the iPad for live video streams.

⑥ The scrubber control where the video is presently playing. This is draggable on desktops but not on the iPad. To move the scrubber on the iPad, touch anywhere on the scrubber bar to move the scrubber control where you want it. See **Technical Limitations on iPad Media Playback** for more details on this.

⑦ Scrubber/progress bar. Green fill shows the portion of video presently accessible. For recorded videos, the entire video is accessible which is indicated by a bar which is entirely green. For PVR (live) videos, the green fill shows the timeline window presently accessible.

⑧ Bookmark controls. From left to right the controls are: Add, Edit, Delete and near the scrubber bar, the position of a previously added bookmark. Clicking on the bookmark resets the video playback to that stored timeline position.

⑨ Window controls. From left to right the controls are: Full-screen/Normal-screen and Mute/Unmute. The Mute/Unmute control is not shown on the iPad. See **Technical Limitations on iPad Media Playback** for more details on this.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

Technical Limitations on iPad Media Playback

Apple enforces a number of technical limitations on iOS Mobile Safari. These include:

1. Play must be started explicitly by a direct touch action from the user. Therefore, autoplay is not allowed and calling the 'play' API will only cause the play icon to appear on the video viewport awaiting the user's direction action to start playing.
2. Only one media source (whether audio or video) can be played at a time. Therefore, simultaneous media playback is not supported.
3. Volume and mute are controlled solely by physical manipulation of the device's volume controls (or via the volume control UI built into the Apple's own QuickTime Player launched from Safari). Therefore, JavaScript calls on the mute/unmute API have no effect.
4. True full-screen playback is not allowed unless the Apple's own QuickTime Player is launched from Safari with its own control bar thus removing the advanced features of the Barco Session Control Bar. Therefore, the Barco SDK player does not use true full-screen on iOS devices.
5. Support for live playback is limited in that it is not possible to seek to different parts of a live video. Therefore, the skip back, skip forward or catchup functions are not supported. Repositioning the scrubber control is also not supported either for live.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

Installing and Using the Barco Player SDK

Before the SDK player can be embedded within your web application you need to locate the `ipvssdk.4.1.rXX.bXX.tar` distribution and either untar it in a directory on your localhost or untar somewhere into your web server document directory. The base directory when unpacked will be `BarcoSDK`. The directory layout looks like this:

`BarcoSDK`

- `player`
  - `doc` *{where this document resides}*
  - `src`
    - `guiAPIAgent` *{contains some utility code}*
    - `jquery` *{contains a copy of the open-source jQuery library}*
    - `js` *{contains the SDK Player code}*
    - `resources` *{contains CSS/image files used by the SDK Player}*
    - `Player.css` *{CSS for Player.html only}*
    - `Player.html` *{Sample player code as a starting point}*
    - `Player.js` *{JS code for Player.html with example API calls}*
    - `PlayerEmbedURL.html` *{Simpler way to embed the SDK Player}*
  - `rest` *{outside the scope of this document}*
  - `xmpp` *{outside the scope of this document}*

You can either include your code for embedding the SDK player within the `…/BarcoSDK/player/src` directory (where `Player.html` is) as we do for our examples. Or, you can embed from outside being mindful that all scripting and CSS links referencing the SDK player will have to be modified to include `…/BarcoSDK/player/src/…` in the paths.

Best Practices for using the SDK Player in custom web apps

The `BarcoSDK` directory should be left intact and your code should reference the script and resources files with the appropriate path to access those scripts and resources. This separation will allow you to receive updates and bug fixes from Barco with minimal interruptions on your code. For example, if you place your web

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

page/app parallel to the `BarcoSDK` directory your script and resource links should contain paths that look like this:

```
<script src="./BarcoSDK/player/src/js/BarcoPlayerInterface.js"></script>
```

The difference between the two sample HTML files: `Player.html` and `PlayerEmbedURL.html` are explained below.

Simple Player without using the SDK API

The simplest way to embed the player is do it in the same way that the `PlayerEmbedURL.html` does it. This very much follows the same approach that most 3rd parties embed YouTube videos. It's quick and simple but you cannot access the SDK Player API or customize its behavior very much. Similar to YouTube, you include the *mediaURL* and any additional player configuration parameters within the same URL string. There is no need to reference any addition script or resource links in your web page/app. You will have to substitute your own *mediaURL* where the placeholder is shown below. Obtaining a *mediaURL* is outside the scope of this document. Here's an example:

```
<div class="mainContent"  style="margin:auto; width:640px; height:360px;">
  <iframe id="myFrame" style="width:100%;height:100%" frameBorder="0"
     scrolling="no" allowfullscreen>
  </iframe>
</div>
```

This embeds the player within an <iframe> then we pass the *mediaURL* and any player configuration parameters to it like this, all as one string:

`document.getElementById('myFrame').setAttribute('src',` *mediaURL* `+ "&confOpt_playerPref=HLS");`

This bit of JavaScript replaces the empty `src` attribute within the `<iframe>` tag with the URL to play. The string in green is appended to the *mediaURL* as an inline player configuration option and interpreted the same way as these equivalent JavaScript SDK Player API calls:

videoPlayerRef`.player('conf',{'playerPref':'HLS'});`

videoPlayerRef`.player('play',` *medialURL*`);`

See the **Player Configuration** section for more options. Additional player options can be appended to the same URL using the same pattern as above.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

Simple Player using the SDK API

Another way to embed the SDK Player is the same way as is done in `Player.{html,js}`. Before we go there, here is a very simple example of a player page that uses the SDK Player API to play a video. (You will need to place this complete example in the same directory as `Player.html` above if used without modification. The means for obtaining the *mediaURL* is covered in the REST API documentation.)

```
<!DOCTYPE html>
<html>
<head>
      <meta http-equiv="X-UA-Compatible" content="IE=EDGE" />
      <title>Simple Media Player</title>
      <script src="./jquery/jquery-1.11.0.js" type="text/javascript"></script>
      <script src="./jquery/jquery.fullscreen-0.4.1.js"
type="text/javascript"></script>
      <script src="./guiAPIAgent/scripts/BarcoUtil.js"
type="text/javascript"></script>
      <script src="./js/plugin-inheritance.js" type="text/javascript"></script>
      <script src="./js/BarcoPlayerInterface.js" type="text/javascript"></script>
</head>
<body>
  <div id="p1" class="barcoplayer" style="width:640px; height:360px;"></div>
  <script>
    $(document).ready(function() {
            var parentElemRef = $('div.barcoplayer');
            var videoObjRef = parentElemRef.player({playerPref: 'HLS'});
            videoObjRef.player('play', mediaURL);
    });
  </script>
</body>
</html>
```

Notes:

1. The `IE=EDGE` attribute is required for Internet Explorer when running the SDK player on a PC connected to the same intranet as the server. This keeps IE from defaulting into IE 7 compatibility mode which breaks the SDK player code.
2. `BarcoPlayerInterface.js` is the first part of the SDK Player.

   Please do not modify this code per license terms. Barco may provide periodic updates and bug fixes that you may receive in upcoming releases.
3. Player is implemented as a jQuery plugin. Commands to instantiate and make API calls follow jQuery plugin syntax.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

A more complete example of this approach is included in `Player.{html,js}` where you can see how additional API calls are hooked up to button events as-well-as how SDK Player events can be subscribed to.

SDK Player API

The player API consists of a set of commands which control video playback or obtains state information from the player. You can use the Player API to construct your own player controls and timeline output display. When the SDK player object is created, your code needs to specify where in the DOM the player object will appear. In the example below this is the *parentElemRef* which is usually an HTML tag. (You can get a reference to the HTML element using JavaScript calls like getElementById() or the jQuery equivalent.) The player will create an instance of itself as an inner element and return it. Your code can then reference the player instance via a variable reference (e.g. *videoObjRef*) as shown in the example below. When the player object is first created via

`var` *videoObjRef* `=` *parentElemRef*`.player(`*confObj*`);`

the SDK Player API can then be called thus:

videoObjRef`.player(`*command, arguments…*`);`

The *command* is a string containing a word like `'conf'`, `'play'`, etc. which is shown in the 1st column of the table below followed by zero or more *arguments* depending on the *command*.

Examples

Call player init with an in-line *confObj*:

```
<script>
parentElemRef.player({
   playerPref: 'HLS',
   skipBackwardOffset: 30000
});
</script>
```

You can also play using cascaded calls (like in jQuery):

parentElemRef`.player({playerURL:` *mediaURL*`}).player('play');`

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

API Reference

API	Description
`'conf', confObj`	Set or modify the SDK player configuration. Can be called any time after player is created. Multiple calls to `'conf'` with different confObj's will be merged into the overall configuration except where the same property name is used more than once. In that case, the new value for that property replaces the old one.
`'authenticate', { username:`*username*`, password:`*password*`, resource:`*resource*` }`	This method must be called when a user will be using the cut-list feature. These credentials must match the same credentials used during initial client login. Ideally, the same info used for login should be saved and passed into this method. The single argument must be a JavaScript object with the property names given. This method must be called before the `'play'` method is called in order to take effect.
`'play'[,` *mediaURL*`]`	Sends a request to start streaming a video from the *mediaURL*. If the `playerURL` property is specified in the *confObj* passed to `'conf'`, then the *mediaURL* does not have to be specified here.
`'pause'`	Pauses the currently playing clip or stream. `'play'` must be called before this does anything.
`'resume'`	Resumes the currently paused clip or stream
`'skipforward'[,` *offset*`]`	Timeshifts the video forward with the specified optional `offset` (in milliseconds) relative to current position. If `offset` is missing, the configured default is used. See the **Player Configuration** section for details on setting defaults.
`'skipbackward'[,`*offset*`]`	Timeshifts the video backward with the specified optional *offset* (in milliseconds) relative to current position. If *offset* is missing, the configured default is used. See the **Player Configuration** section for details on setting defaults.
`'catchup'`	Timeshifts the video to the current (live) time.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

`'stop'`	Stops the current stream
`'seekTo'`, *startOffset*	Reposition playback within the video/audio stream to a *startOffset* (in milliseconds) relative to the beginning of the recorded stream. Does not work for live. If the time-offset exceeds the stream length, the *startOffset* is set to the current stream length. A negative *startOffset* is treated like zero.
`'mute'`	Mutes the sound.
`'unmute'`	Unmutes the sound.
`'getVersion'`	Returns the current version of the SDK player release as a string, like `'4.1.r15.b5'`
`'getMediaURL'`	Returns the current mediaURL.
`'isPaused'`	Returns `true` if the SDK player is paused.
`'isPlaying'`	Returns `true` if the SDK player is playing.

SDK Player Configuration

When the SDK player object is created, you can pass various configuration options to the player at this time via:

```
var videoObjRef = parentElemRef.player('conf',confObj);
```

you have the option to configure several player properties via the *confObj* argument. The *confObj* is a JavaScript structure containing various properties as keyword-value pairs.

confObj's can be passed to the `'conf'` commands after the player is created and initialized with different sets of configuration options. All of these calls using *confObj*'s will be merged together. If more than one *confObj* specifies the same property name, the value of the last setting takes effect. Properties can be added or modified but not removed.

Property	Default Value	Valid Entries	Description
`playerPref`	`'AUTO'`	`'AUTO'`,	Tells the SDK player which player technologies are

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

		'HLS'	preferred. 'AUTO' uses whatever player technology is best as determined by Barco. 'HLS' uses any HLS compatible player. This might be Flash or HTML5 depending on the platform.
customSkin			Define an optional custom *skinObj* which overrides various presentation aspects of the default control bar UI. See the **Control Bar Skinning API** section for details.
playerURL			Set the *mediaURL* during configuration. Preferred approach is to provide *mediaURL* with play command.
showControlBar	true	true, false	Show the SDK player control bar (all controls). If true, the visibility of the three control groups, just below, can be customized within the control bar. If false, the entire control bar is hidden.
showBookmarkCreation	true	true, false	Show the SDK player bookmark creation controls
showBookmarkPlayback	true	true, false	Show the SDK player bookmark playback controls
controllerBarFixed	false	true, false	When true, the control bar is always fixed (pinned) to the bottom of the viewport with the video stacked above it. When false, the user can decide

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

			whether the control bar is fixed or auto-hiding via the pin/unpin control.
`skipforwardOffset`	`10000`		When the skipforward button is pressed, this time offset (in milliseconds) is used when repositioning. Also applied when the `'skipforward'` API command is invoked without the optional *offset* argument.
`skipbackwardOffset`	`10000`		When the skipbackward button is pressed, this time offset (in milliseconds) is used when repositioning. Also applied when the `'skipbackward'` API command is invoked without the optional *offset* argument.

The following configuration parameters are also available when using the ActiveX RTP Player plug-in within Internet Explorer for V2D encoded streams.

Property	**Default Value**	**Valid Entries**	**Description**
`fps`	`15`	`1-60`	The frame rate the Player will decode at.
`http`	`true`	`true, false`	Specify if the stream transport protocol is HTTP (true) or UDP (false).

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

SDK Player Events

You can use the `subscribe` method to execute your own JavaScript when something happens in the player. Multiple method can be associated with the same event. `unsubscribe` (with the same arguments) can be called to unregister methods. This event mechanism assumes the use of jQuery which is required by the SDK Player for other reasons, hence the '$' object. For example, you can subscribe to 'onStart' events with:

```
$.subscribe("onStart", function(currentTime, duration){
     //do your start processing
});
```

These events are supported by the SDK Player:

Event	Description
`onStart`	The Player has starting playing a video or stream. The arguments passed to this callback are (*currentTime*, *duration*).
`onPause`	The Player has been paused. The arguments passed to this callback are (*currentTime*, *duration*).
`onResume`	A paused video has been resumed. The arguments passed to this callback are (*currentTime*, *duration*).
`onCatchup`	A PVR'ed video has caught up to real-time. The arguments passed to this callback are (*currentTime*, *duration*).
`onSeek`	A video has been time shifted (skip forward or skip backward) via PVR. The arguments passed to this callback are (*currentTime*, *duration*).
`onFullscreen`	When the player enters or leaves full screen. Argument will be `true` or `false` to indicate when full screen is active.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

`onMute`	When the Player has been muted
`onUnmute`	When the Player has been unmuted

`onError`	The Player is in an error state. The argument passed to the callback is an *errorObj* with the following properties:
	`{`
	`  type:` *string*`,`
	`  id:` *string*`,`
	`  message:` *string*`,`
	`  playerId:` *string*
	`}`

`onStop`	The Player has stopped playing a video or stream

Control Bar Skinning API

The default control bar appears the way it is shown on page 5. The default control bar presentation can be modified via a *skinObj* structure which you create. The *skinObj* is then passed to the SDK Player within a *confObj* as a `customSkin` property. You have the option of overriding just those presentation properties that you want to override while staying with the default presentation for everything else.

There three types of values which can be passed as property values. In general, the syntax follows a subset of the equivalent CSS 3 types. The type of value accepted is indicated within a *skinObj* property name itself such as `borderWidth` or `fillColor`.

Type	Syntax	Description
line width	*integer* | "*integer*`px`" | `"medium"` | `"thin"` | `"thick"`	A positive decimal pixel width for a border or a named value with identical meanings as defined in the CSS 3 specification.
color	`"`*colorname*`"` |	A standard CSS 3 *colorname* or a red-green-blue color value where

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

	"#*RGB*" | "#*RRGGBB*" | "rgb(*red*, *green*, *blue*)" | "rgba(*red*, *green*, *blue*, *alpha*)"	*R*, *G*, and *B* in the syntax represents a case-insensitive hexadecimal digit. Or, when enclosed by `rgba(…)` the *red*, *green*, *blue* and *alpha* values are in decimal. Where *red*, *green* and *blue* are integers between 0-255 and *alpha* is a float between 0.0-1.0. 1.0 is fully opaque and the default when not provided. Note: *alpha* may not be honored by some web browsers.
image	{width:*integer*, height:*integer*, image:"url(*base64-encoded image*)"} Be careful *not* to include the ';' after the trailing ')' which is often included by base64 image encoders. External image URLs are not allowed at this time due to the possibility that external URLs can become stale after a DNS or IP address changes or when a server becomes inaccessible. This may be the result of an IT person reconfiguring a network unaware of such URL dependencies. The end result is that images with stale ULRs will disappear.	A JavaScript object containing 3 properties: `width`, `height`, and `image`. Normally the `width` and `height` properties are the same as the native image pixel size but, if different, icon images will be rendered with the specified `height` and `width` properties rather than the native image pixel size. The `image` property follows the CSS 3 syntax for inline images. In the case of PNG images, the *base64-encoded image*, begins with "…". The MIME types `image/jpeg` and `image/gif` are also accepted.

The full *skinObj* data structure is fairly large but only those portions of the data structure where you need to provide overrides are actually required. The structure must begin with a `skinObj` property which is the only property that is always required. The `global` section provides some basic properties that affect the presentation globally such as changing the font colors across all sections of the control bar. The `controlBar` section and subsections provide finer control over the same rendering properties. Here is the complete *skinObj* data structure:

```
{skinObj: {
  global: {
      fontColor: color; affects almost all font colors ,
```


Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

```
    lineColor: color; affects almost all border colors except the tooltip ,
    lineWidth: line width; affects almost border widths except the tooltip
  },
  controlBar: {
    backgroundColor: color; the next property overrides this one ,
    backgroundImage: image; width limit 1-2880; height limit 1-128, repeating ,
    toolTip: {
      fontColor: color (of time value above scrubber when hovering over bar) ,
      backgroundImage: image; width limit 55-60; height limit 20-32
      },
    currentPos: {
      fontColor: color (of time value to the left of the scrubber bar)
    },
    scubberBar: {
      borderColor: color ,
      borderWidth: line width ,
      progressBarFillColor: color (portion of video actually available) ,
      backgroundColor: color (portion of video not available)
    },
    scubberHandle: {
      borderColor: color ,
      borderWidth: line width ,
      fillColor: color
    },
    duration: {
      fontColor: color (of time value to the right of the scrubber bar)
    },
    bookmarkBar: { (appears for HLS media only)
      bookmarkIconImageOn: image; width limit 8-16; height limit 16-20 ,
      bookmarkIconImageOff: image; width limit 8-16; height limit 16-20
    },
    bookmarkPanel: { (appears for HLS media only)
      backgroundColor: color ,
      inputBackgroundColor: color (of text input field backgrounds) ,
      inputTextColor: color (of input text)
    },
    trimEditBar: {
      trimBarInProgressColor: color ,
      trimBarAddColor: color ,
      trimBarDeleteColor: color ,
      fontColor: color ,
      backgroundColor: color ,
      trimUndoIconImage: image; width limit 20-120; height limit 16-32 ,
      trimCancelImage: image; width limit 36-120; height limit 16-32 ,
      trimCancelImageHover: image; width limit 36-120; height limit 16-32 ,
      trimStartImage: image; width limit 36-120; height limit 16-32 ,
      trimStartImageHover: image; width limit 36-120; height limit 16-32 ,
      trimEndImage: image; width limit 36-120; height limit 16-32 ,
      trimEndImageHover: image; width limit 36-120; height limit 16-32 ,
```


Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

```
    trimDoneImage: image; width limit 36-120; height limit 16-32 ,
    trimDoneImageHover: image; width limit 36-120; height limit 16-32 ,
    trimDeleteImage: image; width limit 36-120; height limit 16-32 ,
    trimDeleteImageHover: image; width limit 36-120; height limit 16-32 ,
    trimSaveImage: image; width limit 36-120; height limit 16-32 ,
    trimSaveImageHover: image; width limit 36-120; height limit 16-32 ,
    trimSaveImageDisabled: image; width limit 36-120; height limit 16-32 ,
    trimSaveNewImage: image; width limit 36-120; height limit 16-32 ,
    trimSaveNewImageHover: image; width limit 36-120; height limit 16-32 ,
    trimSaveNewImageDisabled: image; width limit 36-120; height limit 16-32 ,
    trimSaveContImage: image; width limit 36-120; height limit 16-32 ,
    trimSaveContImageHover: image; width limit 36-120; height limit 16-32
  },
  zoomPanel: { (appears on IE only for non-HLS media)
    backgroundColor: color ,
    borderColor: color ,
    textColor: color ,
    textColorHover: color (as cursor hovers over item) ,
    selectedColor: color (of text background after selection) ,
    notchOffset: line width (actually the horizontal pixel offset; default 42px)
  },
  buttonBar: {
    playStartIconImage: image; width limit 36-60; height limit 16-32 ,
    stopIconImage: image; width limit 36-60; height limit 16-32 ,
    recordIconImageOn: image; width limit 36-60; height limit 16-32 ,
    recordIconImageOff: image; width limit 36-60; height limit 16-32 ,
    resumeIconImage: image; width limit 36-60; height limit 16-32 ,
    pauseIconImage: image; width limit 36-60; height limit 16-32 ,
    skipForwardIconImage: image; width limit 36-60; height limit 16-32 ,
    skipBackwardIconImage: image; width limit 36-60; height limit 16-32 ,
    catchupIconImage: image; width limit 36-60; height limit 16-32 ,
    bookmarkAddIconImage: image; width limit 36-60; height limit 16-32 ,
    bookmarkEditIconImage: image; width limit 36-60; height limit 16-32 ,
    bookmarkDeleteIconImage: image; width limit 36-60; height limit 16-32 ,
    trimIconImage: image; width limit 36-60; height limit 16-32 ,
    shareIconImage: image; width limit 36-60; height limit 16-32 ,
    snapShotIconImage: image; width limit 36-60; height limit 16-32 ,
    kbmOnIconImage: image; width limit 36-60; height limit 16-32 ,
    kbmOffIconImage: image; width limit 36-60; height limit 16-32 ,
    zoomIconImage: image; width limit 36-60; height limit 16-32 ,
    nativeResIconImage: image; width limit 36-60; height limit 16-32 ,
    fitWindowIconImage: image; width limit 36-60; height limit 16-32 ,
    fullscreenIconImage: image; width limit 36-60; height limit 16-32 ,
    normalscreenIconImage: image; width limit 36-60; height limit 16-32 ,
    muteIconImage: image; width limit 36-60; height limit 16-32 ,
    unmuteIconImage: image; width limit 36-60; height limit 16-32 ,
    pinIconImage: image; width limit 18-24; height limit 16-32 ,
    unpinIconImage: image; width limit 18-24; height limit 16-32
```


Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

```
          }
        }
      }
    }
```

An example,

```
var greenSkin = {skinObj: {
        global: {
          fontColor: "rgb(128,255,128)"
        },
        controlBar: {
          backgroundColor: "#040",
          scubberBar: {
            progressBarFillColor: "green"
          }
        }
      }};
var videoObjRef = parentElemRef.player('conf',{customSkin: greenSkin});
```

when called before `'play'` is called, the control bar will be rendered like this:

The *skinObj* is validated before being accepted. If the validation does not pass, warnings will be displayed in the browser's JavaScript console log as to why the *skinObj* was not accepted. If your *skinObj* is having no effect, look for messages with the word `WARNING` in the JavaScript console log. The console log message will indicate which property is not being accepted and why.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

Troubleshooting

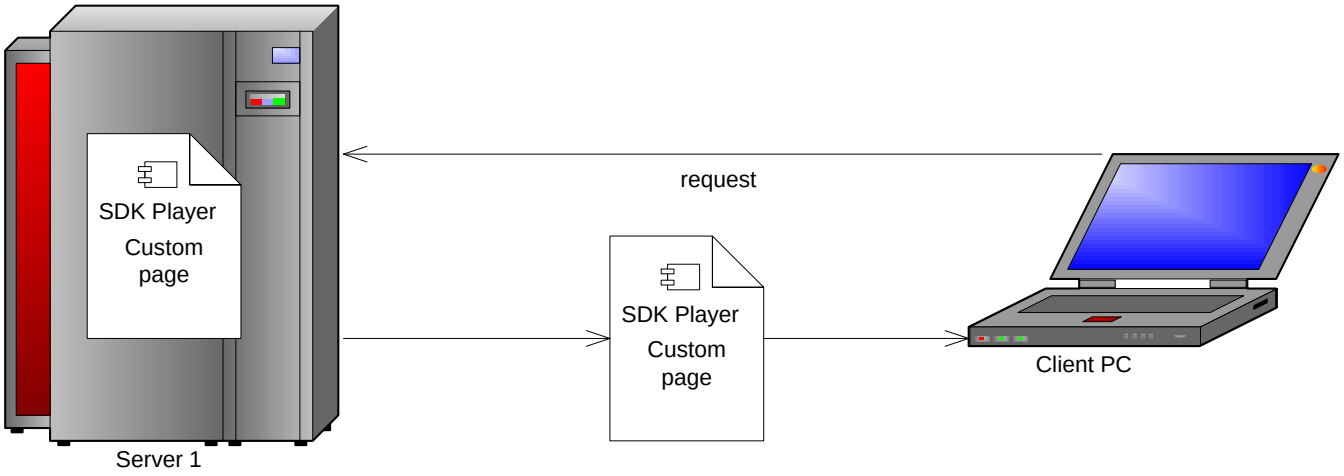
If you are having any issues getting the Player to display video, please check the following:

1. If the video is not appearing at all:
    a. Verify that the *mediaURL* is still valid.
        i. Check that the login session used to acquire *mediaURL* is still valid.
        ii. Check that the *mediaURL* has not expired.
    b. Verify the state of the session from the "Sessions" tab of the Admin UI. Refer to the *Admin UI Setup Guide* for more on the "Sessions" Tab.
2. If you are unable to resolve the problem using the suggestions above, collect the JavaScript console logs from the browser, screenshots of the "Sessions" tab in the Admin UI tool and the *mediaURL*. Then send to Barco Customer Support for further analysis.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

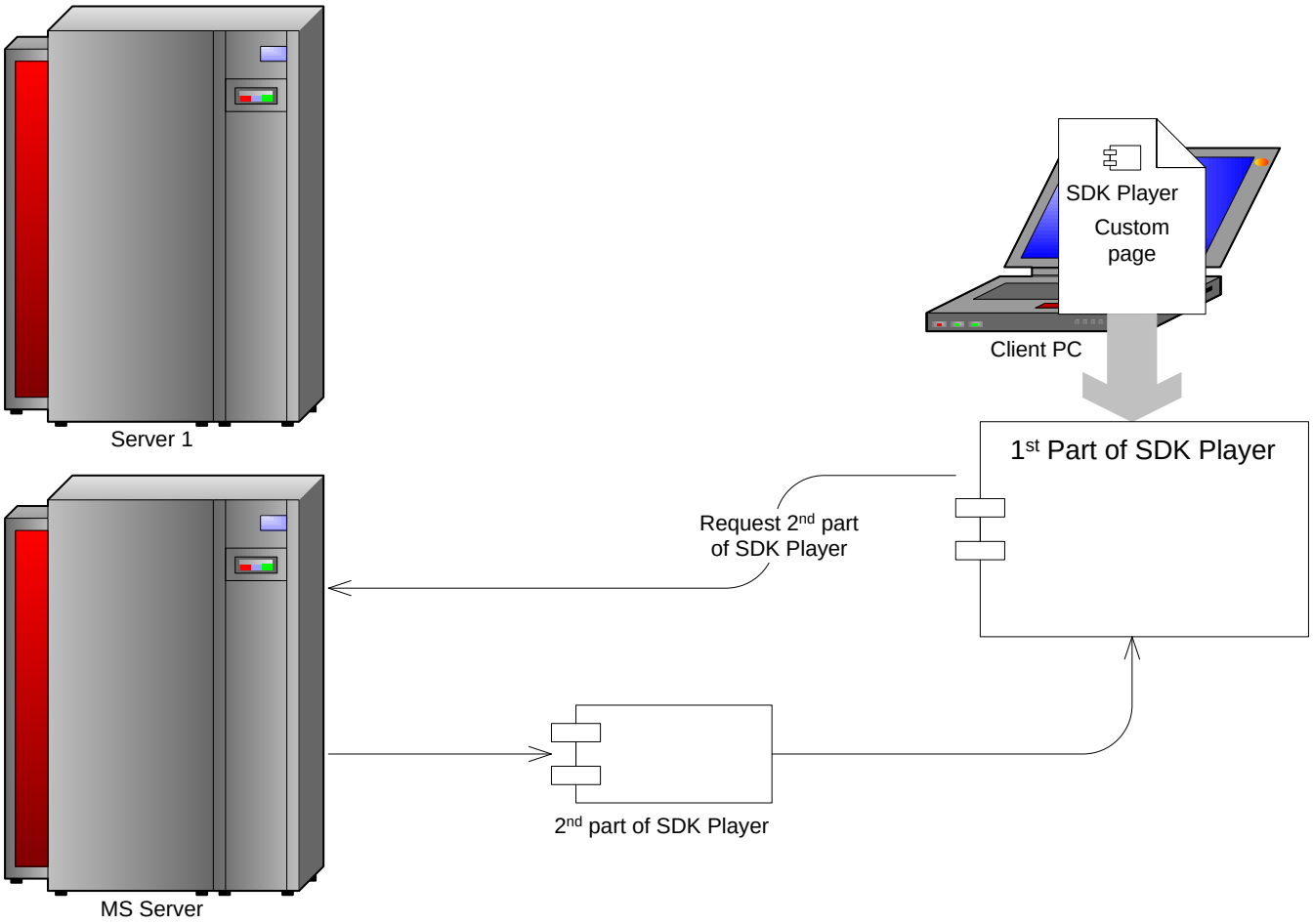
SDK Player Architecture Overview

The SDK Player is actually divided into two parts. This was done to allow the SDK Player to be launched from one server while utilizing video assets controlled by a different server, namely, Barco Management Server. This separation was especially needed for Internet Explorer 9 due to its limited cross-origin access capabilities. The first part of the SDK player is loaded by your custom web page (though the actual SDK Player files might be on a different server or hosted locally).



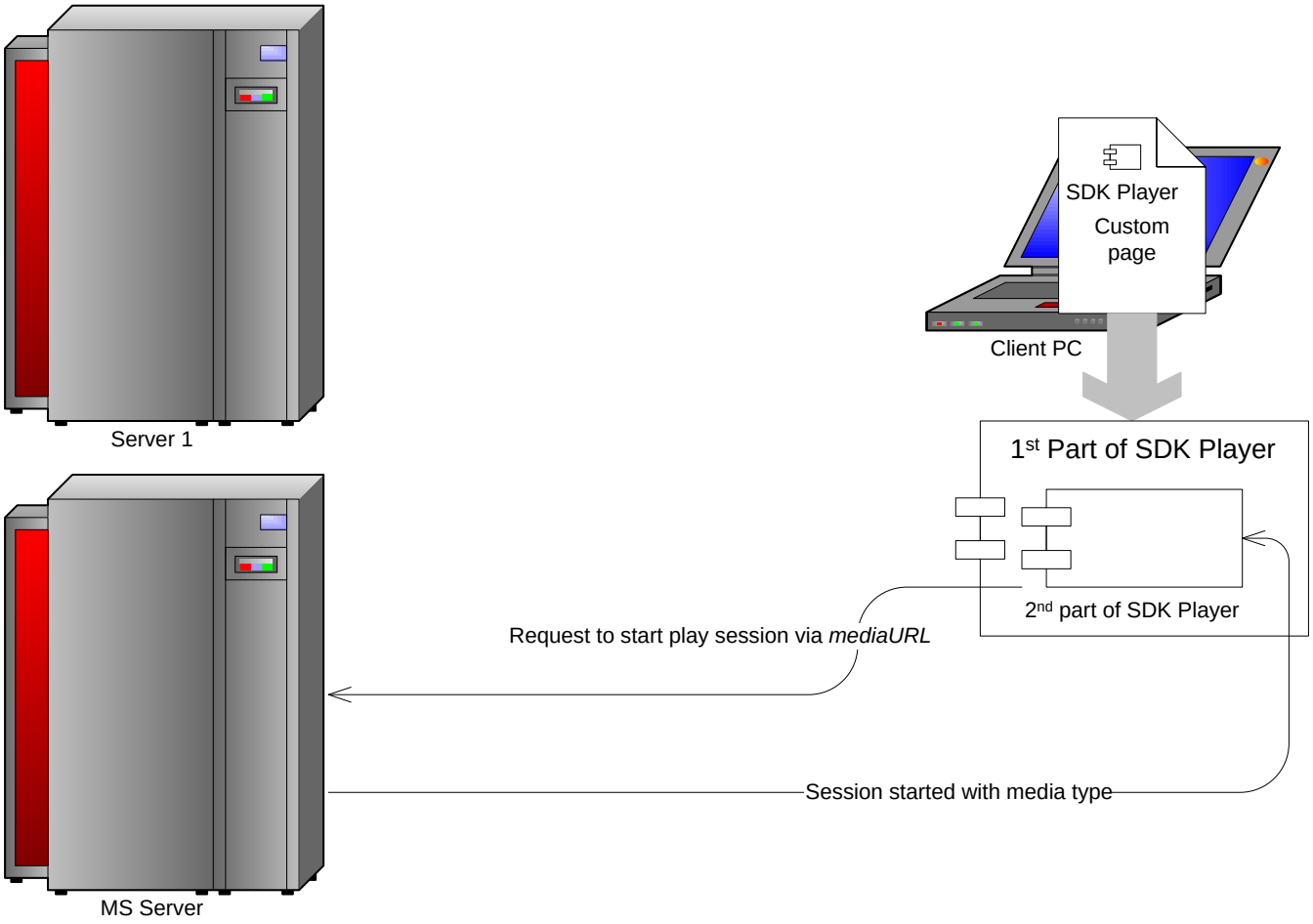
Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

This first part of the SDK Player is included with these SDK sources and you can place these sources wherever you like. Once the custom page with the SDK Player is loaded and asked to play a *mediaURL*, the second part of the SDK Player is loaded by the first part into an iframe from the same MS Server specified by the *mediaURL*.



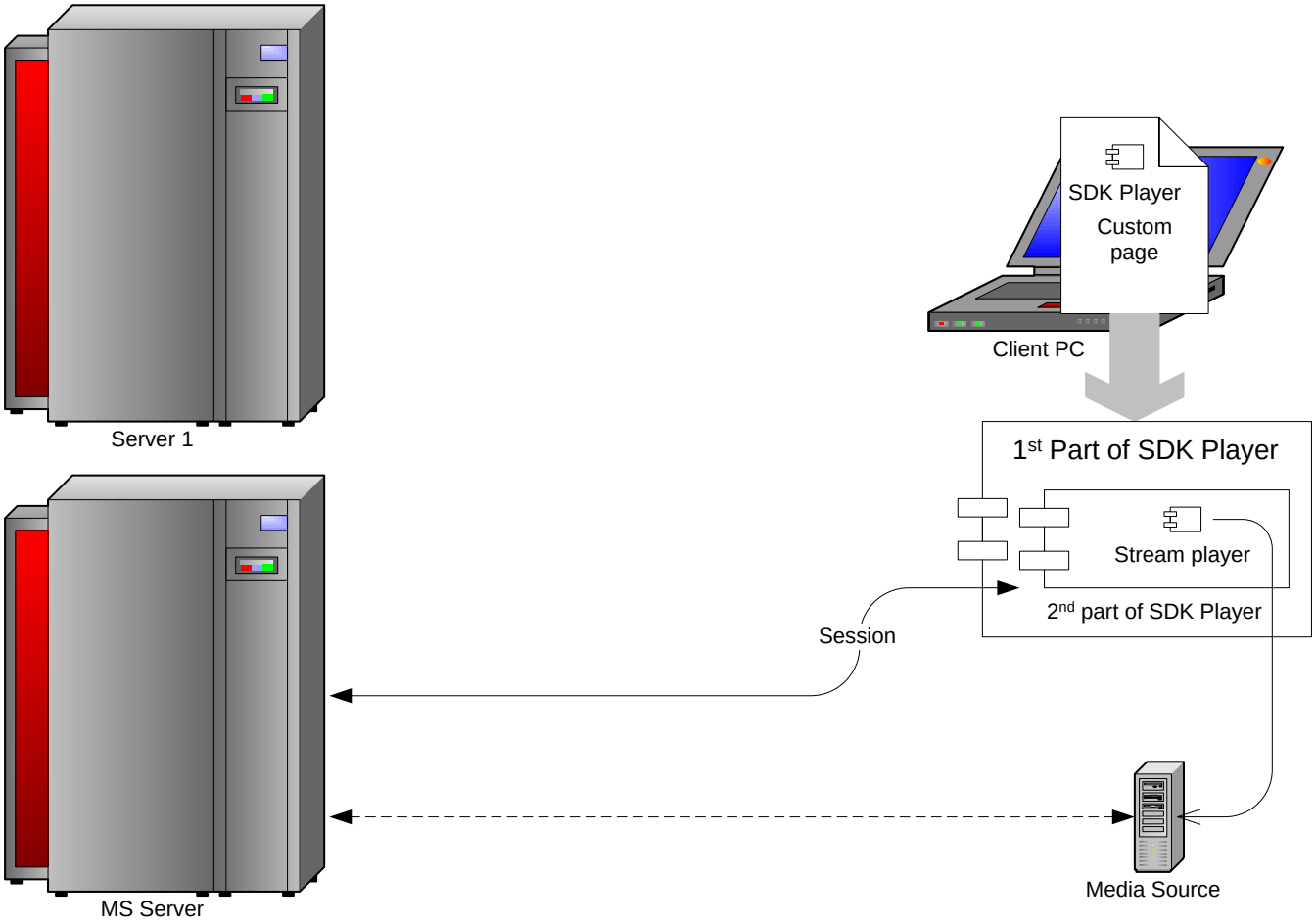
Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

The 2nd part of the SDK Player resides on the MS Server where the *mediaURL* will be sent. This 2nd part is not included with this SDK Player source code. The 2nd part of the SDK Player then makes session requests to the same MS server it was loaded from, thus avoiding cross-origin issues.



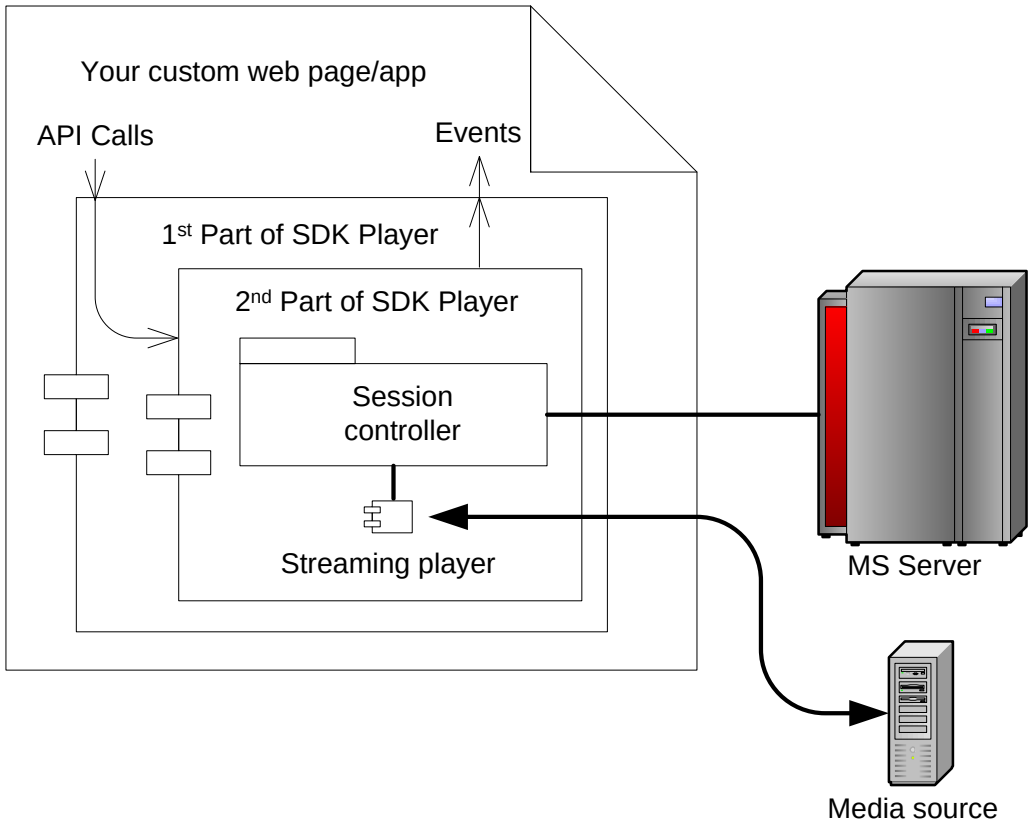
Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

Once the 2nd part of the SDK Player is in a session with the MS Server, it selects the most appropriate low-level streaming video player to load from the same MS Server it was loaded from based on the type of platform, type of media, and the choice of plug-ins or technologies available on the client's web browser. The loaded streaming player then makes direct requests to whatever source contains the most appropriate media and plays the video. Cross-origin issues are not an issue for video playback due to the lower level protocols used (which could be any number of different protocols depending on the streaming player in use).



Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102

A session controller within the 2nd Part of the SDK Player mediates the communications with the MS Server and routes messages to different parts of the SDK player to control playback, scrubbing, recording, bookmarking and error reporting.



API calls to the 1st Part of the SDK Player are sent to the 2nd Part of the SDK Player via a proxy and those calls are routed to either the session controller which talks to the MS Server; or, are routed directly to the streaming player embedded within the 2nd Part of the SDK Player.

Likewise, event notifications from the MS Server or events from the streaming player are published back to the 1st Part of the SDK Player, which then republishes the events to the custom web page/app code hosting the SDK Player.

Barco, Inc.
1287 Anvilwood Ave.
Sunnyvale, Ca 94089
office: +1 408 400 4102