

HLS.js Player

- [Example of stream conversion to HLS and playing it in browser using HLS.js](#)
- [The code of the example](#)
- [Analyzing the code](#)

Example of stream conversion to HLS and playing it in browser using HLS.js

The player shows how to convert stream published on WCS server to HLS and play it in browser. HLS stream cut starts automatically when stream is requested by HLS URL, for example `http://localhost:8082/test/test.m3u8` on the screenshot below

HLS.JS Player Minimal

WCS

http://localhost:8082

Stream

test


Auth

Key

Value

☐ Low latency HLS

Stop



The code of the example

The source code can be accessed on server by the following path:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/hls-js-player

hls-js-player.css -player page styles file
hls-js-player.html - player page
hls.js -player script (<https://github.com/video-dev/hls.js/>, Apache License Version 2.0)
hls-js-player.js - player launch script
hls.min.js - player script (minimized)

The example can be tested using the following URL:

<https://host:8888/client2/examples/demo/streaming/hls-js-player/hls-js-player.html>

Where host is WCS server address

Analyzing the code

To analyze the code get hls-js-player.js file version with hash `ecbadc3` which is available [here](#) and can be downloaded in build [2.0.212](#).

1.A server HLS URL detection

getHLSUrl() [code](#)

```
function initPage() {  
    $("#header").text("HLS.JS Player Minimal");  
    $("#urlServer").val(getHLSUrl());  
    ...  
}
```

2. div element set up to pass to the player

[code](#)

A div element for stream playback is passed to player

```
function initPage() {  
    ...  
    remoteVideo = document.getElementById('remoteVideo');  
    remoteVideo.style = "background-color: lightgrey;";  
}
```

3.Stream name detection (the stream should be published to server)

encodeURIComponent() [code](#)

```
function playBtnClick() {  
    if (validateForm()) {  
        var streamName = $('#playStream').val();  
        streamName = encodeURIComponent(streamName);  
        ...  
    }  
}
```

4.HLS stream URL forming

[code](#)

If authentication key and token are set, they will be included to stream URL

```
function playBtnClick() {
  if (validateForm()) {
    ...
    var videoSrc = $("#urlServer").val() + '/' + streamName + '/' + streamName + '.m3u8';
    var key = $('#key').val();
    var token = $("#token").val();
    if (key.length > 0 && token.length > 0) {
      videoSrc += "?" + key + "=" + token;
    }
    ...
  }
}
```

5. Player starting

[code](#)

If browser does not support MSE, player will not be started and a warning will be displayed

```
function playBtnClick() {
  if (validateForm()) {
    ...
    if (Hls.isSupported()) {
      console.log("Low Latency HLS: "+llHlsEnabled);
      hlsPlayer = new Hls(getHlsConfig(llHlsEnabled));
      hlsPlayer.loadSource(videoSrc);
      hlsPlayer.attachMedia(remoteVideo);
      hlsPlayer.on(Hls.Events.MANIFEST_PARSED, function() {
        console.log("Play with HLS.js");
        remoteVideo.play();
        onStart();
      });
    }
    else {
      $("#notifyFlash").text("Your browser doesn't support MSE technology required to play video");
    }
  }
}
```

6. Playback stopping

[code](#)

```
function stopBtnClick() {
  if (hlsPlayer != null) {
    console.log("Stop HLS segments loading");
    hlsPlayer.stopLoad();
    hlsPlayer = null;
  }
  if (remoteVideo != null) {
    console.log("Stop HTML5 player");
    remoteVideo.pause();
    remoteVideo.currentTime = 0;
    remoteVideo.removeAttribute('src');
    remoteVideo.load();
  }
  onStop();
}
```

7. HLS.js player configuration

[code](#)

```
function getHlsConfig(llHlsEnabled) {
  var config = {
    lowLatencyMode: false,
    enableWorker: true,
    backBufferLength: 90
  };
  if(llHlsEnabled) {
    // Here we configure HLS.JS for lower latency
    config = {
      lowLatencyMode: llHlsEnabled,
      enableWorker: true,
      backBufferLength: 90,
      liveBackBufferLength: 0,
      liveSyncDuration: 0.5,
      liveMaxLatencyDuration: 5,
      liveDurationInfinity: true,
      highBufferWatchdogPeriod: 1,
    };
  }
  return config;
}
```