

# HLS.js Player

- [Пример конвертации видеопотока в HLS и отображения его в браузере при помощи HLS.js](#)
- [Код примера](#)
- [Работа с кодом примера](#)

## Пример конвертации видеопотока в HLS и отображения его в браузере при помощи HLS.js

Данный плеер демонстрирует возможности WCS по преобразованию опубликованного на сервере потока в HLS и воспроизведению его в браузере. Нарезка потока в HLS запускается автоматически, при обращении к потоку, опубликованному на сервере, по HLS URL, например, для потока на рисунке ниже <http://localhost:8082/test/test.m3u8>

### HLS.JS Player Minimal


**WCS**

**Stream**

**Auth**

Key	Value
-----	-------

Low latency HLS



# Код примера

Код данного примера находится на сервере по следующему пути:

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

hls-js-player.css - файл стилей страницы с плеером

hls-js-player.html - страница с плеером

hls.js -скрипт, обеспечивающий работу плеера (<https://github.com/video-dev/hls.js/>, Apache License Version 2.0)

hls-js-player.js - скрипт, обеспечивающий запуск плеера

hls.min.js - скрипт, обеспечивающий работу плеера (минимизированная версия)

Тестировать данный пример можно по следующему адресу:

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

Здесь host - адрес вашего WCS-сервера.

## Работа с кодом примера

Для разбора кода возьмем версию файла hls-js-player.js с хешем ecbadc3, которая находится [здесь](#) и доступна для скачивания в соответствующей сборке [2.0.212](#).

### 1. Определение HLS URL сервера

getHLSUrl() [code](#)

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

### 2. Настройка div элемента для передачи плееру

[code](#)

Плееру передается имя div-элемента, в котором должен быть проигран поток.

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

### 3. Определение имени потока (должен быть опубликован на сервере)

encodeURIComponent() [code](#)

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

### 4. Формирование URL HLS-потока

[code](#)

Если указаны ключ и токен авторизации, они будут включены в URL потока

```

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. Запуск плеера

[code](#)

Если браузер не поддерживает технологию MSE, плеер не запустится, будет выведено предупреждение

```

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. Остановка воспроизведения

[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. Настройка HLS.js плеера

[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;
}
```