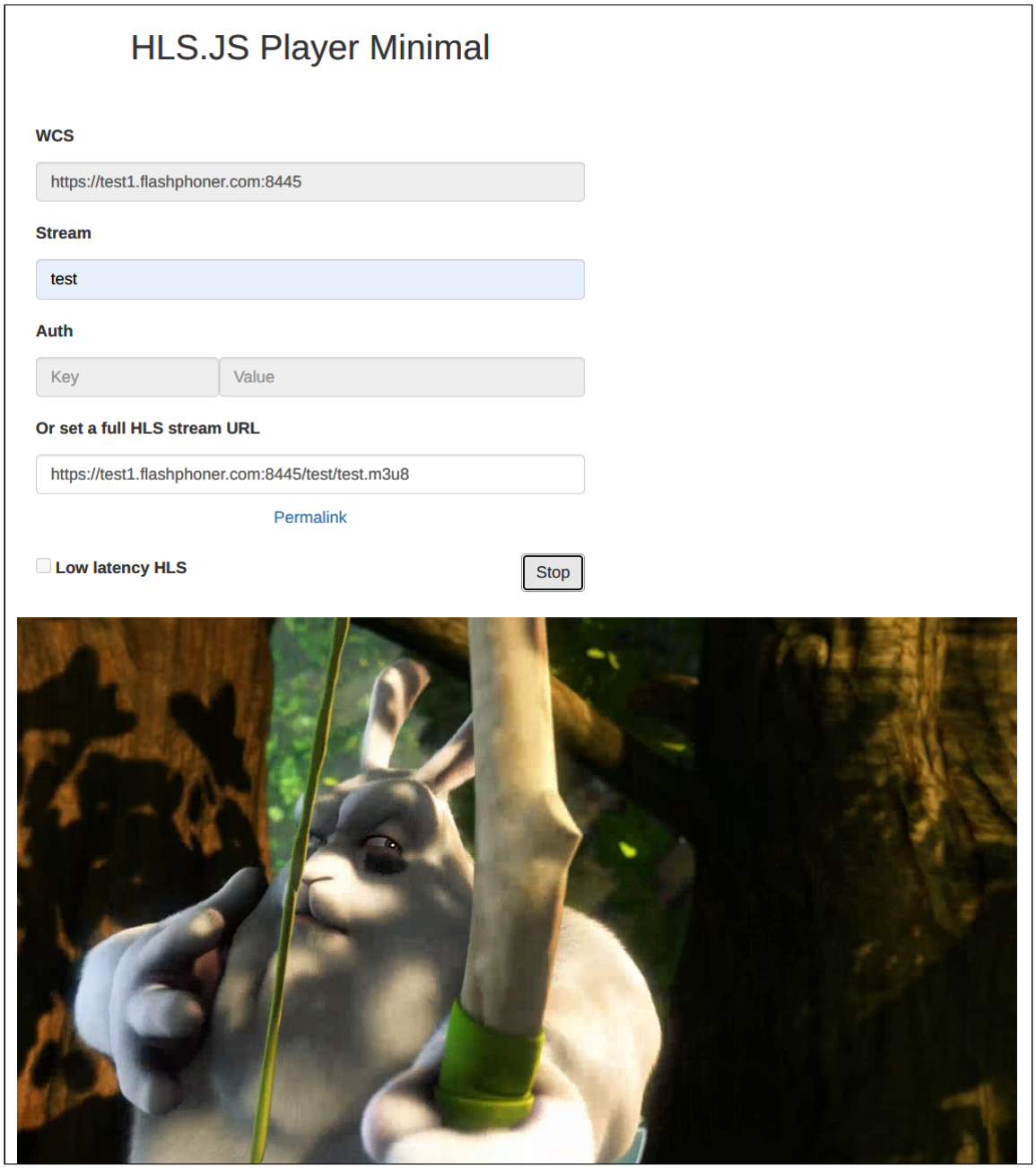
HLS.js Player

Описание

Данный пример демонстрирует возможности WCS по преобразованию опубликованного на сервере потока в HLS и воспроизведению его в браузере при помощи библиотеки HLS.js. Нарезка потока в HLS запускается автоматически, при обращении к потоку, опубликованному на сервере, по HLS URL, например, для потока на рисунке ниже `https://test1.flashphoner.com:8445/test/test.m3u8`



Начиная со сборки 2.0.244, пример поддерживает следующие параметры:

- `llhls` - включает поддержку Low Latency HLS: `false` или `true`
- `src` - полный HLS URL потока для проигрывания, должен быть закодирован при помощи URI encode, например `https%3A%2F%2Ftest1.flashphoner.com%3A8445%2Ftest%2Ftest.m3u8`
- `autoplay` - автоматически запустить проигрывание указанного HLS URL, при этом все поля ввода и кнопки скрываются: `false` (по умолчанию) или `true`

Пример открытия плеера с параметрами, как на скриншоте выше (ссылка в поле `Permalink`)

```
https://test1.flashphoner.com:8444/client2/examples/demo/streaming/hls-js-player/hls-js-player.html?llhls=false&src=https%3A%2F%2Ftest1.flashphoner.com%3A8445%2Ftest%2Ftest.m3u8
```

Пример вызова плеера с автозапуском

```
https://test1.flashphoner.com:8444/client2/examples/demo/streaming/hls-js-player/hls-js-player.html?llhls=false&src=https%3A%2F%2Ftest1.flashphoner.com%3A8445%2Ftest%2Ftest.m3u8&au
```

При этом воспроизведение потока будет запущено автоматически, с отключенным звуком. Для включения звука зритель должен использовать кнопку регулятора громкости в интерфейсе плеера.

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)
- `../hls-player/player-page.html` - общая страница плеера для трех HLS примеров

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 `1703e13`, которая находится здесь и доступна для скачивания в соответствующей сборке 2.0.244.

1. Загрузка страницы плеера

code

```
const loadPlayerPage = function() {
    loadPage("../hls-player/player-page.html", "playerPage", initPage );
}
```

2. Инициализация HTML-страницы плеера

code

Если браузер не поддерживает технологию MSE, плеер не будет инициализирован, и будет выведено предупреждение

```
const initPage =  function() {
    if (playSrc) {
        setValue("fullLink", decodeURIComponent(playSrc));
    } else if (autoplay) {
        console.warn("No HLS URL set, autoplay disabled");
        autoplay = false;
    }
    if (llHlsEnabled) {
        setCheckbox("llHlsEnabled", llHlsEnabled);
    }
    remoteVideo = document.getElementById('remoteVideo');
    if (Hls.isSupported()) {
        console.log("Using HLS.JS " + Hls.version);
        if (autoplay) {
            // There should not be any visible item on the page unless player
            hideAllToAutoplay();
            // The player should use all available page width
            setUpPlayerItem(true);
            // The player should be muted to automatically start playback
            initVideoPlayer(remoteVideo, true);
            playBtnClick();
        } else {
            setText("header", "HLS.JS Player Minimal");
```

```
            showItem("llHlsMode");
            displayCommonItems();
            setUpButtons();
            enablePlaybackStats();
            // The player should have a maximum fixed size
            setUpPlayerItem(false);
            // The player can be unmuted because user should click Play
button
            initVideoPlayer(remoteVideo, false);
        }
    } else {
        setText("notifyFlash", "Your browser doesn't support MSE technology
required to play video");
        disableItem("applyBtn");
        toggleInputs(false);
    }
}
```

3. Инициализация видео элемента для проигрывания

code

```
const initVideoPlayer = function(video, muted) {
    if (video) {
        video.style.backgroundColor = "black";
        video.muted = muted;
    }
}
```

4. Формирование URL HLS-потока

code

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

```
const getVideoSrc = function(src) {
    let videoSrc = src;
    if (validateForm()) {
        let streamName = getValue('playStream');
        streamName = encodeURIComponent(streamName);
        videoSrc = getValue("urlServer") + '/' + streamName + '/' +
streamName + '.m3u8';
        let key = getValue('key');
        let token = getValue("token");
        if (key.length > 0 && token.length > 0) {
            videoSrc += "?" + key + "=" + token;
        }
    }
    setValue("fullLink", videoSrc);
    return videoSrc;
}
```

5. Настройка HLS.js плеера

code

```
const getHlsConfig = function(llHlsEnabled) {
    let config = {
        lowLatencyMode: false,
        enableWorker: true,
        backBufferLength: 90,
        manifestLoadingTimeOut: 15000
    };
    console.log("Low Latency HLS: "+llHlsEnabled)
    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,
           manifestLoadingTimeOut: 15000
        };
    }
    return config;
}
```

6. Запуск плеера

code

```
const playBtnClick = function() {
    let videoSrc = getVideoSrc(getValue("fullLink"));
    if (videoSrc) {
        llHlsEnabled = getCheckbox("llHlsEnabled");
        hlsPlayer = new Hls(getHlsConfig(llHlsEnabled));
        hlsPlayer.on(Hls.Events.MANIFEST_PARSED, function() {
            console.log("Play with HLS.js");
            remoteVideo.play();
        });
        remoteVideo.onplaying = () => {
            console.log("playing event fired");
            displayPermalink(videoSrc);
        }
        hlsPlayer.loadSource(videoSrc);
        hlsPlayer.attachMedia(remoteVideo);
        onStarted();
    }
}
```

7. Остановка воспроизведения

code

```
const stopBtnClick = function() {
    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();
    }
    onStopped();
}
```

8. Получение доступной статистики воспроизведения из HTML5 video элемента

HTML5Stats code

```
const PlaybackStats = function(interval) {
    const playbackStats = {
        interval: interval || STATS_INTERVAL,
        timer: null,
        stats: null,
        start: function() {
            let video = remoteVideo;

            playbackStats.stop();
            stats = HTML5Stats(video);
            playbackStats.timer = setInterval(playbackStats.displayStats,
playbackStats.interval);
            setText("videoWidth", "N/A");
            setText("videoHeight", "N/A");
            setText("videoRate", "N/A");
            setText("videoFps", "N/A");
            showItem("stats");
        },
        stop: function() {
            if (playbackStats.timer) {
                clearInterval(playbackStats.timer);
                playbackStats.timer = null;
            }
            playbackStats.stats = null;
            hideItem("stats");
        },
        displayStats: function() {
            if (stats.collect()) {
                let width = stats.getWidth();
                let height = stats.getHeight();
                let bitrate = stats.getBitrate();
```

```
                let fps = stats.getFps();

                setText("videoWidth", width);
                setText("videoHeight", height);

                if (bitrate !== undefined) {
                    setText("videoRate", Math.round(bitrate));
                }
                if (fps !== undefined) {
                    setText("videoFps", fps.toFixed(1));
                }
            }
        }
    };
    return playbackStats;
}
```