

In a browser with Delight Player

- [Overview](#)
 - [Supported platforms and browsers](#)
 - [Supported technologies](#)
- [Playback via WebRTC](#)
 - [Using WebSDK](#)
 - [Testing](#)
 - [Player page example code](#)
 - [Using JavaScript and HTML5](#)
 - [Testing](#)
 - [Player page example code](#)
- [Playback via HLS](#)
 - [Testing](#)
 - [Player page example code](#)
- [Known issues](#)

Overview

A stream published on WCS server can be played in a browser-based VR player, [Delight Player](#) for example. This way stream can be played in virtual and mixed reality devices if one of browsers supported works on this device. Note that the better quality of stream published, the better stream playback quality in VR device.

Supported platforms and browsers

	Chrome	Firefox	Safari 11	Edge
Windows	+	+		+
Mac OS	+	+	+	
Android	+	+		
iOS	-	-	+	

Supported technologies

- WebRTC
- HLS

Playback via WebRTC

Delight Player can be integrated with WCS to play the stream by two ways:

1. Using WebSDK
2. Using JavaScript and HTML5

Using WebSDK

To play a stream via WebRTC in Delight Player or any other custom JavaScript player, the video element of the page in which a stream will be played should be passed as `remoteVideo` parameter to the `session.createStream()` WebSDK function

`session.createStream()` code

```
session.createStream({
  name: document.getElementById('playStream').value,
  display: display,
  remoteVideo: video
})
...
```

Testing

1. For the test we use:

- WCS server
- [Media Devices](#) web application to publish FullHD stream
- [Delight](#) VR player to play the stream

2. Set the stream resolution to 1920x1080

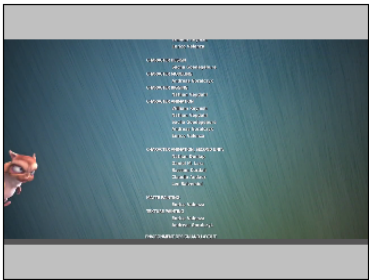
Screen share	☐ off	
Size	1920	1080
FPS	30	
Bitrate	min 0	max 0

3. Set to WCS field stream namewss://test1.flashphoner.com:8443/testand press Start to publish

Media Devices

Video stats
Bytes sent: 8460431
Packets sent: 7719
Frames encoded: 1274
Audio stats
Bytes sent: 200204
Packets sent: 2237

Local

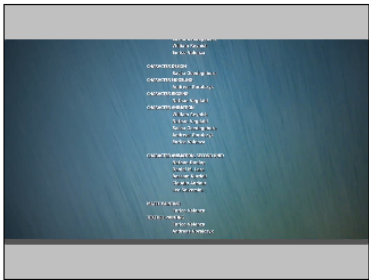


1920x1080

WCS

PUBLISHING

Preview



1920x1080

Video stats
Bytes received: 8422719
Packets received: 6870
Frames decoded: 1247
Audio stats
Bytes received: 197275
Packets received: 2202

4. Play the stream test in VR player



A view angle can be changed with mouse drag in browser on PC or Mac, and can be changed depending on iOS device or dedicated VR device tilt angle.

Player page example code

1. Declaration of video element to play the stream, stream name input field and Play/Stop buttons

```
<div style="width: 50%;" id="display">
  <dl8-live-video id="remoteVideo" format="STEREO_TERPON">
    <source>
  </dl8-live-video>
</div>
<input class="form-control" type="text" id="playStream" placeholder="Stream Name">
<button id="playBtn" type="button" class="btn btn-default" disabled>Play</button>
<button id="stopBtn" type="button" class="btn btn-default" disabled>Stop</button>
```

2. Player readiness event handling

```
document.addEventListener('x-dl8-evt-ready', function () {
  dl8video = document.getElementById('remoteVideo');
  $('#playBtn').prop('disabled', false).click(function() {
    playStream();
  });
});
```

3. Establishing connection with a server and stream creation

```

        var video = dl8video.contentElement;
        Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function
(session) {
            var session = Flashphoner.getSessions()[0];
            session.createStream({
                name: document.getElementById('playStream').value,
                display: display,
                remoteVideo: video
            }).on(STREAM_STATUS.PLAYING, function (stream) {
                ...
            }).play();
        })
    }
}

```

4. Playback start in VR player and Stop button handling

```

        ...
    }).on(STREAM_STATUS.PLAYING, function (stream) {
        dl8video.start();
        $('#stopBtn').prop('disabled', false).click(function() {
            $('#playBtn').prop('disabled', false);
            $('#stopBtn').prop('disabled', true);
            stream.stop();
            dl8video.exit();
        });
    }).play();
})

```

Full source code of the sample VR player page

Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>WebRTC Delight</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script type="text/javascript" src="../../../../flashphoner.js"></script>
    <script type="text/javascript" src="../../dependencies/jquery/jquery-1.12.0.js"></script>
    <script type="text/javascript" src="../../dependencies/js/utls.js"></script>
    <script src="dl8-66b250447635476d123a44a391c80b09887e831e.js" async></script>
    <meta name="dl8-custom-format" content='{ "name": "STEREO_TERPON", "base": "STEREO_MESH", "params": { "uri":
"03198702.json" } }'>
  </head>
  <body>
    <div style="width: 50%;" id="display">
      <dl8-live-video id="remoteVideo" format="STEREO_TERPON">
        <source>
      </dl8-live-video>
    </div>
    <input class="form-control" type="text" id="playStream" placeholder="Stream Name">
    <button id="playBtn" type="button" class="btn btn-default" disabled>Play</button>
    <button id="stopBtn" type="button" class="btn btn-default" disabled>Stop</button>
    <script>
      Flashphoner.init({flashMediaProviderSwfLocation: '../../../../media-provider.swf'});
      var SESSION_STATUS = Flashphoner.constants.SESSION_STATUS;
      var STREAM_STATUS = Flashphoner.constants.STREAM_STATUS;
      var STREAM_STATUS_INFO = Flashphoner.constants.STREAM_STATUS_INFO;
      var playBtn = document.getElementById('playBtn');
      var display = document.getElementById('display');
      var dl8video = null;
      var url = setURL();
      document.addEventListener('x-dl8-evt-ready', function () {
        dl8video = document.getElementById('remoteVideo');
        $('#playBtn').prop('disabled', false).click(function() {
          playStream();
        });
      });
      function playStream() {
        $('#playBtn').prop('disabled', true);
        $('#stopBtn').prop('disabled', false);
        var video = dl8video.contentElement;
        Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function
(session) {
          var session = Flashphoner.getSessions()[0];
          session.createStream({
            name: document.getElementById('playStream').value,
            display: display,
            remoteVideo: video
          }).on(STREAM_STATUS.PLAYING, function (stream) {
            dl8video.start();
            $('#stopBtn').prop('disabled', false).click(function() {
              $('#playBtn').prop('disabled', false);
              $('#stopBtn').prop('disabled', true);
              stream.stop();
              dl8video.exit();
            });
          }).play();
        })
      }
    </script>
  </body>
</html>
```

To play a stream via WebRTC in Delight Player or any other custom JavaScript player, a mock element for displaying the stream should be created

```
var mockRemoteDisplay = $('<div></div>');
var mockRemoteVideo = $('<video></video>', {id: 'mock-REMOTE_CACHED_VIDEO'});
mockRemoteDisplay.append(mockRemoteVideo);
```

Mock element `mockRemoteDisplay` is passed as `display` parameter to `WebSDKSession.createStream()` function, and `mockRemoteVideo` element is passed as stream source to VR player

`session.createStream()` code

```
session.createStream({
    name: $('#streamName').val(),
    display: mockRemoteDisplay.get(0)
}).on(STREAM_STATUS.PLAYING, function (stream) {
    var srcObject = mockRemoteVideo.get(0).srcObject;
    video.srcObject = srcObject;
    dl8video.start();
    ...
}).play();
```

Testing

1. For the test we use:

- WCS server
- [Media Devices](#) web application to publish FullHD stream
- [Delight](#) VR player to play the stream

2. Set the stream resolution to 1920x1080

Screen share	☐ off	
Size	1920	1080
FPS	30	
Bitrate	min	max
	0	0

3. Set to WCS field stream name `wss://test1.flashphoner.com:8443/test` and press Start to publish

Media Devices

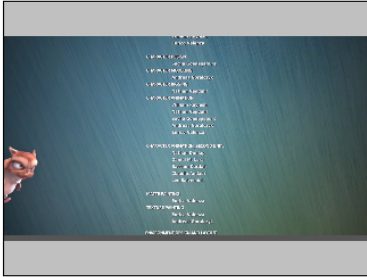
Video stats

Bytes sent: 8460431
Packets sent: 7719
Frames encoded: 1274

Audio stats

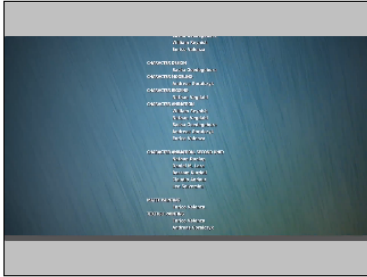
Bytes sent: 200204
Packets sent: 2237

Local



1920x1080

Preview



1920x1080

Video stats

Bytes received: 8422719
Packets received: 6870
Frames decoded: 1247

Audio stats

Bytes received: 197275
Packets received: 2202

WCS

wss://test1.flashphoner.com:8443

PUBLISHING

4. Play the stream test in VR player



test

Play Stop

Player page example code

1. Declaration of video element to play the stream, stream name input field and Play/Stop buttons

```

<div style="width: 50%;">
  <dl8-live-video id="remoteVideo" format="STEREO_TERPON" muted="true">
    <source>
  </dl8-live-video>
</div>
  <input class="form-control" type="text" id="streamName" placeholder="Stream Name">
  <button id="playBtn" type="button" class="btn btn-default" disabled>Play</button>
  <button id="stopBtn" type="button" class="btn btn-default" disabled>Stop</button>

```

2. Player readiness event handling

```

document.addEventListener('x-dl8-evt-ready', function () {
    dl8video = $('#remoteVideo').get(0);
    $('#playBtn').prop('disabled', false).click(function() {
        publishStream();
    });
});

```

3. Creating mock elements to play a stream

```

var mockRemoteDisplay = $('<div></div>');
var mockRemoteVideo = $('<video></video>', {id: 'mock-REMOTE_CACHED_VIDEO'});
mockRemoteDisplay.append(mockRemoteVideo);

```

4. Establishing connection to the server and stream creation

```

    var video = dl8video.contentElement;
    Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function
(session) {

    var session = Flashphoner.getSessions()[0];
    session.createStream({
        name: $('#streamName').val(),
        display: mockRemoteDisplay.get(0)
    }).on(STREAM_STATUS.PLAYING, function (stream) {

        ...

    }).play();
    })

```

5. Playback start in VR player and Stop button handling

```

    ...
    }).on(STREAM_STATUS.PLAYING, function (stream) {
        var srcObject = mockRemoteVideo.get(0).srcObject;
        video.srcObject = srcObject;
        dl8video.start();
        mockRemoteVideo.get(0).pause();
        mockRemoteVideo.get(0).srcObject = null;
        $('#stopBtn').prop('disabled', false).click(function() {
            stream.stop();
            $('#playBtn').prop('disabled', false);
            $('#stopBtn').prop('disabled', true);
            dl8video.exit();
        });
    }).play();

```

Full source code of the sample VR player page

Code

```

<!DOCTYPE html>
<html>
  <head>
    <title>WebRTC Delight</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script type="text/javascript" src="../../../../flashphoner.js"></script>
    <script type="text/javascript" src="../../dependencies/jquery/jquery-1.12.0.js"></script>
    <script type="text/javascript" src="../../dependencies/js/Utils.js"></script>
    <script src="dl8-66b250447635476d123a44a391c80b09887e831e.js" async></script>
    <meta name="dl8-custom-format" content='{ "name": "STEREO_TERPON", "base": "STEREO_MESH", "params": { "uri":
"03198702.json" } }'>
  </head>
  <body>
    <div style="width: 50%;">
      <dl8-live-video id="remoteVideo" format="STEREO_TERPON" muted="true">
        <source>
      </dl8-live-video>
    </div>
    <input class="form-control" type="text" id="streamName" placeholder="Stream Name">
    <button id="playBtn" type="button" class="btn btn-default" disabled>Play</button>
    <button id="stopBtn" type="button" class="btn btn-default" disabled>Stop</button>
    <script>
      Flashphoner.init({flashMediaProviderSwfLocation: '../../media-provider.swf'});
      var SESSION_STATUS = Flashphoner.constants.SESSION_STATUS;
      var STREAM_STATUS = Flashphoner.constants.STREAM_STATUS;
      var STREAM_STATUS_INFO = Flashphoner.constants.STREAM_STATUS_INFO;
      var playBtn = $('#playBtn').get(0);
      var dl8video = null;
      var url = setURL();
      document.addEventListener('x-dl8-evt-ready', function () {
        dl8video = $('#remoteVideo').get(0);
        $('#playBtn').prop('disabled', false).click(function() {
          publishStream();
        });
      });

      var mockRemoteDisplay = $('<div></div>');
      var mockRemoteVideo = $('<video></video>', {id: 'mock-REMOTE_CACHED_VIDEO'});
      mockRemoteDisplay.append(mockRemoteVideo);
      function publishStream() {
        $('#playBtn').prop('disabled', true);
        $('#stopBtn').prop('disabled', false);
        var video = dl8video.contentElement;
        Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function
(session) {

          var session = Flashphoner.getSessions()[0];
          session.createStream({
            name: $('#streamName').val(),
            display: mockRemoteDisplay.get(0)
          }).on(STREAM_STATUS.PLAYING, function (stream) {
            var srcObject = mockRemoteVideo.get(0).srcObject;
            video.srcObject = srcObject;
            dl8video.start();
            mockRemoteVideo.get(0).pause();
            mockRemoteVideo.get(0).srcObject = null;
            $('#stopBtn').prop('disabled', false).click(function() {
              stream.stop();
              $('#playBtn').prop('disabled', false);
              $('#stopBtn').prop('disabled', true);
              dl8video.exit();
            });
          });
        }).play();
      }
    </script>
  </body>
</html>

```

Playback via HLS

When problem occurs with stream playback in Delight Player via WebRTC, stream can be played via HLS

Testing

1. For the test we use:

- WCS server
- [Media Devices](#) web application to publish FullHD stream
- [Delight](#) VR player to play the stream

2. Set the stream resolution to 1920x1080

The screenshot shows the Media Devices configuration interface. It has four rows of controls:

- Screen share**: A toggle switch set to "off".
- Size**: Two input fields, the first containing "1920" and the second containing "1080".
- FPS**: A single input field containing "30".
- Bitrate**: Two input fields labeled "min" and "max", both containing "0".

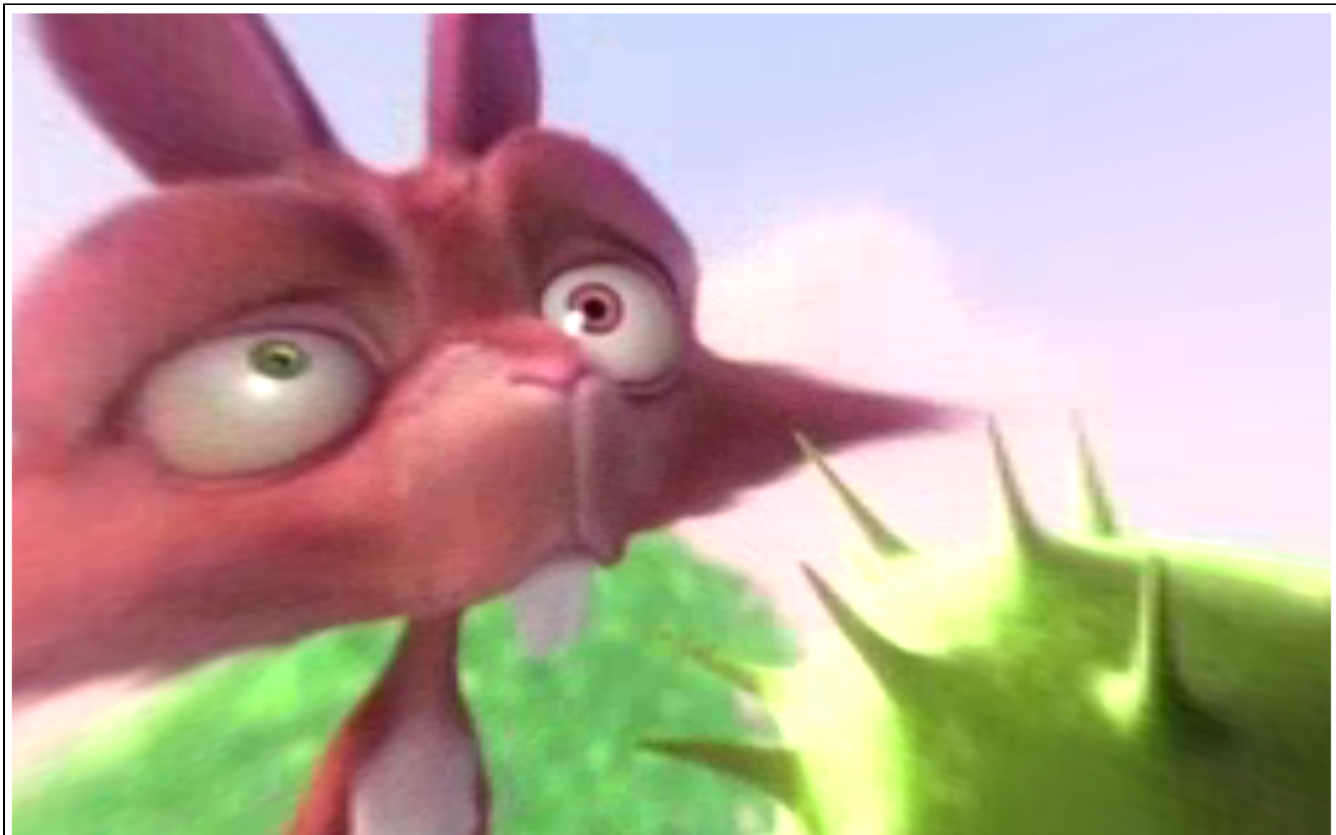
3. Set to WCS field stream name `wss://test1.flashphoner.com:8443/test` and press Start to publish

The screenshot shows the Media Devices application interface. It features two video feeds: "Local" on the left and "Preview" on the right, connected by a right-pointing arrow. Below the feeds is a "WCS" field with the URL `wss://test1.flashphoner.com:8443/test` and a "PUBLISHING" button. Statistics are displayed on both sides of the feeds.

Local (Video stats)	Preview (Video stats)
Bytes sent: 8460431	Bytes received: 8422719
Packets sent: 7719	Packets received: 6870
Frames encoded: 1274	Frames decoded: 1247

Local (Audio stats)	Preview (Audio stats)
Bytes sent: 200204	Bytes received: 197275
Packets sent: 2237	Packets received: 2202

4. Play the stream `test` in VR player



test

Play

Stop

Player page example code

1. Declaration of video element to play the stream, stream name input field and Play/Stop buttons

```
<div style="width: 50%;" id="display">
  <dl8-live-video id="remoteVideo" format="MONO_360">
    <source type="application/x-mpegurl" id="hlsSource"/>
  </dl8-live-video>
</div>
<input class="form-control" type="text" id="playStream" placeholder="Stream Name">
<button id="playBtn" type="button" class="btn btn-default" disabled>Play</button>
<button id="stopBtn" type="button" class="btn btn-default" disabled>Stop</button>
```

2. Player readiness event handling

```
document.addEventListener('x-dl8-evt-ready', function () {
  dl8video = document.getElementById('remoteVideo');
  $('#playBtn').prop('disabled', false).click(playStream);
});
```

3. Getting server URL to play via HLS

```
var hlsUrl = getHLSUrl();
```

4. Playback start in VR player and Stop button handling

```

...
var video = dl8video.contentElement;
var streamName = document.getElementById('playStream').value;
$('#hlsSource').attr("src",hlsUrl + "/" + streamName + "/" + streamName + ".m3u8");
dl8video.start();
$('#stopBtn').prop('disabled', false).click(function() {
    $('#playBtn').prop('disabled', false);
    $('#stopBtn').prop('disabled', true);
    dl8video.exit();
});

```

Full source code of the sample VR player page

Code

```

<!DOCTYPE html>
<html>
  <head>
    <title>WebRTC Delight</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script type="text/javascript" src="../../../../flashphoner.js"></script>
    <script type="text/javascript" src="../../dependencies/jquery/jquery-1.12.0.js"></script>
    <script type="text/javascript" src="../../dependencies/js/Utils.js"></script>
    <script src="dl8-66b250447635476d123a44a391c80b09887e831e.js" async></script>
  </head>
  <body>
    <div style="width: 50%;" id="display">
      <dl8-live-video id="remoteVideo" format="MONO_360">
        <source type="application/x-mpegurl" id="hlsSource"/>
      </dl8-live-video>
    </div>
    <input class="form-control" type="text" id="playStream" placeholder="Stream Name">
    <button id="playBtn" type="button" class="btn btn-default disabled>Play</button>
    <button id="stopBtn" type="button" class="btn btn-default disabled>Stop</button>

    <script>
      var playBtn = document.getElementById('playBtn');
      var display = document.getElementById('display');
      var dl8video = null;
      var hlsUrl = getHLSUrl();
      document.addEventListener('x-dl8-evt-ready', function () {
        dl8video = document.getElementById('remoteVideo');
        $('#playBtn').prop('disabled', false).click(playStream);
      });

      function playStream() {
        $('#playBtn').prop('disabled', true);
        $('#stopBtn').prop('disabled', false);
        var video = dl8video.contentElement;
        var streamName = document.getElementById('playStream').value;
        $('#hlsSource').attr("src",hlsUrl + "/" + streamName + "/" + streamName + ".m3u8");
        dl8video.start();
        $('#stopBtn').prop('disabled', false).click(function() {
          $('#playBtn').prop('disabled', false);
          $('#stopBtn').prop('disabled', true);
          dl8video.exit();
        });
      }

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

```

Known issues

1. A stream is not played in Delight Player via WebRTC or is played with freezes.

Symptoms: a stream is not played at all (MS Edge) or is played with constant freezes (iOS Safari)

Solution: use HLS to play the stream

2. VR view does not work in Delight Player via HLS in MS Edge on Windows 10 Mobile.

Symptoms: the stream is played in Delight Player via HLS, but the picture is flat

Solution: use device on actual platform with more browsers supported.

3. A stream is not played in Delight Player via HLS.

Symptoms: a stream is not played in Delight Player, download indicator shows 99%, then black screen, or CORS error is displayed

Solution: use [nginx as reverse proxy](#) to play HLS stream in Safari browser