In a browser via Websocket + Canvas, WSPlayer

- Overview
  - Supported platforms and browsers
  - Supported codecs
  - Operation flowchart
- Quick manual on testing
  - Publishing a video stream on the server and playing it using HTML5 + Canvas (WSPlayer) in a browser
- Call flow
- Known issues

The combination of WebSocket + HTML5 Canvas technologies is recommended to use for playing a video stream if a client browser does not support WebRTC and latency must be minimal. You can find thorough description of the WSPlayer player based on these technologies in this article.

Overview

Not all browsers have support for WebRTC. For example, the main method to deliver a live video stream to the Safari browser under iOS 9 and iOS 10 is HLS (HTTP Live Streaming). This protocol can lead to latencies more than 15 seconds.

Web Call Server sends a video stream to a browser via the Websocket protocol, which allows reducing latency down to 1-3 seconds and produces real time video in comparison with HLS. The stream is played in a browser using the Canvas HTML5 element.

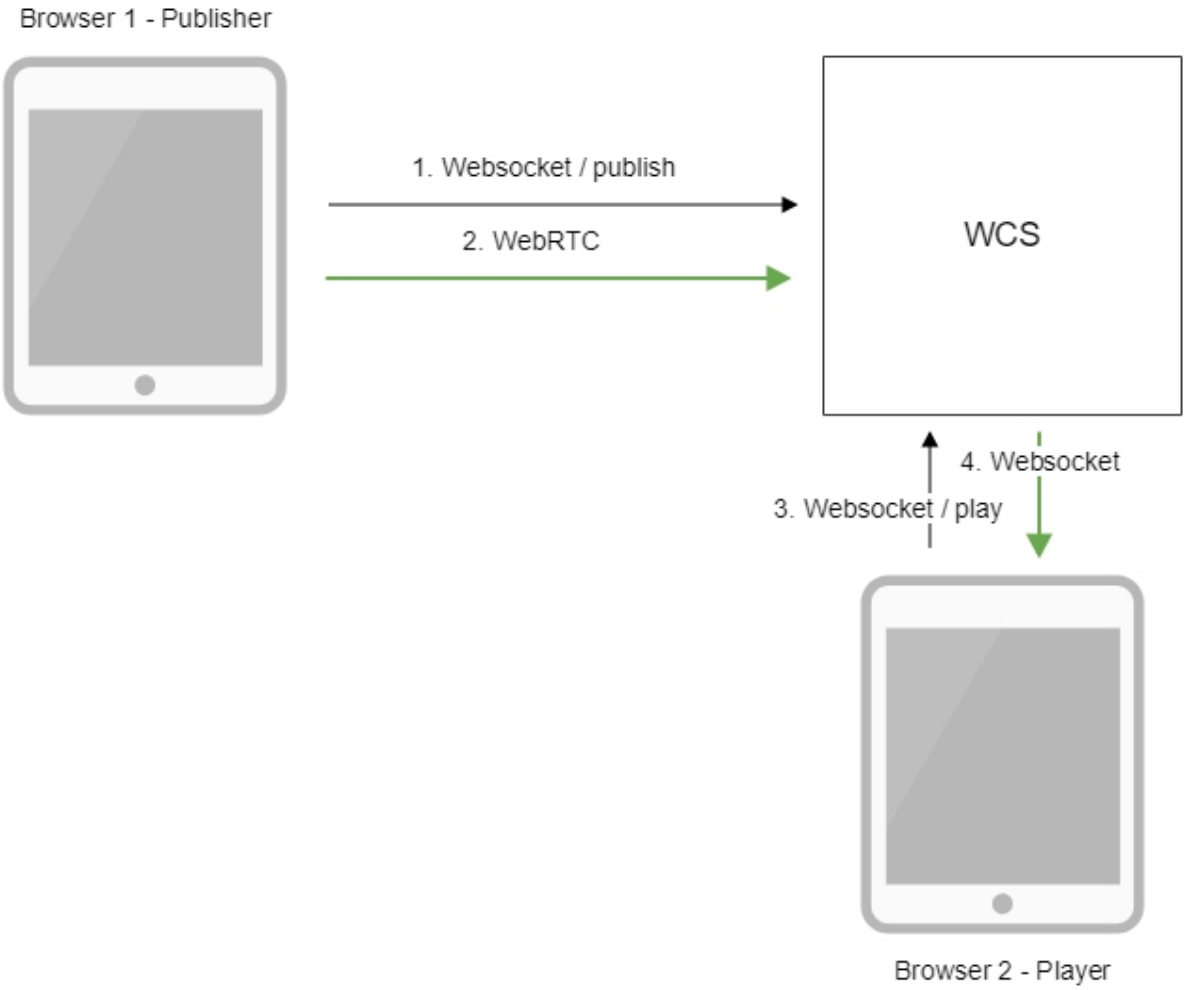
Supported platforms and browsers

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

Supported codecs

- Video: MPEG
- Audio: G.711

Operation flowchart



1. The browser connects to the server via the Websocket protocol and sends the publish command.
2. The browser captures the microphone and the camera and sends the WebRTC stream to the server.
3. The second browser establishes connection also via Websocket and sends the play command.
4. The second browser receives the MPEG + G.711 stream via Websocket and plays this stream on the page using HTML5 Canvas.

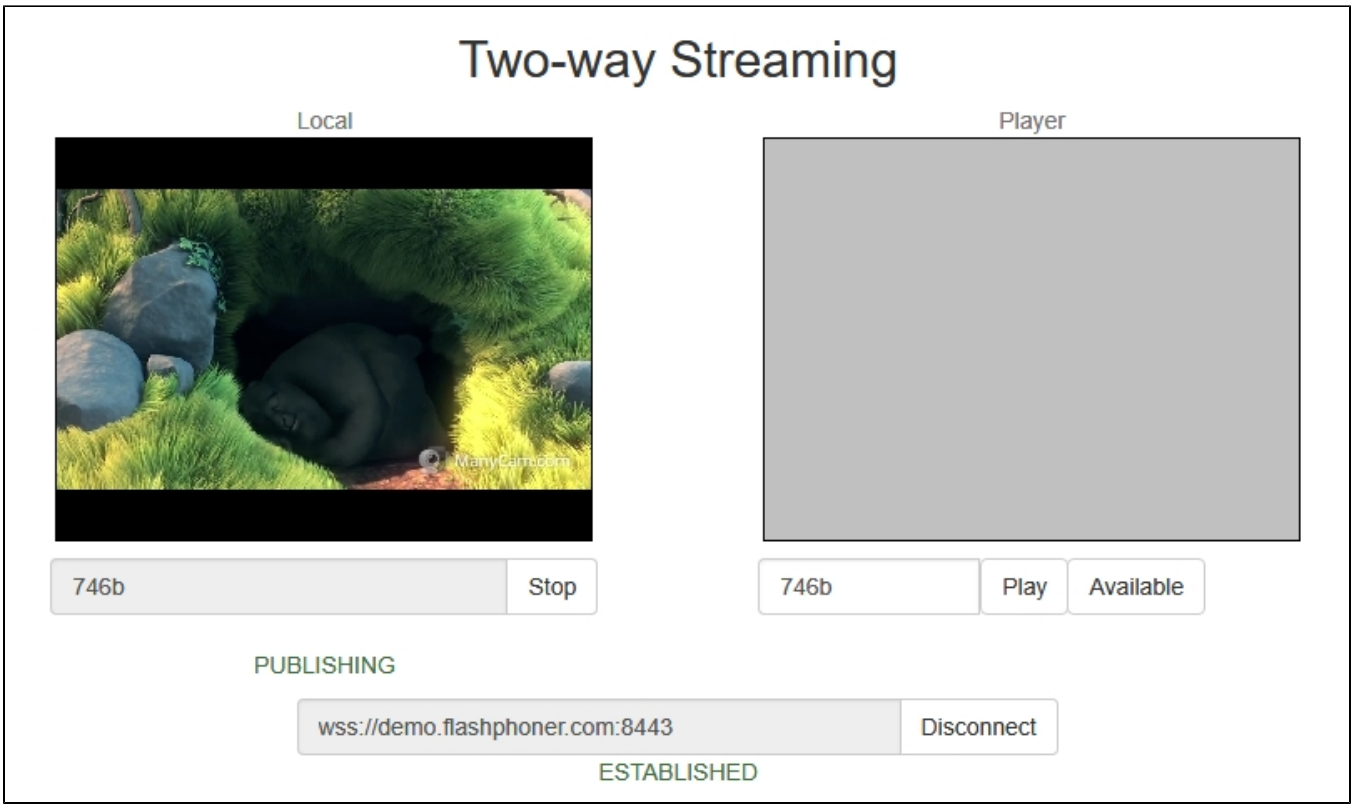
Quick manual on testing

Publishing a video stream on the server and playing it using HTML5 + Canvas (WSPlayer) in a browser

1. For this test we use:

- the demo server at demo.flashphoner.com;
- theTwo Way Streamingweb application to publish the stream
- thePlayerweb applicatiion to play the stream

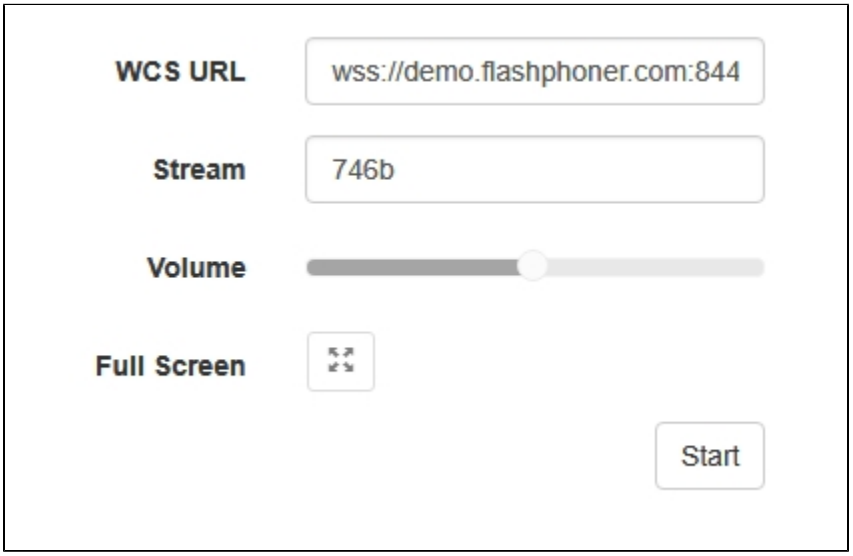
2. Open the Two Way Streaming web app. Click Connect, then click Publish. Copy the identifier of the stream:



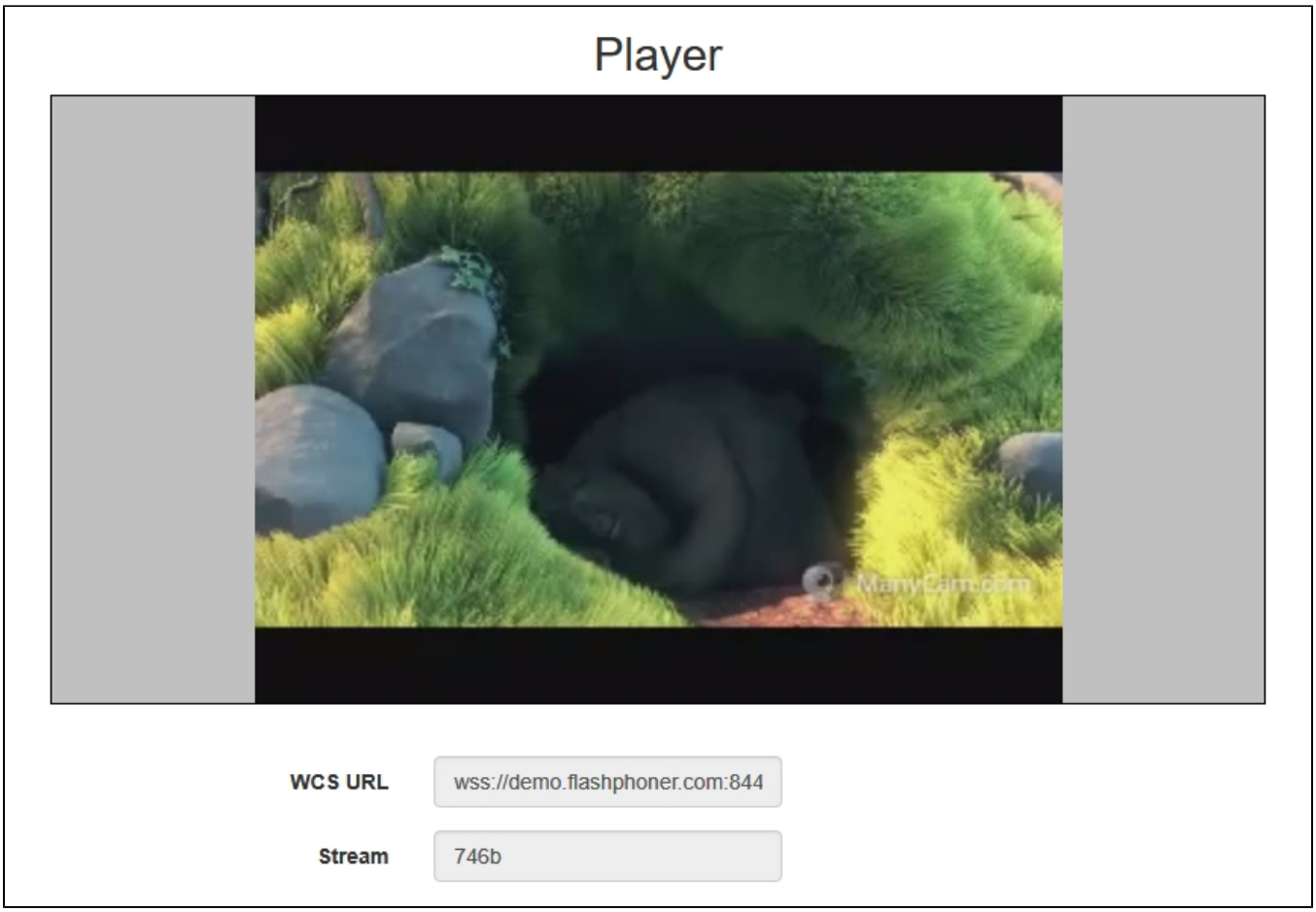
3. Open the Player web application and specify WSPlayer in the parameters of the URL

https://demo.flashphoner.com/client2/examples/demo/streaming/player/player.html?mediaProvider=WSPlayer

4. In the Stream field enter the identifier of the stream:



5. Click the Start button. The stream starts playing:

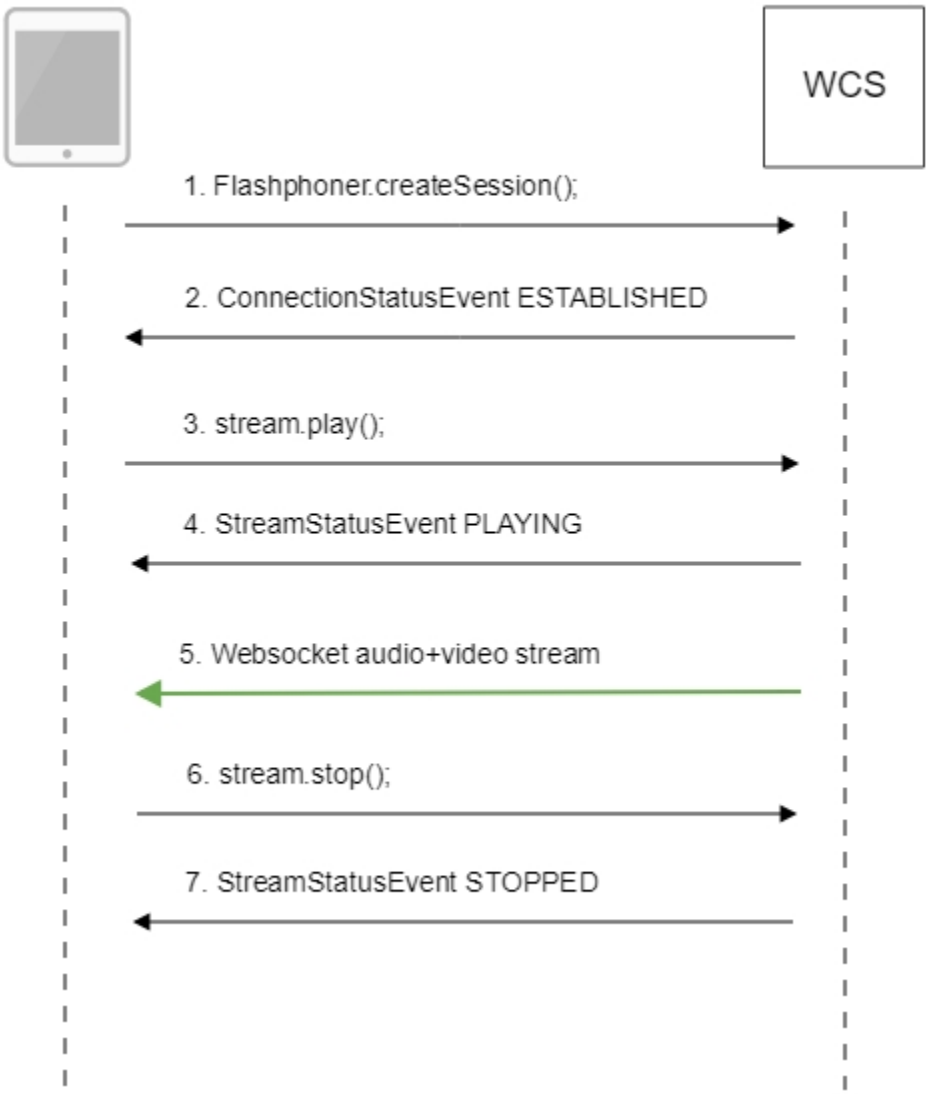


Call flow

Below is the call flow when using the Player example to play the stream using WSPlayer.

player.html

player.js



1. Establishing connection to the server.

init():code

```
    if (Flashphoner.getMediaProviders()[0] == "WSPlayer") {
        resolution_for_wsplayer = {playWidth:640,playHeight:480};
    }
```

Flashphoner.createSession();code

```
    Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
        setStatus(session.status());
        //session connected, start playback
        playStream(session);
    }).on(SESSION_STATUS.DISCONNECTED, function(){
        setStatus(SESSION_STATUS.DISCONNECTED);
        onStopped();
    }).on(SESSION_STATUS.FAILED, function(){
        setStatus(SESSION_STATUS.FAILED);
        onStopped();
    });
```

2. Receiving from the server an event confirming successful connection.

ConnectionStatusEvent ESTABLISHEDcode

```
    Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
        setStatus(session.status());
        //session connected, start playback
        playStream(session);
    }).on(SESSION_STATUS.DISCONNECTED, function(){
        ...
    }).on(SESSION_STATUS.FAILED, function(){
        ...
    });
```

3. Playing the stream.

stream.play();code

```
    if (Flashphoner.getMediaProviders()[0] === "MSE" && mseCutByIFrameOnly) {
        ...
    }
    if (resolution_for_wsplayer) {
        options.playWidth = resolution_for_wsplayer.playWidth;
        options.playHeight = resolution_for_wsplayer.playHeight;
    } else if (resolution) {
        ...
    }
    stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
        ...
    });
    stream.play();
```

4. Receiving from the server an event confirming successful playing of the stream.

StreamStatusEvent, status PLAYINGcode

```
    stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
        ...
    }).on(STREAM_STATUS.PLAYING, function(stream) {
        $("#preloader").show();
        setStatus(stream.status());
        onStarted(stream);
    }).on(STREAM_STATUS.STOPPED, function() {
        ...
    }).on(STREAM_STATUS.FAILED, function(stream) {
        ...
    }).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
        ...
    });
    stream.play();
```

5. Receiving the audio and video stream via Websocket and playing via MSE

6. Stopping the playback of the stream.

stream.stop();code

```
function onStarted(stream) {
    $("#playBtn").text("Stop").off('click').click(function(){
        $(this).prop('disabled', true);
        stream.stop();
    }).prop('disabled', false);
    ...
}
```

7. Receiving from the server an event confirming the playback of the stream is stopped.

StreamStatusEvent, status STOPPEDcode

```
    stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
        ...
    }).on(STREAM_STATUS.PLAYING, function(stream) {
        ...
    }).on(STREAM_STATUS.STOPPED, function() {
        setStatus(STREAM_STATUS.STOPPED);
        onStopped();
    }).on(STREAM_STATUS.FAILED, function(stream) {
        ...
    }).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
        ...
    });
    stream.play();
```

Known issues

1. Fullscreen mode is not supported for WSPlayer in iOS

Symptoms: Stream.fullScreen() call does not switch playback to full screen mode using WSPlayer mediaprovider

Solution: if possible, update device to latest iOS version and use WebRTC in Safari browser

2. WSPlayer does not support audio only stream playback

Symptoms: audio only stream does not play, there's no sound but stream is in PLAYING status

Solution:use aaudio+video streams, mute video if necessary (black screen will be displayed)

3. Two streams cannot be played simultaneously by WSPlayer using the same Websocket connection on the same page

Symptoms: two streams cannot be played in 2Players example using main browsers (Chrome, Firefox, Safari) while connecting to WCS server via HTTP

Решение: use a separate Websocket connection for each stream on the same page while playing them by WSPlayer