

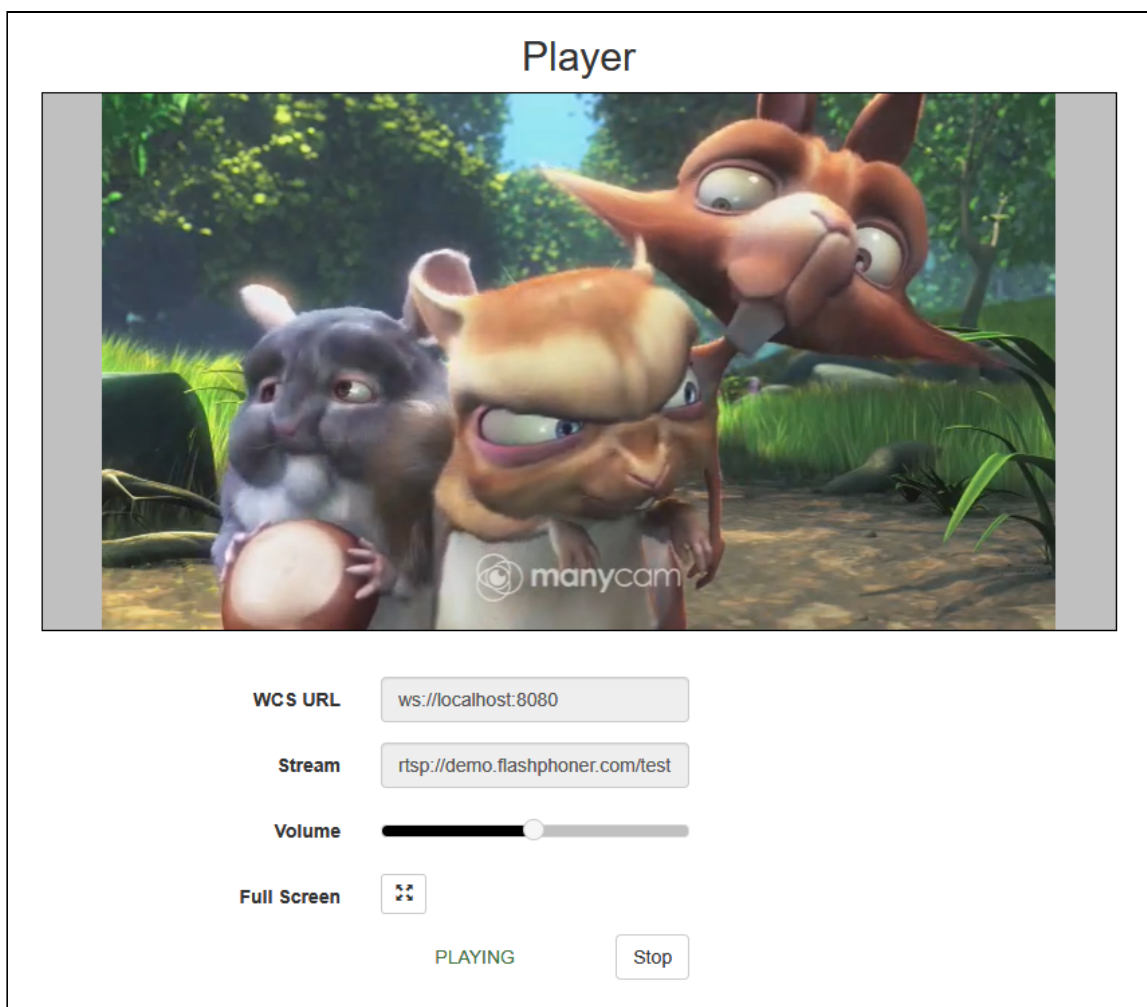
Player

Player example

This player can be used to play any type of stream on Web Call Server

- RTSP
- WebRTC
- RTMP

On the screenshot below an RTSP stream is being playing in the web player.



Code of the example

The path to the source code of the example on WCS server is:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/player

- `player.css` - file with styles
- `player.html` - page of the player
- `player.js` - script providing functionality for the player

This example can be tested using the following address:

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

Here host is the address of the WCS server.

Analyzing the code

To analyze the code, let's take the version of file `player.js` with hash `7fff01f`, which is available [here](#) and can be downloaded with corresponding build [2.0.219](#).

1. Initialization of the API

`Flashphoner.init()` [code](#)

```
Flashphoner.init({
  receiverLocation: '../dependencies/websocket-player/WSReceiver2.js',
  decoderLocation: '../dependencies/websocket-player/video-worker2.js',
  preferredMediaProvider: mediaProvider
});
```

2. Connection to server

`Flashphoner.createSession()` [code](#)

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED,
function(session){
  ...
}).on(SESSION_STATUS.DISCONNECTED, function(){
  ...
}).on(SESSION_STATUS.FAILED, function(){
  ...
});
```

3. Receiving the event confirming successful connection

`SESSION_STATUS.ESTABLISHED` [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(){
    ...
  }).on(SESSION_STATUS.FAILED, function(){
    ...
  });

```

4. Playback of video stream

`Session.createStream()`, `Stream.play()` [code](#)

```

stream = session.createStream(options).on(STREAM_STATUS.PENDING, function
(stream) {
    ...
});
stream.play();

```

When stream is created, the following parameters are passed

- `streamName` - name of the stream
- `remoteVideo` - `div` element, in which the video will be displayed
- picture resolution to transcode
- `unmutePlayOnStart: false` - disables automatic audio unmuting for autoplay to conform browsers requirements

[code](#)

```

var options = {
  name: streamName,
  display: remoteVideo,
  flashShowFullScreenButton: true
};
if (Flashphoner.getMediaProviders()[0] === "MSE" && mseCutByIFrameOnly) {
  options.mediaConnectionConstraints = {
    cutByIFrameOnly: mseCutByIFrameOnly
  }
}
if (resolution_for_wsplayer) {
  options.playWidth = resolution_for_wsplayer.playWidth;
  options.playHeight = resolution_for_wsplayer.playHeight;
} else if (resolution) {
  options.playWidth = resolution.split("x")[0];
  options.playHeight = resolution.split("x")[1];
}
if (autoplay) {
  options.unmutePlayOnStart = false;
}

```

5. Receiving the event confirming successful stream playback

`STREAM_STATUS.PLAYING` [code](#)

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

6. Handling the event about channel bandwidth

`STREAM_EVENT`, `STREAM_EVENT_TYPE.NOT_ENOUGH_BANDWIDTH` [code](#)

```
}).on(STREAM_EVENT, function(streamEvent){
    if (STREAM_EVENT_TYPE.NOT_ENOUGH_BANDWIDTH === streamEvent.type) {
        var info = streamEvent.payload.info.split("/");
        var remoteBitrate = info[0];
        var networkBandwidth = info[1];
        console.log("Not enough bandwidth, consider using lower video
resolution or bitrate. Bandwidth " + (Math.round(networkBandwidth / 1000)) +
" bitrate " + (Math.round(remoteBitrate / 1000)));
    } else if (STREAM_EVENT_TYPE.RESIZE === streamEvent.type) {
        ...
    }
});
```

7. Handling the event about changing stream picture size

`STREAM_EVENT`, `STREAM_EVENT_TYPE.RESIZE` [code](#)

```
}).on(STREAM_EVENT, function(streamEvent){
    ...
    } else if (STREAM_EVENT_TYPE.RESIZE === streamEvent.type) {
        console.log("New video size: " +
streamEvent.payload.streamerVideoWidth + "x" +
streamEvent.payload.streamerVideoHeight);
    }
});
```

8. Playback stop

`Stream.stop()` [code](#)

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

9. Receiving the event confirming successful stream playback stop

`STREAM_STATUS.STOPPED` [code](#)

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING, function
(stream) {
    ...
}).on(STREAM_STATUS.STOPPED, function () {
    $("#preloader").hide();
    setStatus(STREAM_STATUS.STOPPED);
    onStopped();
}).on(STREAM_STATUS.FAILED, function() {
    ...
}).play();
```

10. Playback volume control

`Stream.unmuteRemoteAudio()`, `Stream.setVolume()` [code](#)

```
$("#volumeControl").slider({
    range: "min",
    min: 0,
    max: 100,
    value: currentVolumeValue,
    step: 10,
    animate: true,
    slide: function(event, ui) {
        //WCS-2375. fixed autoplay in ios safari
        if (stream.isRemoteAudioMuted()) {
            stream.unmuteRemoteAudio();
        }
        currentVolumeValue = ui.value;
        stream.setVolume(currentVolumeValue);
    }
}).slider("disable");
```

11. Playback automatic start on page load

[code](#)

```
if (autoplay) {  
    // We can start autoplay with muted audio only, so set volume slider to 0  
    #WCS-2425  
    $("#volumeControl").slider('value', 0);  
    $("#playBtn").click();  
}
```