

# Publisher and player channel quality control

- [Server configuration](#)
- [Channel quality displaying on client side](#)
- [Testing](#)
- [Recommendations to publishers](#)
- [Recommendations to subscribers](#)

When a WebRTC stream is publishing, picture quality depends on media data transfer channel between client and server, especillay for high definition streams (HD, FullHD, 4K). The ability to control channel quality and notify publisher about bandwidth decrease in time using WebSDK was added since build [5.2.398](#). Subscriber, in its turn, can be notified about bandwidth decrease since build [5.2.409](#).

The publishing and playback bitrate values on client side are periodically comparing with server side one. The steady divergence of those values means channel bandwidth decrease. The peaks and sudden changes are smoothed by [Kalman filter](#).

## Server configuration

Current server side publishing bitrate values sending to client for later comparison is enabled with the following parameter in [flashphoner.properties](#) file

```
inbound_video_rate_stat_send_interval=1
```

Current server side playback bitrate values sending is enabled with the following parameter

```
outbound_video_rate_stat_send_interval=1
```

The settings above define bitrate values sending interval in seconds. It is recommended to send bitrate to client every second.

## Channel quality displaying on client side

Let's look at channel quality and bitrate changing graphs displaying using [Media Devices](#) example.

1. A function to prepare to display graphs [code](#)

```
function createOrClearChart(chartId, bitrateComparisonChart) {
    if (!bitrateComparisonChart) {
        var canvas = document.getElementById(chartId);
        var ctx = canvas.getContext('2d');
        bitrateComparisonChart = new ComparisonChart(ctx);
    } else {
        bitrateComparisonChart.clearBitrateChart();
    }
    return bitrateComparisonChart;
}
```

function usage while publishing [code](#)

```
function publish() {
    ...
    publishConnectionQualityStat.chart = createOrClearChart('publishBitrateChart', publishConnectionQualityStat.chart);

    publishStream = session.createStream({
        ...
    });
    publishStream.publish();
}
```

function usage while playing [code](#)

```
function play() {
    ...
    playConnectionQualityStat.chart = createOrClearChart('playBitrateChart', playConnectionQualityStat.chart);

    previewStream = session.createStream({
        ...
    });
    previewStream.play();
}
```

## 2. Channel quality and bitrate values receiving, bitrate graphs displaying

CONNECTION\_QUALITY.UPDATE

event handling while publishing [code](#)

```
publishStream = session.createStream({
    ...
}).on(CONNECTION_QUALITY.UPDATE, function (quality, clientFiltered, serverFiltered) {
    updateChart(quality, clientFiltered, serverFiltered, publishConnectionQualityStat);
});
publishStream.publish();
```

while playing [code](#)

```
previewStream = session.createStream({
    ...
}).on(CONNECTION_QUALITY.UPDATE, function (quality, clientFiltered, serverFiltered) {
    updateChart(quality, clientFiltered, serverFiltered, playConnectionQualityStat);
});
previewStream.play();
```

a function to update graphs and quality [code](#)

```
function updateChart(calculatedQuality, clientFiltered, serverFiltered, connectionQualityStat) {
    var timestamp = new Date().valueOf();
    connectionQualityStat.connectionQualityUpdateTimestamp = timestamp;
    connectionQualityStat.chart.updateChart(clientFiltered, serverFiltered);
    connectionQualityStat.quality = calculatedQuality;
}
```

## 4. Set channel quality to UNKNOWN, if CONNECTION\_QUALITY.UPDATE event is not received

while publishing [code](#)

```
function loadStats() {
    if (publishStream) {
        ...
        if(new Date().valueOf() - CONNECTION_QUALITY_UPDATE_TIMEOUT_MS >
publishConnectionQualityStat.connectionQualityUpdateTimestamp) {
            publishConnectionQualityStat.quality = CONNECTION_QUALITY.UNKNOWN;
        }
        ...
    }
}
```

while playing [code](#)

```
function loadStats() {
    ...
    if (previewStream) {
        ...
        if(new Date().valueOf() - CONNECTION_QUALITY_UPDATE_TIMEOUT_MS >
publishConnectionQualityStat.connectionQualityUpdateTimestamp) {
            publishConnectionQualityStat.quality = CONNECTION_QUALITY.UNKNOWN;
        }
        ...
    }
}
```

## 5. Channel quality displaying

while publishing [code](#)

```
function loadStats() {
    if (publishStream) {
        ...
        if (publishConnectionQualityStat.quality !== undefined) {
            showStat({"quality": publishConnectionQualityStat.quality}, "outConnectionStat");
        }
        ...
    }
}
```

while playing [code](#)

```
function loadStats() {
    if (publishStream) {
        ...
        if (playConnectionQualityStat.quality !== undefined) {
            showStat({"quality": playConnectionQualityStat.quality}, "inConnectionStat");
        }
        ...
    }
}
```

a function to display quality [code](#)

```
function showStat(stat, type) {
    Object.keys(stat).forEach(function(key) {
        if (typeof stat[key] !== 'object') {
            let k = key.split(/(?=[A-Z])/);
            let metric = "";
            for (let i = 0; i < k.length; i++) {
                metric += k[i][0].toUpperCase() + k[i].substring(1) + " ";
            }
            if ($("#" + key + "-" + type).length == 0) {
                let html = "<div style='font-weight: bold'>" + metric.trim() + ": <span id='" + key + "-" +
+ type + "' style='font-weight: normal'></span>" + "</div>";
                // $(html).insertAfter("#" + type);
                $("#" + type).append(html);
            } else {
                $("#" + key + "-" + type).text(stat[key]);
            }
        }
    });
}
```

# Testing

1. For the test we use:

- [WCS5.2.409](#) or newer
- Media Devices example in Chrome browser
- publishing channel with 100 Mbps upload and download bandwidth
- bandwidth shaping tool, [winShaper](#) on Windows or [Network Link Conditioner](#) on MacOS for example

2. Publish and play 720p stream on Media Devices page

## Media Devices

### Video stats

Codec: H264  
Codec Rate: 90000  
Fir Count: 0  
Pli Count: 2  
Nack Count: 0  
Packets Sent: 22877  
Bytes Sent: 22689293  
Height: 720  
Width: 1280  
Bitrate: 5407720

### Audio stats

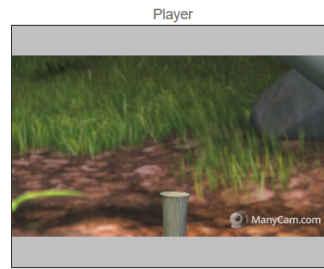
Codec: opus  
Codec Rate: 48000  
Packets Sent: 2111  
Bytes Sent: 196162  
Bitrate: 37808

### Connection

Quality: PERFECT



PUBLISHING



PLAYING

### Video stats

Codec: H264  
Codec Rate: 90000  
Fir Count: 0  
Pli Count: 2  
Nack Count: 0  
Packets Received: 11748  
Bytes Received: 12118656  
Packets Lost: 0  
Height: 720  
Width: 1280  
Bitrate: 5521368

### Audio stats

Codec: opus  
Codec Rate: 48000  
Packets Received: 1110  
Bytes Received: 102910  
Packets Lost: 0  
Bitrate: 37712

### Connection

Quality: PERFECT

wss://centos1.flashphoner.com:8443

Disconnect

Timeout

1000

msec

ESTABLISHED

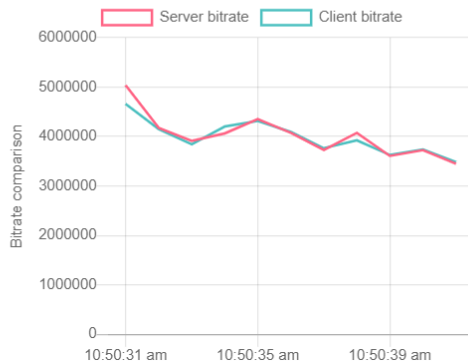
The PERFECT channel quality is displayed for publisher and player

3. Check publishing and playing bitrate graphs on perfect channel

☒ Send Audio

Publish bitrate chart

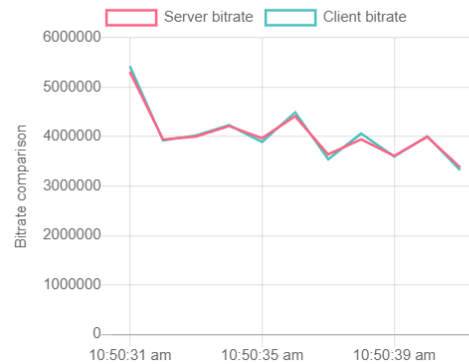
show



☒ Play audio

Play bitrate chart

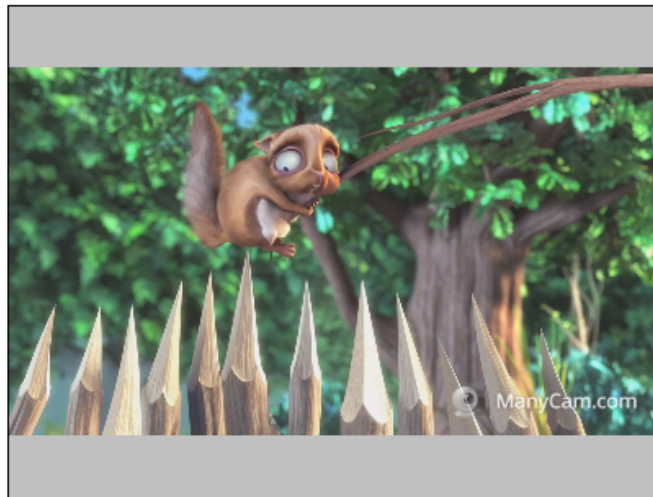
show



4. Shape outgoing traffic to 768 kbps, simulating a typical 3G connection

**Video stats****Codec:** H264**Codec Rate:** 90000**Fir Count:** 0**Pli Count:** 8315**Nack Count:** 1562**Packets Sent:** 410527**Bytes Sent:** 408125317**Height:** 720**Width:** 1280**Bitrate:** 4171488**Audio stats****Codec:** opus**Codec Rate:** 48000**Packets Sent:** 40545**Bytes Sent:** 3618193**Bitrate:** 37200**Connection****Quality:** BAD

Local



1280x720

test

Stop

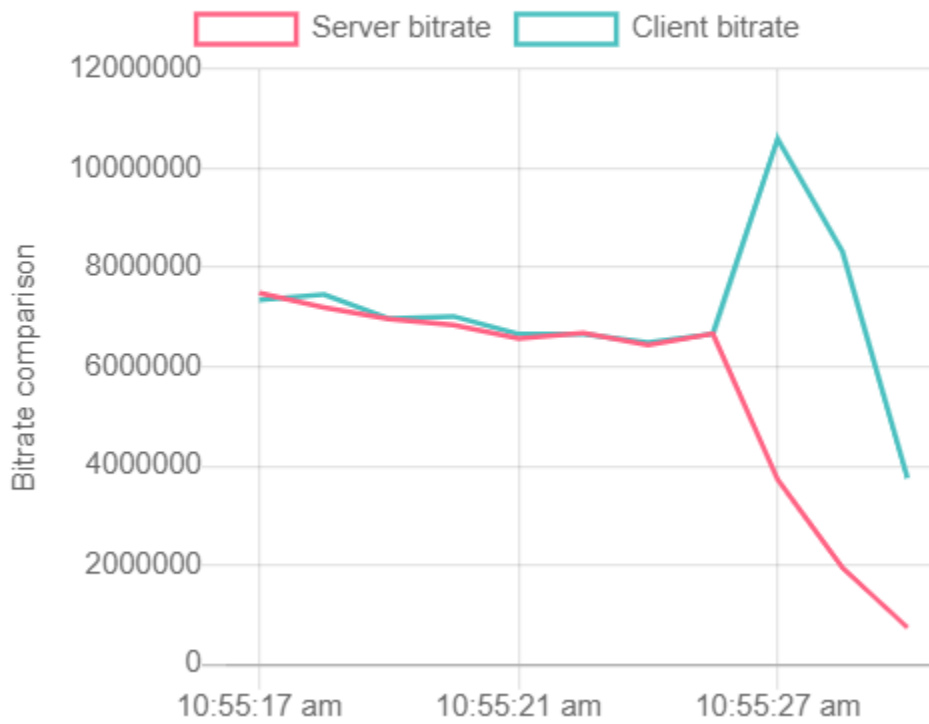
PUBLISHING

The PERFECT value changes to BAD for publisher

Publishing bitrate graph looks as follows

## Publish bitrate chart

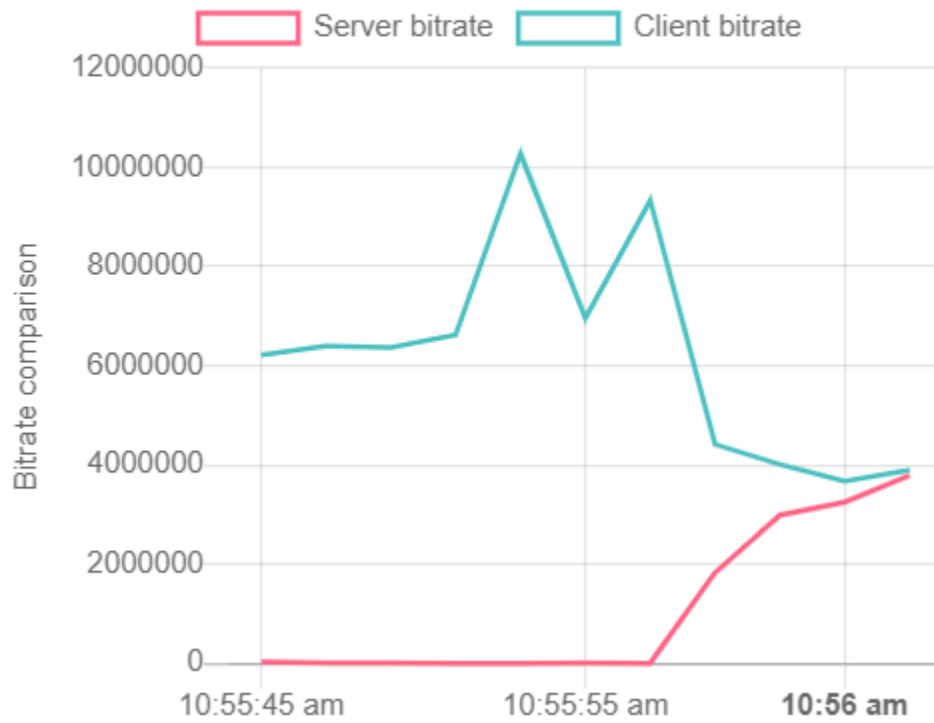
show



5. Stop bandwidth shaping, check publishing bitrate graphs

## Publish bitrate chart

show



After the graphs converge again, the PERFECT publisher channel quality value is displayed.

6. Shape incoming traffic to 768 kbps

### Player



1280x720

test

Stop

PLAYING

### Video stats

**Codec:** H264

**Codec Rate:** 90000

**Fir Count:** 0

**Pli Count:** 81

**Nack Count:** 0

**Packets Received:** 266189

**Bytes Received:** 271314911

**Packets Lost:** 0

**Height:** 720

**Width:** 1280

**Bitrate:** 145864

### Audio stats

**Codec:** opus

**Codec Rate:** 48000

**Packets Received:** 33312

**Bytes Received:** 2967388

**Packets Lost:** 0

**Bitrate:** 31032

### Connection

**Quality:** BAD

The PERFECT value changes to BAD for subscriber, picture freeze and artefacts are observed

Playing bitrate graph looks as follows

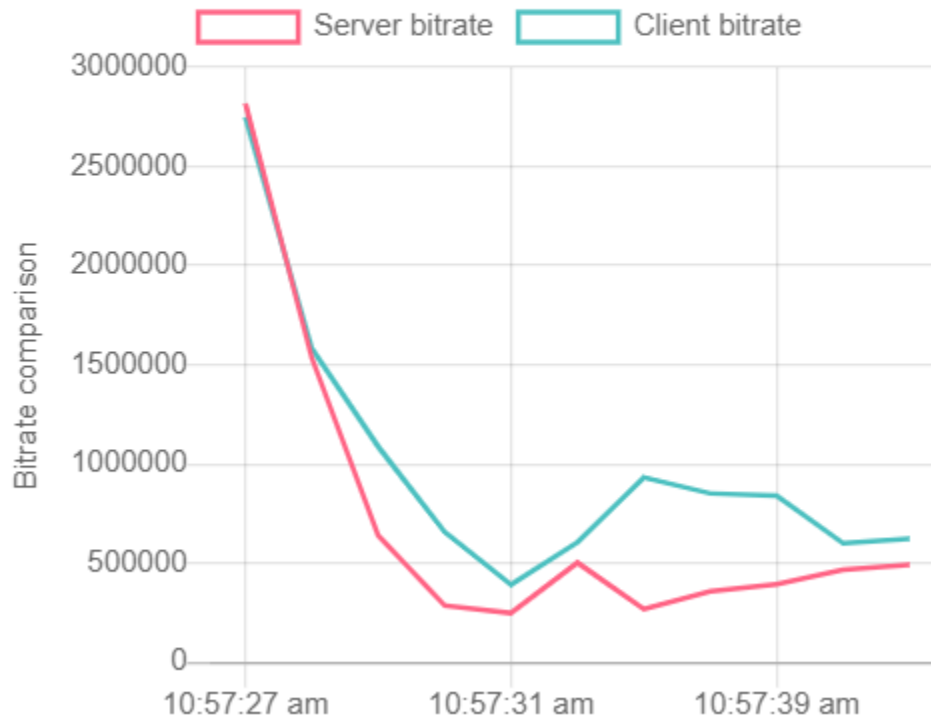




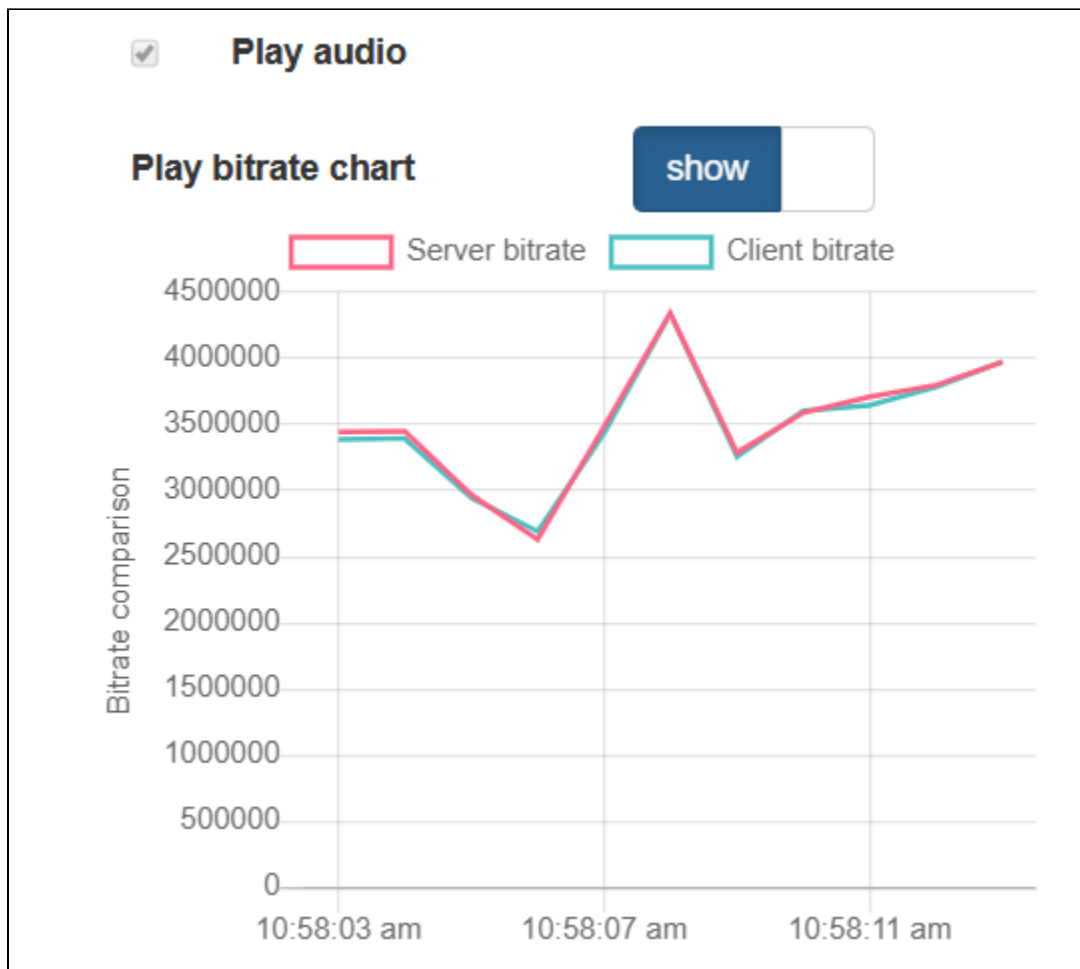
**Play audio**

**Play bitrate chart**

show



7. Stop bandwidth shaping, check playing bitrate graphs



After the graphs converge again, the PERFECT subscriber channel quality value is displayed, picture is restored

## Recommendations to publishers

If channel quality is displayed as PERFECT or GOOD, it means channel bandwidth is enough to publish a stream with a current bitrate

If channel quality is changed steadily to BAD, it means channel bandwidth is not enough, and subscribers are viewing a problems. It is recommended to lower publishing bitrate and/or resolution if possible.

If channel quality is changed steadily to UNKNOWN, video frames can not reach the server. It is recommended to republish stream.

## Recommendations to subscribers

If channel quality is displayed as PERFECT or GOOD, it means channel bandwidth is enough to play a stream with a current bitrate. If the problems occur while playing stream in this case, the source of the problems is probably on publisher side.

If channel quality is changed steadily to BAD, it means channel bandwidth is not enough, picture freeze and artefacts are observed. It is recommended to request the stream with lower bitrate and/or resolution if possible.

If channel quality is changed steadily to UNKNOWN, video frames can not be received from the server. It is recommended to reconnect and restart the stream playback.