From an IP camera via RTSP

Overview

A video stream is captured from an RTSP source that provides audio and video in the supported codecs. Then, the server transforms this video stream for playing in browsers and mobile devices.

RTSP sources

- IP cameras
- Media servers
- Surveillance systems
- Conference servers

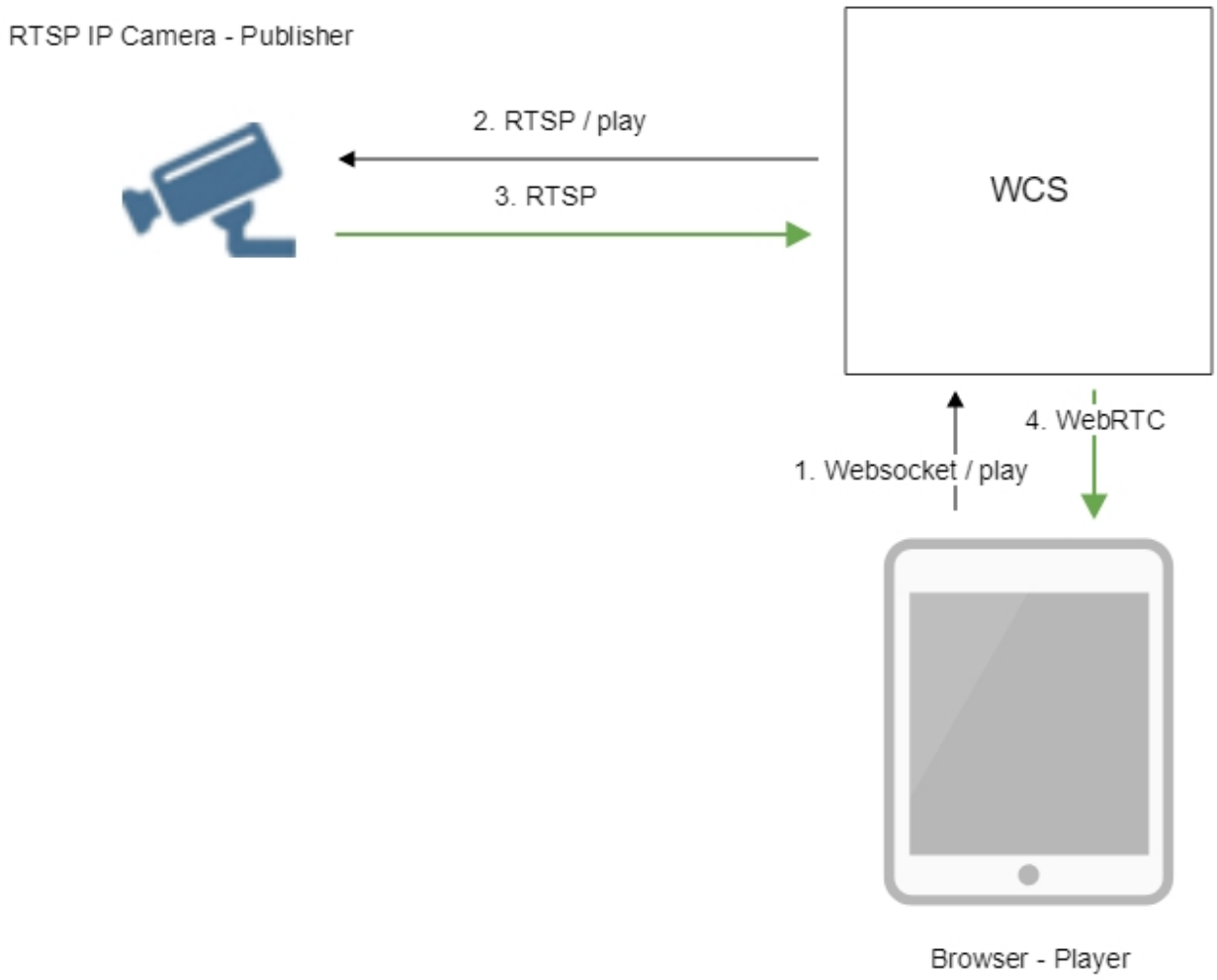
Supported codecs

- H.264
- H265 (since5.2.1579)
- VP8
- AAC
- G.711
- Speex

Supported platforms and browsers

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

Operation flowchart



1. The browser establishes a connection to the server via the Websocket protocol and sends the play command.

2. The server connects to the RTSP source and send the play command.

3. The RTSP source sends the RTSP stream to the server.

4. The server transforms the stream to WebRTC and gives the stream to the browser.

Configuration

RTSP client binding to certain address

Sometimes, when IP camera should be connected through VPN, RTSP client should be bound to certain IP address. The optionrtsp_client_address in settings fileflashphoner.propertiesdefines this address, for example:

```
rtsp_client_address=172.16.0.3
```

RTSP stream capturing via UDP

By default, RTSP streams are captured via TCP. Stream capturing over UDP can be turned on if necessary with the following parameter

```
rtsp_interleaved_mode=false
```

Audio and video track selection in RTSP stream

By default, audio and video tracks in RTSP tream are selected dynamically according to camera SDP. Tracks order can be set explicitly if necessary with the following parameter, for example

```
rtsp_interleaved_channels=2-3;0-1
```

Where

- 2-3 - audio track channels
- 0-1 - video track channels

Playing RTSP stream in AnnexB form

Some IP cameras are publishing H264 stream in AnnexB format, Honeywell MAXPRO Video Streamer for example. To play video from such cameras, the following parameter is added since build5.2.636to enable AnnexB format parsing

```
h264_check_and_skip_annexb=true
```

Since build5.2.946,this parameter is removed from settings, and AnnexB frames are detected and played automatically.

Audio codecs exclusion

Sometimes it is necessary to capture a stream from camera without audio, or disable some audio codecs to escapr sound transcoding. This can be done with the following parameter listing audio codec names to be excluded while capturing stream, for example

```
rtsp_client_strip_audio_codecs=PCMA,PCMU
```

This setting will exclude PCMA (alaw) and PCMU (ulaw) codecs. A stream with audio track in those codecs will be captured from a camera as video only.

The listed codecs are excluded at SDP level, by names.

H264 packetization mode configuration

According to H264 specifications, if H264 packetization mode is not set explicitly in stream SDP, if should be set to 0. However, some RTSP cameras may send a stream encided with packetization mode 1, and may omit this mode setting in SDP. This behaviour may lead to stream transcoding and picture quality dropping while playing the stream in Safari browser.

Since build5.2.820it is possible to set default packetization mode to use such cameras with the following parameter

```
default_packetization_mode=1
```

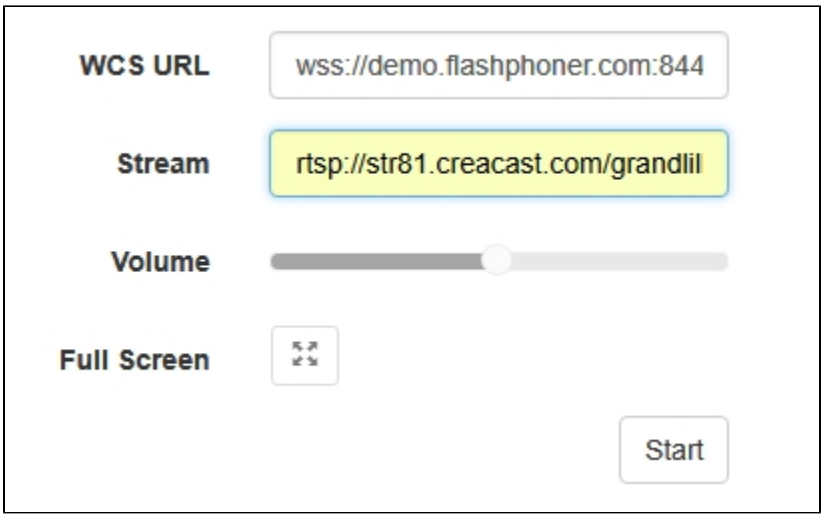
Quick manual on testing

Capturing of a video stream from the IP camera and playing it in a browser

1. For this test we use:

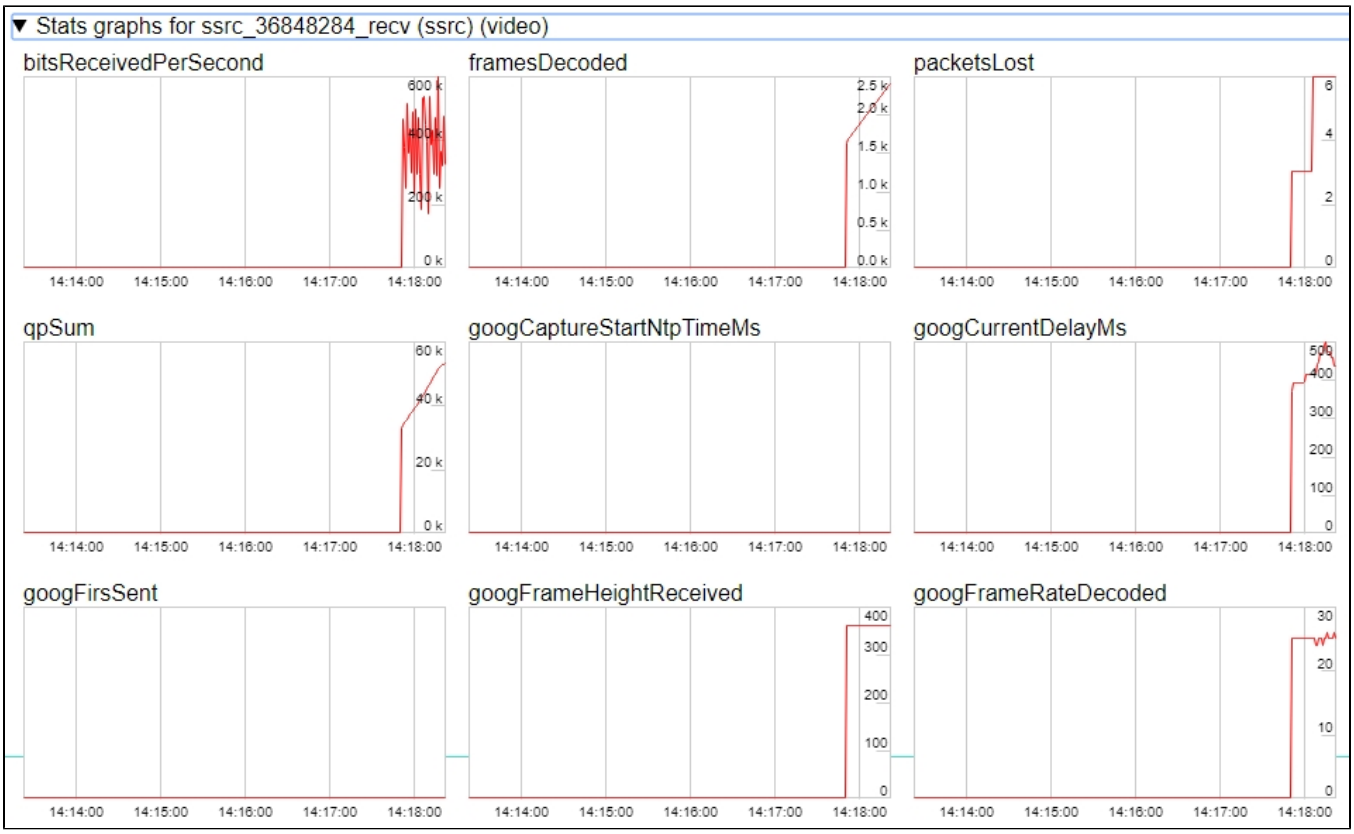
- the demo server atdemo.flashphoner.com;
- thePlayerweb application to play the captured stream in the browser.

2. Open the Player web app and specify the URL of the camera in the "Stream" field:



3. Click the "Start" button. Broadcasting of the captured stream begins.

4. WebRTC internals diagrams:



Stream capture from the IP camera management by REST API

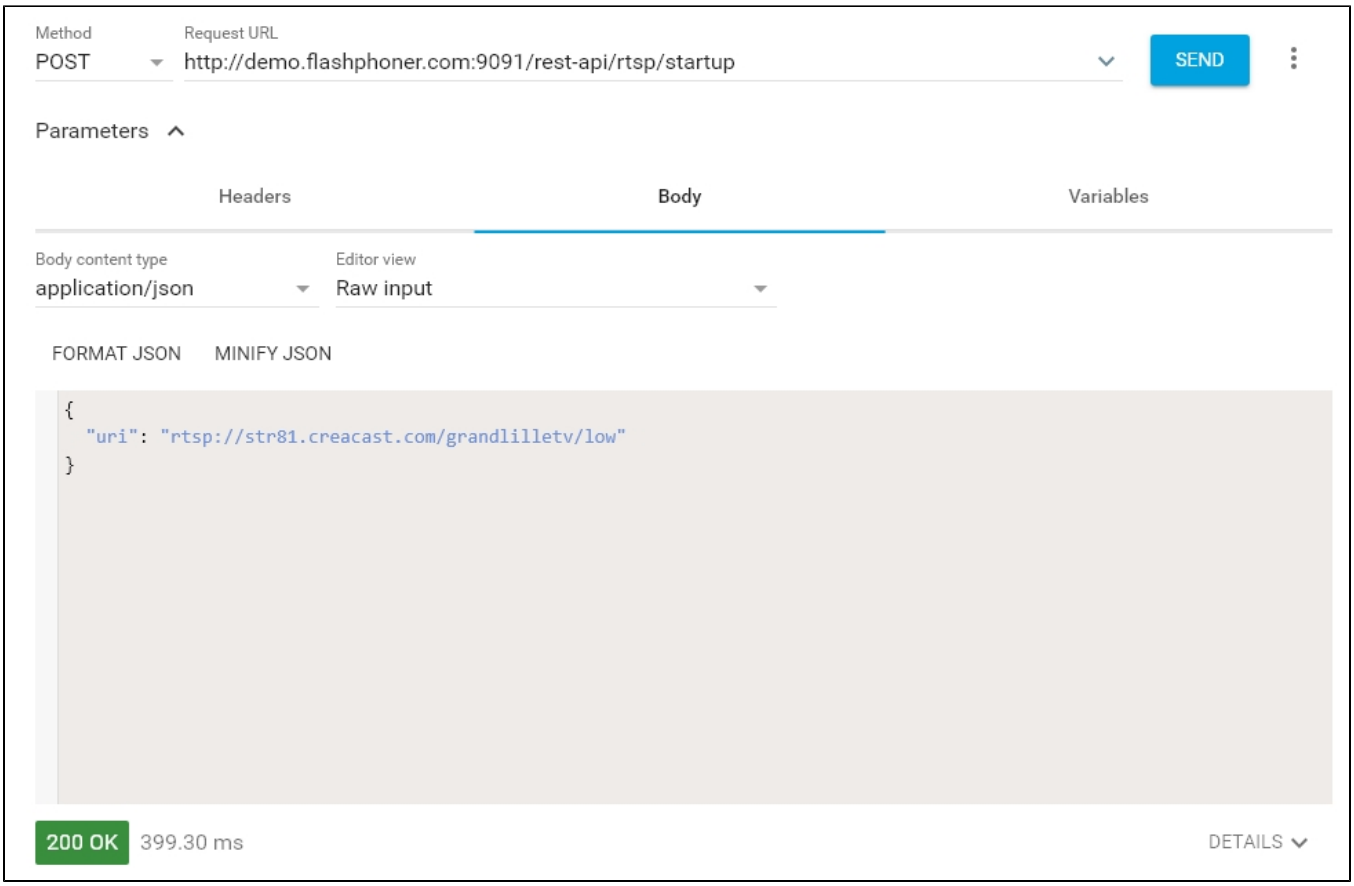
Usually, it is enough to set the camera URL as stream name to capture stream from IP camera. However, it is possible to manage RTSP stream capture by REST API if necessary.

Testing

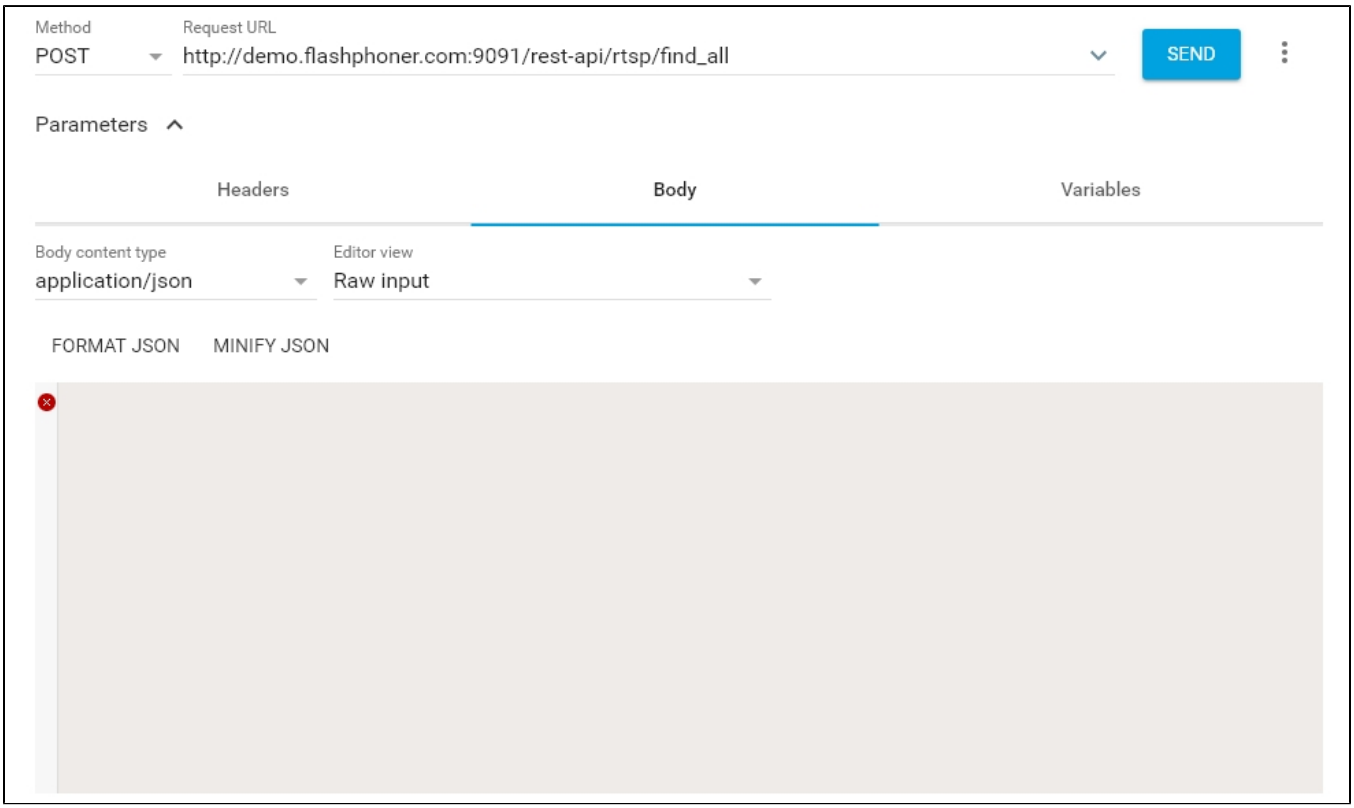
1. For this test we use:

- the demo server atdemo.flashphoner.com;
- the Chrome browser and theREST-clientto send queries to the server;
- thePlayerweb application to play the captured stream in the browser.

2. Open the REST client. Send the /rtsp/startup query specifying the URL of the web camera in parameters:

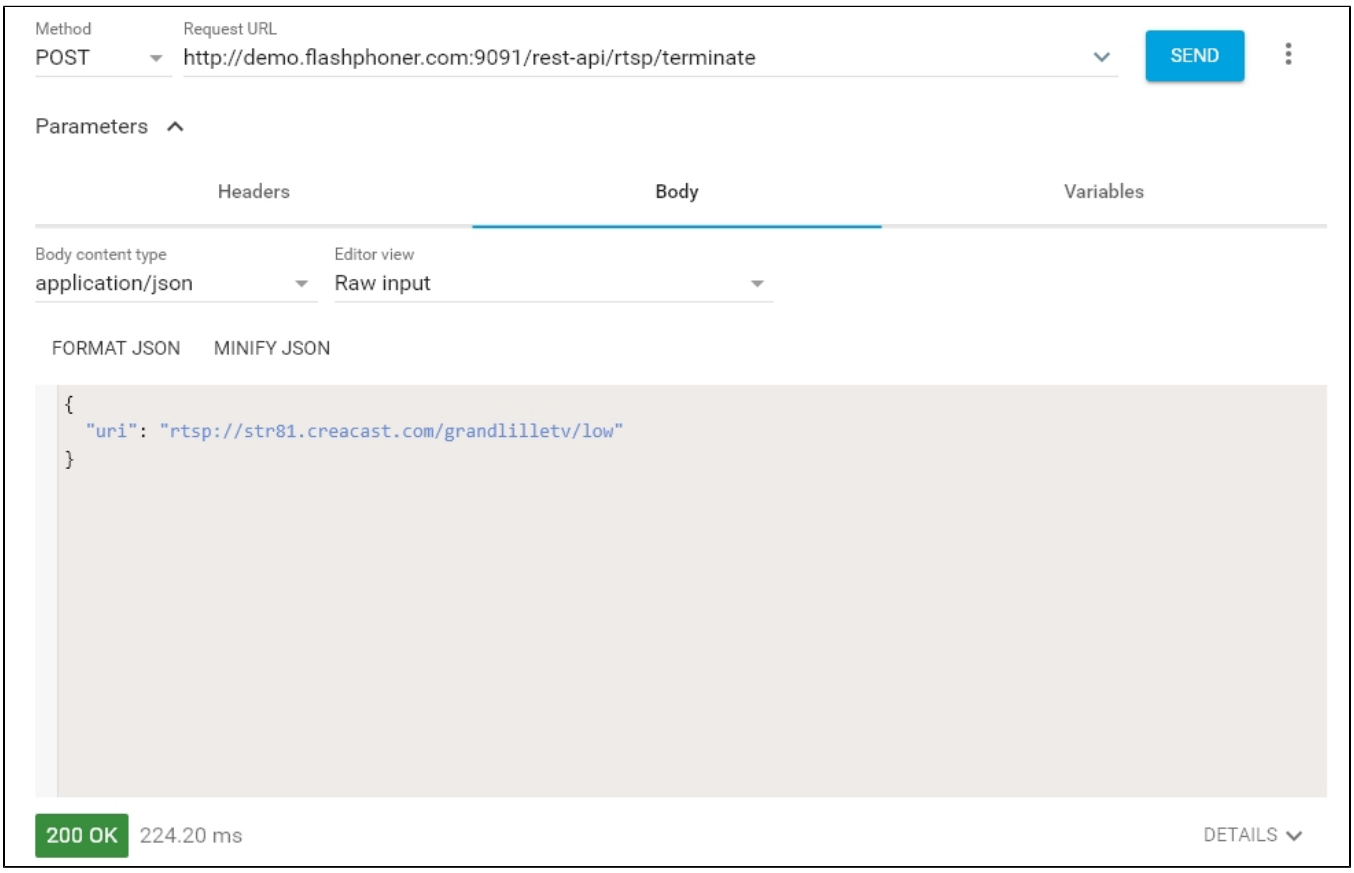


3. Make sure the stream is captured by the server. To do this, send the /rtsp/find_all query:

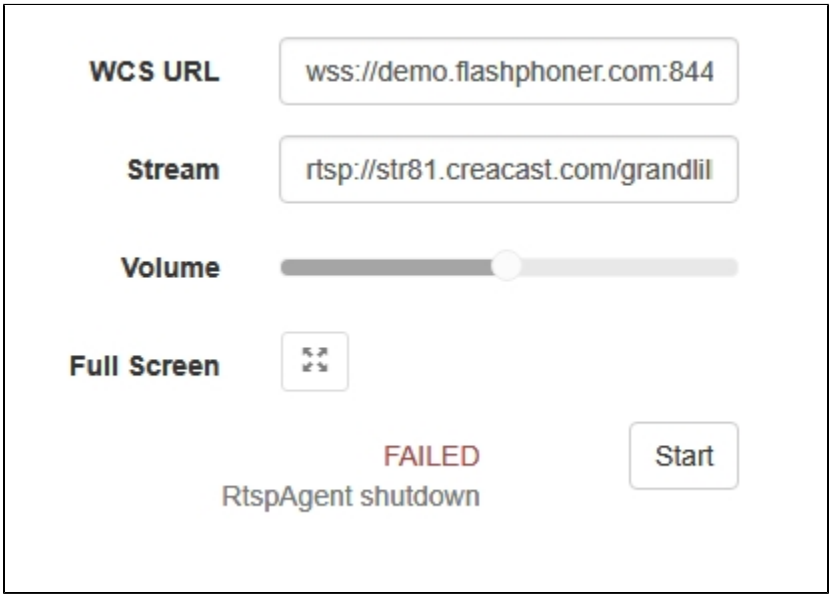


4. Open the Player web app and in the "Stream" field specify the URL of the web camera and click Start. Browser starts to play the stream:

5. Send the /rtsp/terminate query specifying the URL of the web camera in parameters:



6. Stream playback will terminate displaying an error:



REST-queries

A REST-query should be HTTP/HTTPS POST request as follows:

- HTTP:http://test.flashphoner.com:8081/rest-api/rtsp/startup

- HTTPS:https://test.flashphoner.com:8444/rest-api/rtsp/startup

Where:

- test.flashphoner.com - is the address of the WCS server
- 8081 - is the standard REST / HTTP port of the WCS server
- 8444 - is the standard HTTPS port
- rest-api - is the required part of the URL
- /rtsp/startup - REST-method to use

REST-methods and response statuses

REST-method	Example of REST-query	Example of response	Response statuses	Description

/rtsp /startup	<pre>{
	"uri":"rtsp://myserver.com /live/myStream",
	"localStreamName": "myRTSPstream"
	}</pre>
			409 - Conflict
			
			500 - Internal error
				Pull the RTSP stream by the specified URL

/rtsp /find_all		<pre>{
		"uri": "rtsp://myserver.com /live/myStream",
		"status": "PLAYING",
		"localStreamName": "myRTSPstream"
		}</pre>
			200 – streams found
			404 – streams not found
				Find all pulled RTSP-streams

/rtsp /terminate	<pre>{
	"uri":"rtsp://myserver.com /live/myStream"
	}</pre>
			200 - stream terminated
			404 - stream not found
				Terminate the pulled RTSP stream

Parameters

Parameter name	Description	Example
uri	URL of the RTSP stream	rtsp://myserver.com/live/myStream
localStreamName	Name to set to the stream captured	myRTSPstream
status	Current status of the stream	PLAYING

RTSP stream repeatedly capturing with the same URI

/rtsp/startup query returns 409 Conflict while trying to repeatedly capture RTSP stream with the same URI.If the stream from the IP camera is already published on the server, it is necessary to subscribe to it.

Call flow

Below is the call flow when using the Player example

player.html

player.js

1. Establishing a connection to the server.

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. Request to play the stream.

session.createStream(), stream.play();code

IP camera URL is passed to createStream() method as stream name

```
function playStream(session) {
    var streamName = $('#streamName').val();
    var options = {
        name: streamName,
        display: remoteVideo,
        flashShowFullScreenButton: true
    };
    ...
    stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
        ...
    });
    stream.play();
}
```

4. Request from WCS to the RTSP source to broadcast the stream.

5. Broadcasting the RTSP stream

6. Receiving from the server an event confirming successful capturing and playing of the stream.

StreamStatusEvent, статус PLAYINGcode

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

7. Sending audio- and video stream via WebRTC

8. Stopping playing the stream.

stream.stop();code

```
function onStarted(stream) {
    $("#playBtn").text("Stop").off('click').click(function(){
        $(this).prop('disabled', true);
        stream.stop();
    }).prop('disabled', false);
    $("#fullScreenBtn").off('click').click(function(){
       stream.fullScreen();
    }).prop('disabled', false);
    $("#volumeControl").slider("enable");
    stream.setVolume(currentVolumeValue);
}
```

9. Receiving from the server an event confirming successful stop of the stream playback.

StreamStatusEvent, статус 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();
```

RTSP connection reuse

If other subscribers request the stream captured from RTSP IP camera, the previous RTSP connection will be used if all subscribers set the same camera URL. For example, two requests to the same IP camera

```
rtsp://host:554/live.sdp
```

and

```
rtsp://host:554/live.sdp?p=1
```

are differ, then two RTSP connections will be created if streams from both URLs are requested.

Stream capture authentication

WCS supports RTSP stream capture authentication by user name and password, user data should be set in stream URL, for example

```
rtsp://user:password@hostname/stream
```

If name or password contains any special characters, they should be escaped such as

```
rtsp://user:p%40ssword@hostname/stream
```

Where

- user is user name

- p@ssword is password with character '@', it is escaped in URL.

Another IP address redirection handling

Some IP cameras return 302 Moved Temporarily in response to DESCRIBE or OPTIONS query to redirect a client to another IP address for RTSP connection establishing. WCS supports this feature since build 5.2.179.

In this case, if the IP camera redirects requests to another address, and if client establishes connections separately to this camera and directly to the camera where requests are redirected, it is two different streams for WCS. The pulling agent s are created for every of those streams, and subscribers connect to one of those agents depending on address set on connection establishing.

RTSP pulled stream publishing with a given name

The ability was added to publish RTSP pulled stream with a given name since build5.2.479. The stream name should be set with`toStream`parameter of /rtsp/startup REST query, for example

```
POST /rest-api/rtsp/startup HTTP/1.1
Content-Length: 75
Content-Type: application/json

{
    "toStream": "stream1",
    "uri": "rtsp://myserver.com/live/myStream"
}
```

By default, if`toStream`paremeter is not set, the stream name will be formed from RTSP URI. If RTSP stream with such URI is already pulled, or teh stream with given name is already exists, server returns 409 Conflict.

If the name is set to the RTSP stream pulled, this stream can be played in CDN by its name (by default, CDN playback is not available for RTSP streams because they always are pulled locally).

Capturing H265 RTSP stream

Since build 5.2.1579 it is possible to capture RTSP stream published by camera as H265. To enable this, H265 should be added to supported codecs list

```
codecs=opus,alaw,ulaw,g729,speex16,g722,mpeg4-generic,telephone-event,h264,vp8,flv,mpv,h265
```

and to exclusion lists

```
codecs_exclude_sip=mpeg4-generic,flv,mpv,h265
codecs_exclude_sip_rtmp=opus,g729,g722,mpeg4-generic,vp8,mpv,h265
codecs_exclude_sfu=alaw,ulaw,g729,speex16,g722,mpeg4-generic,telephone-event,flv,mpv,h265
```

Stream captured may be played as WebRTC, RTMP, MSE, HLS with transcoding and as RTSP without transcoding

A stream should not contain B-frames! If B-frames occur in the stream, it may be played as RTSP without transcoding only

A first subscriber issue

Before WCS build 5.2.1760 RTSP streams may start to play with a huge delay for the first subscriber. This was due to key frames missing caused by subscriber thread starting after publisher thread. Since build 5.2.1760 the behavior was changed: publisher thread starts after subscriber thread. This may be reverted if necessary by the following parameter

```
agent_use_subscriber_listener=false
```

Stream timestamp fix

In some RTSP streams a frame timestamps may be in wrong order, for example two subsequent frames may have the same timestamp. Such stream may not be displayed or periodically display a gray square while playing via WebRTC. Since build 5.2.1794 the following parameter is availbel to fix a broken timestamps

```
jitter_buffer_attempt_to_correct_broken_timestamp=true
```

In this case RTSP capturing client log may contain a messages as follows

```
Non-monotonous timestamp in input stream; previous: 453424, current: 453424; changing to 453425. This may
result in incorrect timestamps in the output
```

and the problem stream should play normally.

Known issues

1. A stream containing B-frames does not play or plays with artifacts (latencies, lags)

Symptoms:

- a stream sent by the RTMP encoder does not play or plays with latencies or lags
- warnings in theclient log:

```
09:32:31,238 WARN 4BitstreamNormalizer - RTMP-pool-10-thread-5 It is B-frame!
```

Solution

- change the encoder settings so, that B-frames were not used (lower encoding profile, specify in the command line etc)
- transcode the stream, in this case there will be no B-frames in transcoded stream

2. AAC frames of type 0 are not supported by decoder and will be ignored while stream pulled playback

In this case, warnings will be displayed in theclient log:

```
10:13:06,815 WARN AAC - AudioProcessor-c6c22de8-a129-43b2-bf67-1f433a814ba9 Dropping AAC frame that starts with
0, 119056e500
```

Solution: use Fraunhofer AAC codec with the following parameter inflashphoner.propertiesfile

```
use_fdk_aac=true
```

3. When publishing and then playing and recording H264 + AAC stream video may be out of sync with sound, or no sound at all.

Symptoms: when playing H264 + AAC stream published on server, and when recordingsuch stream, sound is out of sync with video or absent

Solution:

a) set the following parameter inflashphoner.propertiesfile

```
disable_drop_aac_frame=true
```

This parameter also turns off AAC frames dropping.

b) use Fraunhofer AAC codec

```
use_fdk_aac=true
```

4. Sound may be distorted or absent when resampled to 11025 Hz

Symptoms: when H264 + AAC stream published on WCS server is played with AAC sample rate 11025 Hz, sound is distorted or absent

Solution: do not use 11025 Hz sample rate, or escape AAC sound resampling to this rate, for example, do not set this sample rate in SDP settings.

5. Connection to the IP camera is lost on error in any track (audio or video)

Symptoms: connection to the IP camera is lost if one of tracks returns error 4**.
Solution: this behavior is enabled by default. However if one-time errors are not critical and should not terminate broadcasting, in theflashphoner.propertiesfiles set

```
rtsp_fail_on_error_track=false
rtp_force_synchronization=true
```

6. All the characters in a stream name, that are not allowed in URI, should be escaped

Symptoms: RTSP stream is not played with 'Bad URI' error
Solution: any character that is not allowed in URI, should be escaped in stream URL, for example

```
rtsp://hostname/c@@lstream/channel1
```

should be set as

```
rtsp://hostname/c%40%40lstream/channel1
```

7. Some IP cameras do not support`cnonce`field in RTSP connection message header.

Symptoms: RTSP stream is played with VLC, but is not played with WCS.

Solution: set the following parameter inflashphoner.propertiesfile

```
rtsp_auth_cnonce=
```

with empty value.

8. Streams from some cameras cannot be played due to buffer size lack to write RBSP

Symptoms: RTSP stream is not playing with exceptions in server log

```
13:10:16,988 ERROR  BitstreamNormalizer - pool-56-thread-1 Failed to normalize SPS 674d002a95a81e0089f950
java.lang.RuntimeException: Failed to write sps rbsp
```

Solution: increase RBSP buffer size setting (1.5 by default)

```
h264_sps_rbsp_scale=2
```

9. Some camera streams can loose audio and video sync

Symptoms: RTSP stream freezes or cannot be played by HLS (some segments cannot be written), in stream statistics there are ubnormally big AV syncronization values

```
streams_synchronization=camera1/-21800;camera2/2079600704
```

Solution: in build before 5.2.1775 increase synchronization buffers for audio and video tracks

```
audio_incoming_buffer_size=100
video_incoming_buffer_size=100
```

since build 5.2.1775 increase force synchronization timeout for audio and video tracks

```
video_force_sync_timeout=1000
audio_force_sync_timeout=1000
```