

---

# User manual

(RTSP Server)

*Happytimesoft Technology Co., LTD*

---

## Declaration

All rights reserved. No part of this publication may be excerpted, reproduced, translated, annotated or edited, in any form or by any means, without the prior written permission of the copyright owner.

Since the product version upgrade or other reasons, this manual will subsequently be updated. Unless otherwise agreed, this manual only as a guide, this manual all statements, information, recommendations do not constitute any express or implied warranties.

---

[www.happytimesoft.com](http://www.happytimesoft.com)

---

# Table of Contents

<b>Chapter 1 Introduction.....</b>	<b>4</b>
<b>Chapter 2 Key features.....</b>	<b>5</b>
<b>Chapter 3 Function chart.....</b>	<b>8</b>
<b>Chapter 4 Configuration.....</b>	<b>9</b>
4.1 Configuration Templates.....	9
4.2 Configuring Node Description.....	12
4.2.1 System parameters.....	12
4.2.2 User node.....	13
4.2.3 Output node.....	14
4.2.4 Proxy node.....	15
4.2.5 Pusher node.....	16
4.2.6 Backchannel node.....	17
<b>Chapter 5 Data pusher.....</b>	<b>19</b>
<b>Chapter 6 RTSP over HTTP.....</b>	<b>23</b>
<b>Chapter 7 RTSP over Websocket.....</b>	<b>25</b>
<b>Chapter 8 Run RTSP Server.....</b>	<b>26</b>
<b>Chapter 9 Multiple capture devices support.....</b>	<b>27</b>

---

## Chapter 1 Introduction

Happytime RTSP Server is a complete RTSP server application. It can stream audio and video files in various formats.

It can also stream video from camera and live screen, stream audio from audio device.

It can stream H265, H264, MP4, MJPEG video stream and G711, G722, G726, AAC, OPUS audio stream.

These streams can be received/played by standards-compliant RTSP/RTP media clients.

It supports rtsp proxy function.

It supports audio back channel function.

It supports rtsp over http function.

It supports rtsp over https function.

It supports rtp multicast function.

Support for data pusher function.

Enjoying multimedia content from your computer can be a pleasant way for you to spend your free time. However, sometimes you might need to access it from various locations, such as a different computer or a handheld device, Happytime RTSP Server, that can help you achieve quick and efficient results.

---

## Chapter 2 Key features

The server can transmit multiple streams concurrently

It can stream audio and video files in various formats

It can stream audio from audio device

It can stream video from camera and live screen

It can stream H265, H264, MP4, MJPEG video stream

It can stream G711, G722, G726, AAC, OPUS audio stream

It supports rtsp over http function

It supports rtsp over https function

It supports rtsp over websocket function

It supports rtp multicast function

It supports data pusher function

It supports RTSP proxy function, as the following:



Support Audio Backchannel

Happytime rtsp server comply with ONVIF backchannel specification,  
the url is :

<https://www.onvif.org/specs/stream/ONVIF-Streaming-Spec-v1706.pdf>

---

### 5.3.2.1 Example 1: Server without backchannel support:

Client - Server: DESCRIBE rtsp://192.168.0.1 RTSP/1.0  
Cseq: 1  
User-Agent: ONVIF Rtsp client  
Accept: application/sdp  
Require: www.onvif.org/ver20/backchannel

Server - Client: RTSP/1.0 551 Option not supported  
Cseq: 1  
Unsupported: www.onvif.org/ver20/backchannel

### 5.3.2.2 Example 2: Server with Onvif backchannel support:

Client - Server: DESCRIBE rtsp://192.168.0.1 RTSP/1.0  
Cseq: 1  
User-Agent: ONVIF Rtsp client  
Accept: application/sdp  
Require: www.onvif.org/ver20/backchannel

Server - Client: RTSP/1.0 200 OK  
Cseq: 1  
Content-Type: application/sdp  
Content-Length: xxx

v=0  
o= 2890842807 IN IP4 192.168.0.1  
s=RTSP Session with audiobackchannel  
m=video 0 RTP/AVP 26  
a=control:rtsp://192.168.0.1/video  
a=recvonly  
m=audio 0 RTP/AVP 0  
a=control:rtsp://192.168.0.1/audio  
a=recvonly  
m=audio 0 RTP/AVP 0  
a=control:rtsp://192.168.0.1/audioback  
a=rtptime:0 PCMU/8000  
a=sendonly

---

Client - Server:           SETUP rtsp://192.168.0.1/**video** RTSP/1.0  
                          Cseq: 2  
                          Transport: RTP/AVP;unicast;client\_port=4588-4589

Server - Client:       RTSP/1.0 200 OK  
                          Cseq: 2  
                          Session: 123124;timeout=60  
                          Transport:RTP/AVP;unicast;client\_port=4588-4589;  
                          server\_port=6256-6257

Client - Server:           SETUP rtsp://192.168.0.1/**audio** RTSP/1.0  
                          Cseq: 3  
                          Session: 123124  
                          Transport: RTP/AVP;unicast;client\_port=4578-4579

Server - Client:       RTSP/1.0 200 OK  
                          Cseq: 3  
                          Session: 123124;timeout=60  
                          Transport:RTP/AVP;unicast;client\_port=4578-4579;  
                          server\_port=6276-6277

Client - Server:           SETUP rtsp://192.168.0.1/**audioback** RTSP/1.0  
                          Cseq: 4  
                          Session: 123124  
                          Transport: RTP/AVP;unicast;client\_port=6296-6297  
                          Require: www.onvif.org/ver20/backchannel

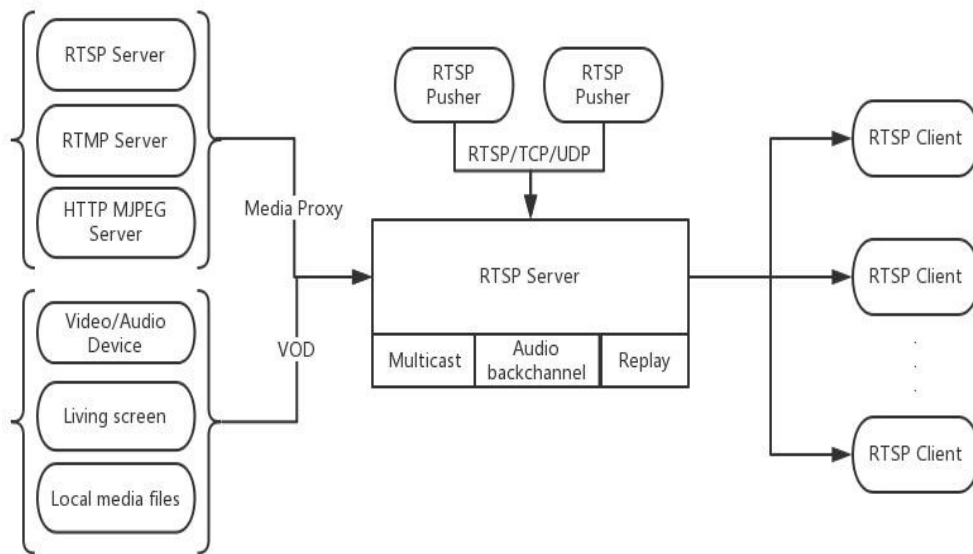
Server - Client:       RTSP/1.0 200 OK  
                          Cseq: 4  
                          Session: 123124;timeout=60  
                          Transport:RTP/AVP;unicast;client\_port=6296-6297;  
                          server\_port=2346-2347

Client - Server:           PLAY rtsp://192.168.0.1 RTSP/1.0  
                          Cseq: 5  
                          Session: 123124  
                          Require: www.onvif.org/ver20/backchannel

Server - Client:       RTSP/1.0 200 OK  
                          Cseq: 5  
                          Session: 123124;timeout=60

---

## Chapter 3 Function chart





---

## Chapter 4 Configuration

### 4.1 Configuration Templates

```
<?xml version="1.0" encoding="utf-8"?>
<config>
  <serverip></serverip>
  <serverip></serverip>
  <serverport>554</serverport>
  <loop_nums>1</loop_nums>
  <multicast>0</multicast>
  <metadata>1</metadata>
  <rtsp_over_http>1</rtsp_over_http>
  <http_port>8080</http_port>
  <rtsp_over_https>1</rtsp_over_https>
  <https_port>443</https_port>
  <need_auth>0</need_auth>
  <log_enable>1</log_enable>
  <log_level>1</log_level>

  <user>
    <username>admin</username>
    <password>admin</password>
  </user>
  <user>
    <username>user</username>
    <password>123456</password>
  </user>

  <output>
    <url>screenlive</url>
    <video>
      <codec>H264</codec>
      <width></width>
      <height></height>
      <framerate></framerate>
      <bitrate></bitrate>
```

---

```
</video>
<audio>
  <codec>G711U</codec>
  <samplerate>8000</samplerate>
  <channels>1</channels>
  <bitrate></bitrate>
</audio>
</output>
```

```
<output>
  <url></url>
  <video>
    <codec>H264</codec>
    <width></width>
    <height></height>
    <framerate></framerate>
    <bitrate></bitrate>
  </video>
  <audio>
    <codec>G711U</codec>
    <samplerate></samplerate>
    <channels></channels>
    <bitrate></bitrate>
  </audio>
</output>
```

```
<proxy>
  <suffix>proxy</suffix>
  <url></url>
  <user></user>
  <pass></pass>
  <transfer>TCP</transfer>
  <output>
    <video>
      <codec>H264</codec>
      <width>640</width>
```

---

```
        <height>480</height>
        <framerate>25</framerate>
        <bitrate></bitrate>
    </video>
    <audio>
        <codec>AAC</codec>
        <samplerate>32000</samplerate>
        <channels>2</channels>
        <bitrate></bitrate>
    </audio>
</output>
</proxy>
```

```
<pusher>
    <suffix>pusher</suffix>
    <video>
        <codec>H264</codec>
    </video>
    <audio>
        <codec>G711U</codec>
        <samplerate>8000</samplerate>
        <channels>1</channels>
    </audio>
    <transfer>
        <mode>UDP</mode>
        <ip></ip>
        <vport>50001</vport>
        <aport>50002</aport>
    </transfer>
    <output>
        <video>
            <codec>H264</codec>
            <width>640</width>
            <height>480</height>
            <framerate>25</framerate>
            <bitrate></bitrate>
```

---

```
</video>
<audio>
  <codec>AAC</codec>
  <samplerate>32000</samplerate>
  <channels>2</channels>
  <bitrate></bitrate>
</audio>
</output>
</pusher>

<backchannel>
  <codec>G711U</codec>
  <samplerate>8000</samplerate>
  <channels>1</channels>
</backchannel>
</config>
```

## 4.2 Configuring Node Description

### 4.2.1 System parameters

#### **<serverip>**

Specify the IP address RTSP server bindings, if not specified, the RTSP server will bind to the default routing interface IP address.

**Note: This node can configure multiple instances, meaning that the server can bind multiple IP addresses or domain names.**

#### **<serverport>**

Specify the port RTSP server binding, the default is 554.

#### **<loop\_nums>**

When streaming video files, specify the number of loop playback, -1 means infinite loop.

#### **<multicast>**

Whether to enable rtp multicast function, 0-disable, 1-enable.

#### **<metadata>**

---

Whether to enable the meta data stream, 0-disable, 1-enable.

**<rtsp\_over\_http>**

Whether to enable rtsp over http function, 0-disable,1-enable.

**<http\_port>**

Specify the HTTP service port for rtsp over http function.

**<rtsp\_over\_https>**

Whether to enable rtsp over https function, 0-disable,1-enable.

**<https\_port>**

Specify the HTTPS service port for rtsp over https function.

**<need\_auth>**

Whether enable the user authentication function,0-disable,1-enable

**<log\_enable>**

Whether enable the log function,0-disable,1-enable

**<log\_level>**

The log level:

TRACE	0
DEBUG	1
INFO	2
WARN	3
ERROR	4
FATAL	5

## 4.2.2 User node

**<user>** : Specify the login username password, it can configure multiple nodes

**<username>**

The login username

**<password>**

The login password

---

### 4.2.3 Output node

**<output>** : Specify the audio and video output parameters, it can configure multiple nodes

**<url>**

Match URL address, it can be filename, or file extension name. Such as:

screenlive : match live screen stream

videodevice : match camera video stream

\*.mp4 : match all mp4 media file

sample.flv : match sample.flv file

If not config this node, it will match all url as the audio/video default output parameters.

The match order from top to bottom, therefore the default output configuration should be placed in the last.

**<video>** : Specify the video output parameters

**<codec>**

Specify the video stream codec, it can specify the following value:

**H264** : output H264 video stream

**H265** : output H265 video stream

**MP4**: output MP4 video stream

**JPEG**: output MJPEG video stream

**<width>**

Specify the output video width, If 0 use the original video width (live screen stream use the screen width, camera stream use the default width)

**<height>**

Specify the output video height, If 0 use the original video height (live screen stream use the screen height, camera stream use the default height)

**<framerate>**

Specify the output video framerate, If 0 use the original video framerate (live screen use the default value 15, camera stream use the default value 25)

**<bitrate>**

Specify the output video bit rate, if 0, automatically calculate the output bit rate, the unit is kb/s.

Note: This parameter is valid only if encoding is required (eg screenlive, videodevice) or if transcoding is required.

---

**<audio>** : Specify the audio output parameters

**<codec>**

Specify the audio stream codec, it can specify the following value:

**G711A**: output G711 a-law audio stream

**G711U**: output G711 mu-law audio stream

**G722**: output G726 audio stream

**G726**: output G726 audio stream

**AAC**: output AAC audio stream

**OPUS**: output OPUS audio stream

**<samplerate>**

Specify the audio sample rate, it can specify the following values:

8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000

If 0 use the original audio sample rate (audio device stream use the default value 8000)

**<channels>**

Specify the audio channel number, 1 is mono, 2 is stereo

If 0 use the original audio channel number (audio device stream use the default value 2)

**Note : G726 only support mono.**

**<bitrate>**

Specify the output video bit rate, if 0, automatically calculate the output bit rate, the unit is kb/s.

Note: This parameter is valid only if encoding is required (eg screenlive, videodevice) or if transcoding is required.

## 4.2.4 Proxy node

**<proxy>** : Specify the rtsp proxy parameters, it can configure multiple nodes

**<suffix>**

Specify the rtsp stream suffix, you can play the proxy stream from:

rtsp://youip/suffix

**<url>**

The original rtsp stream address or http mjpeg stream address.

**<user> <pass>**

Specify the original rtsp stream or http mjpeg stream address login user and password information

---

**<transfer>**

Specify the rtsp client transfer protocol:

TCP: rtsp client uses RTP over TCP

UDP: rtsp client uses RTP over UDP

MULTICAST: rtsp client uses multicast

**<output>**

Specify the stream output parameter. If the parameter does not appear, use the parameters of the original RTSP stream. If it appears and the configured parameters are inconsistent with the parameters of the original RTSP stream, then the transcode output is performed.

The child nodes under this node are consistent with the meaning of the <output> node.

#### 4.2.5 Pusher node

**<pusher>** : Specify the data pusher parameters, it can configure multiple nodes

**<suffix>**

Specify the rtsp stream suffix, you can play the pusher stream from:

rtsp://youip/suffix

**<video>** : Specify the the input video data parameters

**<codec>**

Specify the video codec, it can specify the following value:

**H264** : H264 video stream

**H265** : H265 video stream

**JPEG**: MJPEG video stream

**<audio>** : Specify the input audio data parameters

**<codec>**

Specify the audio codec, it can specify the following value:

**G711A**: G711 a-law audio stream

**G711U**: G711 mu-law audio stream

**G722**: G726 audio stream

**G726**: G726 audio stream

**OPUS**: OPUS audio stream

**<samplerate>**



---

Specify the audio sample rate, it can specify the following values: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000

**<channels>**

Specify the audio channel number, 1 is mono, 2 is stereo

**Note : G726 only support mono.**

**<transfer>**: Specify the data transfer parameters

**<mode>**: **Specify the data transfer protocol**, it can specify the following value:

TCP: use TCP connection to transfer the data

UDP: use UDP connection to transfer the data

RTSP: use RTSP connection to transfer the data, it support FFMPEG rtsp pusher.

**<ip>**: Specified data receiving IP address, if there is no configuration, the default IP address is used.

**<vport>**: Specify the video data receiving port

**<aport>**: Specify the audio data receiving port

**<output>**

Specify the stream output parameter. If the parameter does not appear, use the parameters of the original pusher stream. If it appears and the configured parameters are inconsistent with the parameters of the original pusher stream, then the transcode output is performed.

The child nodes under this node are consistent with the meaning of the <output> node.

## 4.2.6 Backchannel node

**<backchannel>** : specify the audio back channel parameters

**<codec>**

Specify the audio back channel stream codec, it can specify the following value:

**G711A**: G711 a-law audio stream

**G711U**: G711 mu-law audio stream

**G722**: G726 audio stream

**G726**: G726 audio stream

**OPUS**: OPUS audio stream

**<samplerate>**

Specify the audio back channel sample rate, it can specify the following values:

---

8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000

If 0 use the default value 8000

**<channels>**

Specify the audio channel number, 1 is mono, 2 is stereo

If 0 use the default value 1

**Note : G726 only support mono.**

---

## Chapter 5 Data pusher

Data pusher means that RTSP server receives external data sources and then sends them out as RTSP streams.

The data pusher support TCP, UDP and RTSP mode.

Audio and video data are packaged and sent in RTP format.

If it is TCP mode, you need to add 4 bytes in front of the RTP header, as the following:

```
typedef struct
{
    uint32 magic      : 8;
    uint32 channel    : 8;
    uint32 rtp_len    : 16;
} RILF;
```

magic: 0x24

channel: 0

ttp\_len: the RTP load length, including RTP header,

You can download the examples of sending H264 data from the following link:

<http://happytimesoft.com/downloads/happytime-rtsp-h264-data-pusher-example.zip>

**Note:** If you use TCP or UDP mode data push, you need to add <pusher>tag in the rtsp server configuration file, specify the push audio and video parameters and push port, etc.

If you use RTSP mode to push data, no configuration is required. The url suffix of the pushed RTSP address can be any legal string.

---

If it is RTSP mode, it supports standard RTSP push stream, such as FFMPEG rtsp pusher.

FFMPEG rtsp over UDP:

```
ffmpeg -re -i test.mp4 -vcodec libx264 -acodec copy -preset ultrafast -f rtsp  
rtsp://yourip/pusher
```

FFMPEG rtsp over TCP:

```
ffmpeg -re -i test.mp4 -vcodec libx264 -acodec copy -preset ultrafast -f rtsp -rtsp_transport  
tcp rtsp://yourip/pusher
```

Examples of protocols pushed by RTSP are as follows:

C->S:

OPTIONS rtsp://192.168.3.27/mypusher RTSP/1.0 CSeq: 1

User-Agent: happytimesoft rtsp client

S->C:

RTSP/1.0 200 OK

Server: happytime rtsp server V5.0 CSeq: 1

Date: Tue, Sep 15 2020 00:45:17 GMT

Public: DESCRIBE, SETUP, PLAY, PAUSE, OPTIONS, TEARDOWN,  
GET\_PARAMETER, SET\_PARAMETER, ANNOUNCE, RECORD

C->S:

ANNOUNCE rtsp://192.168.3.27/mypusher RTSP/1.0 CSeq: 2

User-Agent: happytimesoft rtsp client Content-type: application/sdp Content-Length: 476

v=0

o=- 0 0 IN IP4 192.168.3.27

s=session

c=IN IP4 192.168.3.27

t=0 0

a=control:\*

m=video 0 RTP/AVP 96 a=rtpmap:96 H264/90000 a=fmtp:96

---

packetization-mode=1;profile-level-id=000015;sprop-parameter-sets=Z2QAFaw07AoDmwEQ  
AA ADABAAAAMDCPFi04A=,aO+8sA==  
a=control:realvideo m=audio 0 RTP/AVP 97  
a=rtpmap:97 MPEG4-GENERIC/8000/2  
a=fmtp:97  
streamtype=5;profile-level-id=1;mode=AAC-hbr;sizelength=13;indexlength=3;indexdeltaleng  
th= 3;config=159056E500  
a=control:realaudio

S->C:

RTSP/1.0 200 OK

Server: happytime rtsp server V5.0 CSeq: 2

Date: Tue, Sep 15 2020 00:45:17 GMT

C->S:

SETUP rtsp://192.168.3.27/mypusher/realvideo RTSP/1.0 CSeq: 3

Transport: RTP/AVP/TCP;unicast;interleaved=0-1 User-Agent: happytimesoft rtsp client

S->C:

RTSP/1.0 200 OK

Server: happytime rtsp server V5.0 CSeq: 3

Date: Tue, Sep 15 2020 00:45:17 GMT

Session: 41

Transport: RTP/AVP/TCP;unicast;interleaved=0-1

C->S:

SETUP rtsp://192.168.3.27/mypusher/realaudio RTSP/1.0 CSeq: 4

Session: 41

Transport: RTP/AVP/TCP;unicast;interleaved=2-3 User-Agent: happytimesoft rtsp client

S->C:

RTSP/1.0 200 OK

Server: happytime rtsp server V5.0 CSeq: 4

Date: Tue, Sep 15 2020 00:45:17 GMT

Session: 41

Transport: RTP/AVP/TCP;unicast;interleaved=2-3

---

C->S:

RECORD rtsp://192.168.3.27/mypusher RTSP/1.0 CSeq: 5

Session: 41 Range: npt=0.0-

User-Agent: happytimesoft rtsp client

S->C:

RTSP/1.0 200 OK

Server: happytime rtsp server V5.0 CSeq: 5

Date: Tue, Sep 15 2020 00:45:17 GMT

Session: 41

---

## Chapter 6 RTSP over HTTP

The key of RTSP over HTTP is to allow RTSP packets to communicate via HTTP port.

We know that the standard port of RTSP is 554, but due to various security policy configurations such as firewalls, there may be restrictions when the client accesses port 554, which prevents the normal transmission of RTSP packets.

But the HTTP port (port 80) is generally open, so there is the idea of letting RTSP packets pass through port 80, namely RTSP over HTTP

The details of RTSP over HTTP are as follows:

First, the client opens two socket connect to the rtsp server HTTP ports. We call these two sockets "data socket" and "command socket".

Step 1. The client sends an HTTP GET command through the "data socket" to request an RTSP connection.

Step 2. The server responds to the HTTP GET command through the "data socket" and responds with success/failure.

Step 3. The client creates a "command socket" and sends an HTTP POST command through the "command socket" to establish an RTSP session.

At this point, the auxiliary function of HTTP is completed, and the server does not return the client's HTTP POST command. Next is the standard process of RTSP on the HTTP port, but it needs to be completed through two sockets. The "command socket" is only responsible for sending, and the "data socket" is only responsible for receiving.

Step 4. The client sends RTSP commands (BASE64 encoding) through the "command socket".

Step 5. The server responds to the RTSP command (in plain text) through the "data socket".

Step 6. Repeat Step4-Step5 until the client sends the RTSP PLAY command and the server responds to the RTSP PLAY command.

---

Step 7. The server transmits audio and video data to the client through the "data socket"

After the data exchange is complete...

Step 8. The client sends the RTSP TEARDOWN command (BASE64 encoding and) through the "command socket"

Step 9. The server responds to the RTSP TEARDOWN command (in plain text) through the "data socket".

Step 10. Close the two sockets.



---

## Chapter 7 RTSP over WebSocket

First establish an HTTP connection, and then upgrade to the websocket protocol, RTSP over websocket protocol upgrade process:

C-->S:

GET /websocket HTTP/1.1 Host: 192.168.3.27

Upgrade: websocket Connection: Upgrade

Sec-WebSocket-Key: KSO+hOFs1q5SkEnx8bvp6w== Origin: http://192.168.3.27

Sec-WebSocket-Protocol: rtsp.onvif.org Sec-WebSocket-Version: 13

S-->C:

HTTP/1.1 101 Switching Protocols Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Accept: G/cEt4HtsYEnP0MnSVkKRk459gM= Sec-WebSocket-Protocol:  
rtsp.onvif.org

Sec-WebSocket-Version: 13

After the protocol upgrade is successful, perform normal rtsp protocol exchange, and send and receive data through websocket connection.

---

## Chapter 8 Run RTSP Server

The server is a console application.

Windows: to run the server, simply type "rtspserver".

Linux: to run the server, type "./start.sh", on linux platform, rtsp server run as daemon by default.

Note : The demo version has the following limitations Maximum support four concurrent sessions.

---

## Chapter 9 Multiple capture devices support

If your system have multiple audio capture device, you can use `rtsp://yourip:port/audiodeviceN`, the N to specify the audio capture device index, start from 0, such as:

`rtsp://192.168.0.100/audiodevice` ; stream audio from the first audio device  
`rtsp://192.168.0.100/audiodevice1` ; stream audio from the second audio device

If your system have multiple video capture device, you can use `rtsp://yourip:port/videodeviceN`, the N to specify the video capture device index, start from 0, such as:

`rtsp://192.168.0.100/videodevice` ; stream video from the first video device  
`rtsp://192.168.0.100/videodevice1` ; stream video from the second video device

If your system have multiple monitors, you can use `rtsp://yourip:port/screenliveN`, the N to specify the monitor index, start from 0, such as:

`rtsp://192.168.0.100/screenlive` ; stream living screen from the first monitor  
`rtsp://192.168.0.100/screenlive1` ; stream living screen the second monitor