User manual

（RTSP Server）

HAPPYTIME SOFTWARE CO., LIMITED

Declaration

Table of Contents

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, living screen and application windows, 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.

It supports SRTP and RTSPS.

It supports HTTP notify.

It supports data pusher/publish 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 living screen.

It can stream video from application windows.

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.

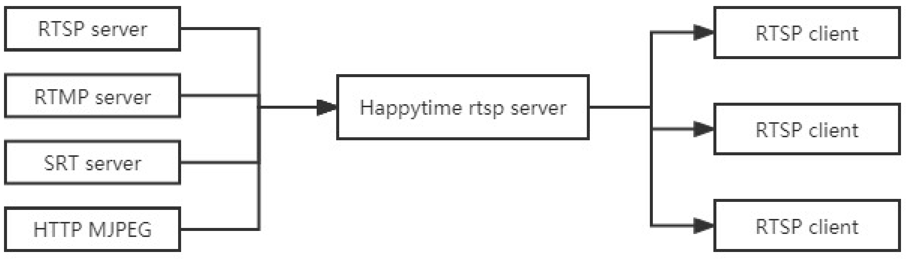
It supports rtp multicast function.

It supports data pusher/publish function.

It supports SRTP and RTSPS.

It supports HTTP notify.

It supports RTSP proxy function, as the following:



It supports audio backchannel.

Happytime rtsp server complies with onvif audio backchannel specification, please refer to the link below for specification details:

https://happytimesoft.com/knowledge/audio-back-channel.html

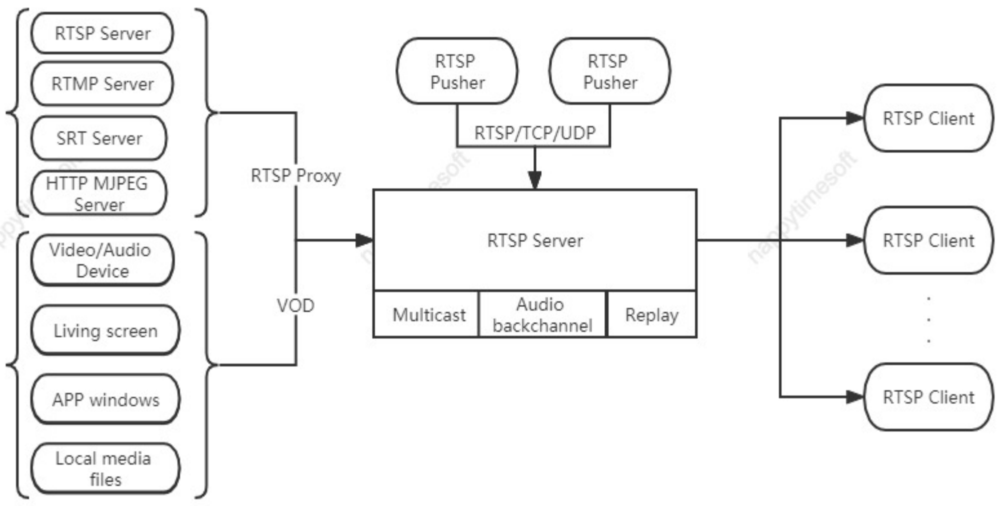
It supports audio and video playback.

Happytime rtsp server complies with onvif audio and video playback specification, please refer to the link below for specification details:

https://happytimesoft.com/knowledge/audio-video-playback.html

Chapter 3 Function chart

The function diagram of Happytime rtsp sever is as follows:



Chapter 4 Configuration

If no configuration file is specified at startup, the default configuration file rtspserver.cfg will be used.

4.1 Configuration Templates

```
<?xml version="1.0" encoding="utf-8"?>
<config>
    <serverip></serverip>
    <rtsp_port>554</rtsp_port>
    <rtsps_enable>1</rtsps_enable>
    <rtsps_port>322</rtsps_port>
    <rtsps_cert>ssl.ca</rtsps_cert>
    <rtsps_key>ssl.key</rtsps_key>
    <loop_nums>1</loop_nums>
    <multicast>0</multicast>
    <udp_base_port>22000</udp_base_port>
    <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>
    <https_cert>ssl.ca</https_cert>
    <https_key>ssl.key</https_key>
    <ipv6_enable>1</ipv6_enable>
    <need_auth>0</need_auth>
    <log_enable>1</log_enable>
    <log_level>1</log_level>

    <http_notify>
            <on_connect></on_connect>
            <on_play></on_play>
            <on_publish></on_publish>
            <on_done></on_done>
```

```
        <notify_method></notify_method>
</http_notify>

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

<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>
    <ondemand>0</ondemand>
    <output>
        <video>
```

```
                <codec>H264</codec>
                <width></width>
                <height></height>
                <framerate></framerate>
                <bitrate></bitrate>
            </video>
            <audio>
                <codec>AAC</codec>
                <samplerate></samplerate>
                <channels></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>RTSP</mode>
            <ip></ip>
            <vport>50001</vport>
            <aport>50002</aport>
        </transfer>
        <output>
            <video>
                <codec></codec>
                <width></width>
                <height></height>
                <framerate></framerate>
                <bitrate></bitrate>
            </video>
            <audio>
                <codec></codec>
                <samplerate></samplerate>
                <channels></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 (Support IPv4 and IPv6) or domain name of the RTSP server, if not specified, the rtsp server will listen to all network interfaces.

<rtsp_port>

Specify the RTSP server port, the default is 554.

Note: On Linux systems, ports below 1024 are reserved by the system and require root privileges to be used.

<rtsps_enable>

Whether to enable the RTSPS (RTSP over SSL/TLS) server, 0 - disable, 1 - enable.

<rtsps_port>

Specify the RTSPS server port, the default is 322.

Note: On Linux systems, ports below 1024 are reserved by the system and require root privileges to be used.

<rtsps_cert>

Specify the RTSPS server certificate file.

<rtsps_key>

Specify the RTSPS server private key file.

Note: The certificate file ssl.ca and private key file ssl.key provided by default are self-signed local hosts certificates, only for testing purposes (browsers may pop up untrusted certificate warnings) and cannot be used in formal deployment environments.

<loop_nums>

When streaming media 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.

<udp_base_port>

UDP media transmission base port, RTSP over UDP mode assign UDP port on this base port.

Each rtsp session needs to assign 8 UDP ports, video RTP/RTCP port, audio RTP/RTCP port, METADATA stream RTP/RTCP port, audio backchannel RTP/RTCP port.

Note: If you need to run multiple instances of rtsp server, this parameter for each instance cannot be the same.

The media base port interval of each instance should be greater than the number of rtsp sessions that the instance needs to support multiplied by 8.

For example, the media base port of instance 1 is 22000, which needs to support up to 100 sessions. Then the media base port of instance 2 should be configured at least as 22000+100 * 8=22800.

<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.

Note: On Linux systems, ports below 1024 are reserved by the system and require root privileges to be used.

<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.

Note: On Linux systems, ports below 1024 are reserved by the system and require root privileges to be used.

<https_cert>

Specify the HTTPS service certificate file.

<https_key>

Specify the HTTPS service key file.

Note: The certificate file ssl.ca and key file ssl.key provided by default are self-signed local hosts certificates, only for testing purposes (browsers may pop up untrusted certificate warnings) and cannot be used in formal deployment environments.

<ipv6_enable>

Indicates whether IPv6 is enabled, 0-disable, 1-enable.

Note: If the server does not specify a server ip in <serverip> and the <ipv6_enable> is 1, and the server has an IPv6 address, the client can connect to the server through the IPv6 address.

<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 http_notify

<http_notify>: Specify the HTTP notification callback address.

<on_connect>:

Sets HTTP connection callback. When clients issues connect command an HTTP request is issued and command processing is suspended until it returns result code. If HTTP 200 code is returned then RTSP session continues.

HTTP request receives a number of arguments. POST method is used with application/x-www-form-urlencoded MIME type. The following arguments are passed to caller:

protocol=rtsp

call=connect

addr - client IP address

url - URL requested by the client

name - stream name

clientid - rtsp session id

<on_play>:

Sets HTTP play callback. Each time a clients issues play command an HTTP request is issued and command processing is suspended until it returns result code. If HTTP 200 code is returned then RTSP session continues.

HTTP request receives a number of arguments. POST method is used with application/x-www-form-urlencoded MIME type. The following arguments are passed to caller:

protocol=rtsp

call=play

addr - client IP address

url - URL requested by the client

name - stream name

clientid - rtsp session id

<on_publish>:

The same as on_play above with the only difference that this node sets callback on publish command. Instead of remote pull push is performed in this case.

<on_done>:

Sets play/publish terminate callback. All the above applies here. However HTTP status code is not checked for this callback.

<notify_method>

Sets HTTP method for notifications. Default is POST with application/x-www-form-urlencoded content type.

Support GET and POST method.

4.2.3 User node

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

<username>

The login username.

<password>

The login password.

4.2.4 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, or special suffix. Such as:

screenlive: match living screen stream.

videodevice: match camera video stream.

audiodevice: match microphone audio 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 25, 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 (screenlive, videodevice) or 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 G722 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 audio 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 (screenlive, videodevice) or transcoding is required.

4.2.5 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://[serverip]:[rtsp_port]/[suffix]

<url>

The original rtsp/rtmp/srt/http mjpeg stream address.

<user> <pass>

Specify the original rtsp/rtmp/srt/http mjpeg stream login username and password.

<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

<ondemand>

Connect on demand, 1-Connect when needed, 0-Always keep connected.

<output>

Specify the stream output parameter. If the parameter does not appear, use the parameters of the original RTSP/RTMP/SRT/HTTP MJPEG stream. If it appears and the configured parameters are inconsistent with the parameters of the original RTSP/RTMP/SRT/HTTP MJPEG 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 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://[serverip]:[rtsp_port]/[suffix]

<video>: Specify 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

MP4: MPEG4 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: G722 audio stream

G726: G726 audio stream

OPUS: OPUS audio stream

AAC: AAC 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 supports standard rtsp push, such as FFMPEG rtsp push.

<ip>: Specify 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.

Note : <ip>, <vport>, <aport> these parameters are valid when **<mode>** is TCP or UDP.

<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.

4.2.7 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

AAC: AAC 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/publish

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

The data pusher supports 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

rtp_len: the RTP payload length, including RTP header.

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 push.

FFMPEG rtsp over UDP:

ffmpeg -re -i test.mp4 -vcodec libx264 -acodec copy -preset ultrafast -f rtsp rtsp://[serverip]:[rtsp_port]/pusher

FFMPEG rtsp over TCP:

ffmpeg -re -i test.mp4 -vcodec libx264 -acodec copy -preset ultrafast -f rtsp -rtsp_transport tcp rtsp://[serverip]:[rtsp_port]/pusher

After the above ffmpeg rtsp push command is running, you can use the following address to play the rtsp stream:

rtsp://[serverip]:[rtsp_port]/pusher

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 sockets 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".

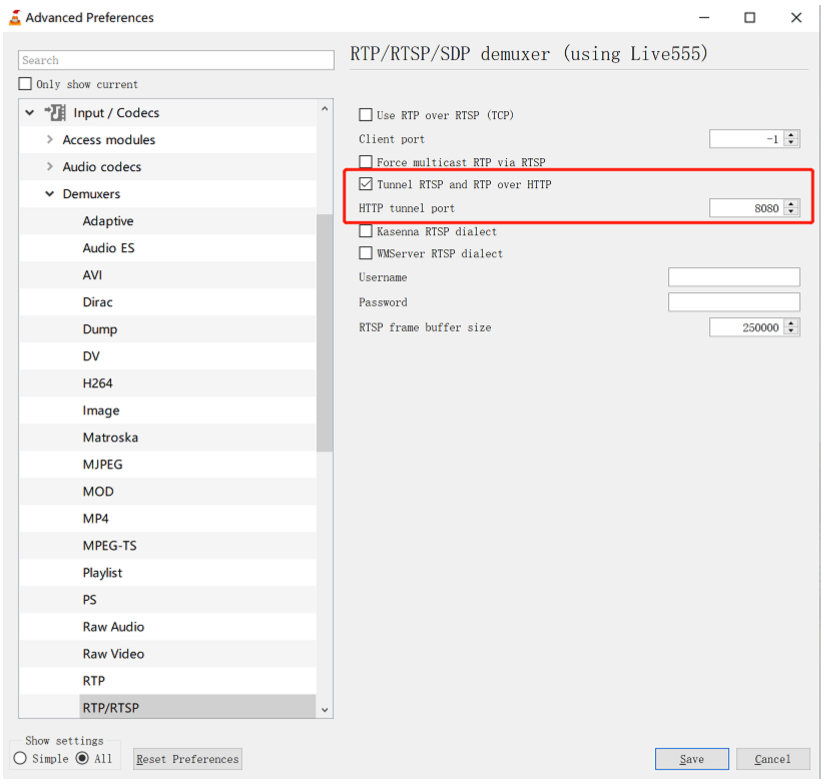
After the data exchange is complete...

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.

VLC supports RTSP over HTTP, the settings as the follows:

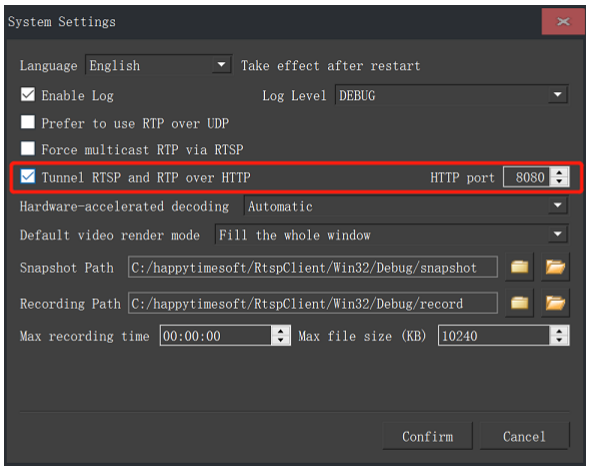

Advanced Preferences

RTP/RTSP/SDP demuxer (using Live555)

☐ Use RTP over RTSP (TCP)
Client port: -1
☐ Force multicast RTP via RTSP
☑ Tunnel RTSP and RTP over HTTP
HTTP tunnel port: 8080
☐ Kasenna RTSP dialect
☐ WMServer RTSP dialect
Username:
Password:
RTSP frame buffer size: 250000

Show settings: ○ Simple [● All]

Happytime rtsp client (https://happytimesoft.com/products/rtsp-client/index.html) supports RTSP over HTTP, the setting as the following:


System Settings

Language: English — Take effect after restart
☑ Enable Log — Log Level: DEBUG
☐ Prefer to use RTP over UDP
☐ Force multicast RTP via RTSP
☑ Tunnel RTSP and RTP over HTTP — HTTP port: 8080
Hardware-accelerated decoding: Automatic
Default video render mode: Fill the whole window
Snapshot Path: C:/happytimesoft/RtspClient/Win32/Debug/snapshot
Recording Path: C:/happytimesoft/RtspClient/Win32/Debug/record
Max recording time: 00:00:00 — Max file size (KB): 10240

Happytime rtsp client also supports rtsp streams starting with http:// or https://.

If url starts with http://, it is considered to be a rtsp over http stream.

If url starts with https://, it is considered to be a rtsp over https stream.

Chapter 7 RTSP over WebSocket

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

Client-->Server:

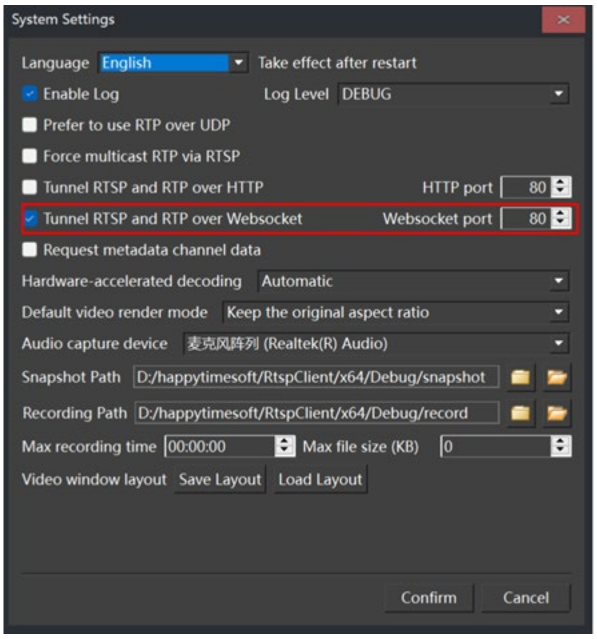
GET /websocket HTTP/1.1 Host: 192.168.3.27
Upgrade: websocket Connection: Upgrade
Sec-WebSocket-Key: KSO+hOFs1q5SkEnx8bvp6w== Origin: http://192.168.3.27
Sec-WebSocket-Protocol: rtsp.onvif.org Sec-WebSocket-Version: 13

Server-->Client:

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, send and receive data through WebSocket connection.

Happytime rtsp client (https://happytimesoft.com/products/rtsp-client/index.html) supports RTSP over WebSocket, the setting as the following:



System Settings

Language: English — Take effect after restart
✓ Enable Log — Log Level: DEBUG
☐ Prefer to use RTP over UDP
☐ Force multicast RTP via RTSP
☐ Tunnel RTSP and RTP over HTTP — HTTP port: 80
✓ Tunnel RTSP and RTP over Websocket — Websocket port: 80
☐ Request metadata channel data
Hardware-accelerated decoding: Automatic
Default video render mode: Keep the original aspect ratio
Audio capture device: 麦克风阵列 (Realtek(R) Audio)
Snapshot Path: D:/happytimesoft/RtspClient/x64/Debug/snapshot
Recording Path: D:/happytimesoft/RtspClient/x64/Debug/record
Max recording time: 00:00:00 — Max file size (KB): 0

Happytime rtsp client also supports rtsp streams starting with ws://.

If url starts with ws://, it is considered to be a rtsp over WebSocket stream.

Chapter 8 RTP Multicast

To enable the rtp multicast function, it needs to specify the <multicast> to 1 in the configuration file.

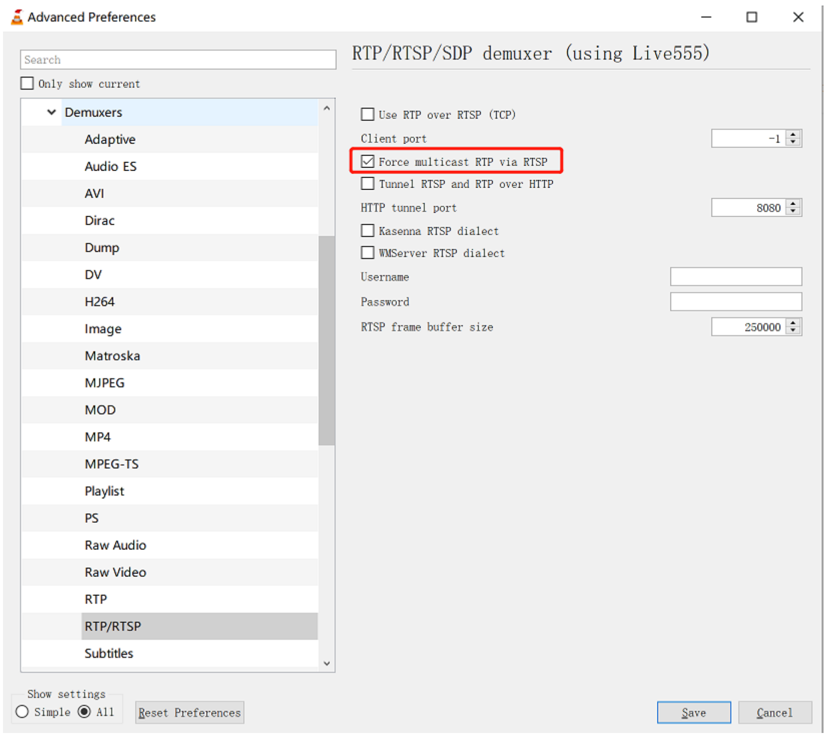
The rtsp server does not support the configuration of multicast addresses.

Different rtsp stream addresses use multicast, randomly assigned multicast addresses starting with 232.

Different rtsp sessions use rtp multicast to play the same rtsp stream, using the same multicast address. Only the first rtsp session sends audio and video data, and subsequent sessions refer to the first rtsp session.

The rtp multicast stream address is the same as the normal rtsp stream address.

Use VLC to test rtp multicast, use the following settings:



RTP/RTSP/SDP demuxer (using Live555)

☐ Use RTP over RTSP (TCP)
Client port -1
☑ Force multicast RTP via RTSP
☐ Tunnel RTSP and RTP over HTTP
HTTP tunnel port 8080
☐ Kasenna RTSP dialect
☐ WMServer RTSP dialect
Username
Password
RTSP frame buffer size 250000

Show settings: Simple [✓ All]

Chapter 9 Audio back channel

The backchannel connection handling is done using RTSP [RFC 2326]. Therefore a mechanism is introduced which indicates that a client wants to build up a backchannel connection. RTSP provides feature-tags to deal with such functionality additions. A device that supports bi-directional connections (e.g., audio or metadata connections) shall support the introduced RTSP extensions.

9.1 RTSP Require- Tag

The RTSP standard [RFC 2326] can be extended by using additional headers objects. For that purpose a Require tag is introduced to handle special functionality additions (see [RFC 2326], 1.5 Extending Rtsp and 12.32 Require).

The Require-tag is used to determine the support of this feature. This header shall be included in any request where the server is required to understand that feature to correctly perform the request.

A device that supports backchannel shall understand the backchannel tag:

www.onvif.org/ver20/backchannel

An RTSP client that wants to build up an RTSP connection with a data backchannel shall include the Require header in its requests.

9.2 Connection setup for a bi- directional connection

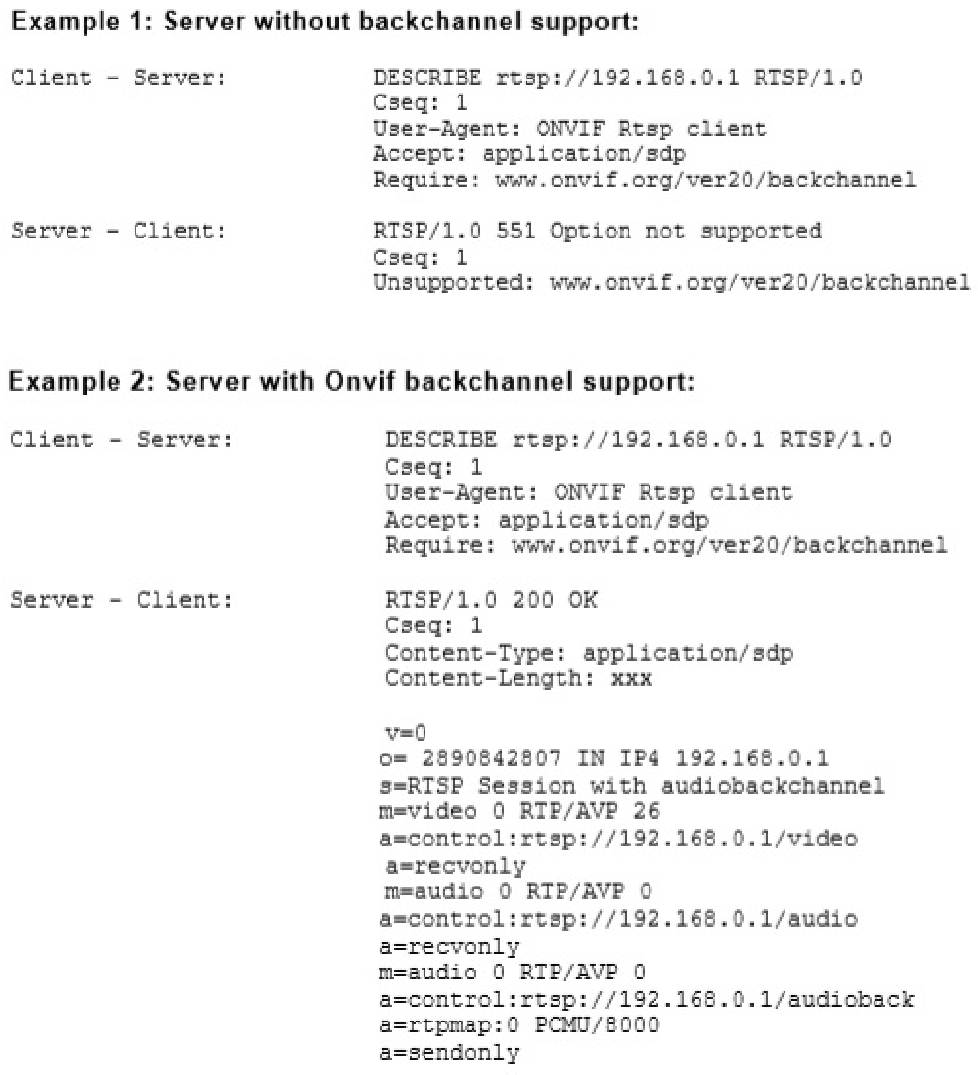
A client shall include the feature tag in its DESCRIBE request to indicate that a bidirectional data connection shall be established.

A server that understands this Require tag shall include an additional media stream in its SDP file.

An RTSP server that does not understand the backchannel feature tag or does not support bidirectional data connections shall respond with an error code 551 Option not supported according to the RTSP standard. The client can then try to establish an RTSP connection without backchannel.

A SDP file is used to describe the session. To indicate the direction of the media data the server shall include the a=sendonly in each media section representing media being sent from the client to the server and a=recvonly attributes in each media section representing media being sent from the server to the client.

The server shall list all supported decoding codecs as own media section and the client chooses which one is used. The payload type and the encoded bitstream shall be matched with one of the a=rtpmap fields provided by the server so that the server can properly determine the audio decoder.



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
```

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=rtpmap:0 PCMU/8000
                          a=sendonly
```

This SDP file completely describes the RTSP session. The server gives the client its control URLs to setup the streams.

In the next step the client can setup the sessions:

```
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
```

The third setup request establishes the audio backchannel connection.

In the next step the client starts the session by sending a PLAY request.

```
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
```

After receiving the OK response to the PLAY request the client MAY start sending audio data to the server. It shall not start sending data to the server before it has received the response.

The Require-header indicates that a special interpretation of the PLAY command is necessary. The command covers both starting of the video and audio stream from NVT to the client and starting the audio connection from client to server.

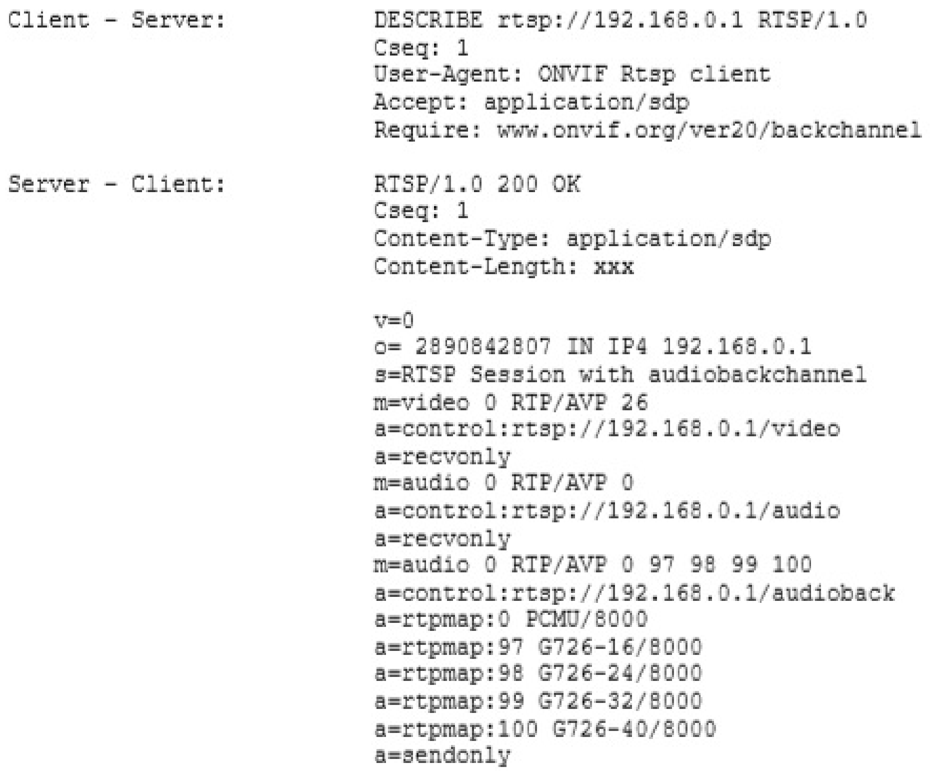
To terminate the session the client sends a TEARDOWN request.

```
Client - NVT:       TEARDOWN rtsp://192.168.0.1 RTSP/1.0
                    Cseq: 6
                    Session: 123124
                    Require: www.onvif.org/ver20/backchannel

NVT - Client:       RTSP/1.0 200 OK
                    Cseq: 6
                    Session: 123124
```

9.3 Example

Server with Onvif backchannel support (with multiple decoding capability)

If a device supports multiple audio decoders as backchannel, it can signal such capability by listing multiple a=rtpmap fields illustrated as follows.



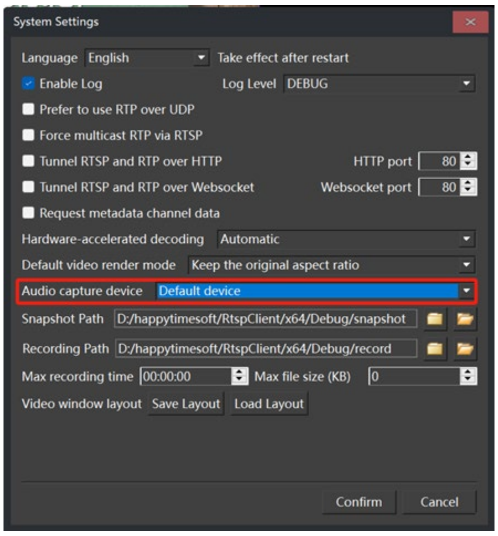
```
Client - Server:          DESCRIBE rtsp://192.168.0.1 RTSP/1.0
                          Cseq: 1
                          User-Agent: ONVIF Rtsp client
                          Accept: application/sdp
                          Require: www.onvif.org/ver20/backchannel

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 97 98 99 100
                          a=control:rtsp://192.168.0.1/audioback
                          a=rtpmap:0 PCMU/8000
                          a=rtpmap:97 G726-16/8000
                          a=rtpmap:98 G726-24/8000
                          a=rtpmap:99 G726-32/8000
                          a=rtpmap:100 G726-40/8000
                          a=sendonly
```

9.4 Test audio backchannel

Happytime RTSP client

(https://www.happytimesoft.com/products/rtsp-client/index.html) supports audio backchannel. The testing steps are as follows:

1. In the system settings UI, select audio capture device, as the following:


System Settings

Language: English — Take effect after restart

- [✓] Enable Log — Log Level: DEBUG
- [ ] Prefer to use RTP over UDP
- [ ] Force multicast RTP via RTSP
- [ ] Tunnel RTSP and RTP over HTTP — HTTP port: 80
- [ ] Tunnel RTSP and RTP over Websocket — Websocket port: 80
- [ ] Request metadata channel data

Hardware-accelerated decoding: Automatic

Default video render mode: Keep the original aspect ratio

Audio capture device: Default device

Snapshot Path: D:/happytimesoft/RtspClient/x64/Debug/snapshot

Recording Path: D:/happytimesoft/RtspClient/x64/Debug/record

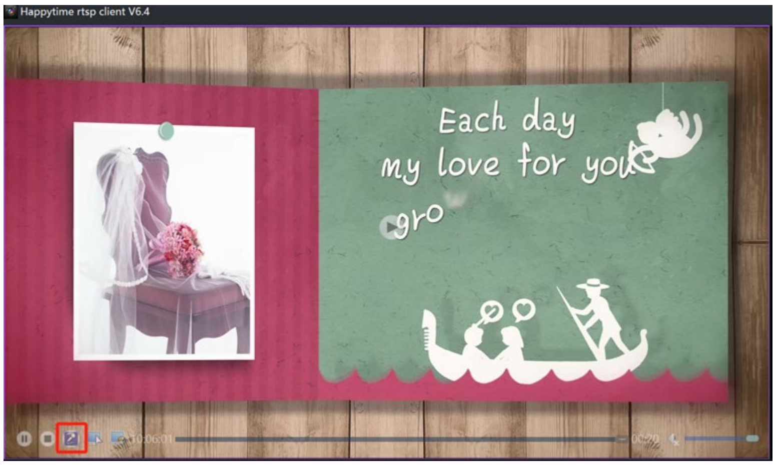
Max recording time: 00:00:00 — Max file size (KB): 0

Video window layout: Save Layout | Load Layout

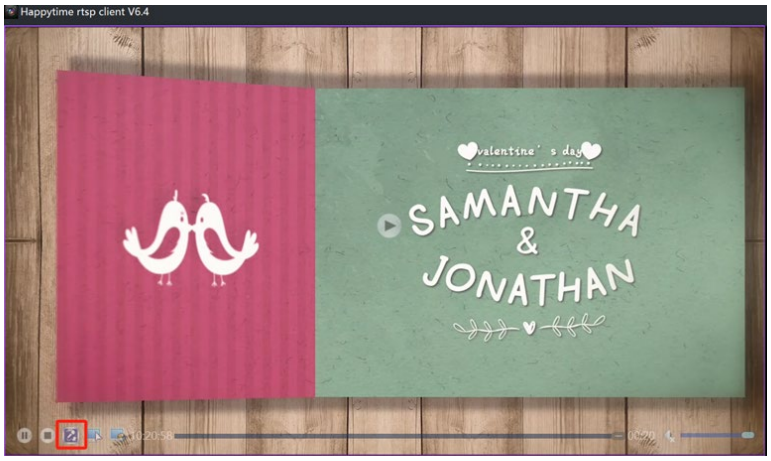
Confirm | Cancel

2. Play a normal RTSP stream address.

3. Click on the microphone icon in the playback window, as shown below:



4. If the audio backchannel connection is successful, the microphone icon will change to the following image:



5. Speak into the microphone, and the RTSP server will play the received audio backchannel data.

Chapter 10 Run RTSP Server

The server is a console application.

Windows: to run the server, type "rtspserver" or double click rtspserver.exe.

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

rtsp server supports the following command line options:

-c config specify the configuration file

-c option specifies the configuration file,if not specified, the default configuration rtspserver.cfg is used.

-l [device|videodevice|audiodevice|window]

-l device list available video and audio capture device

-l videodevice list available video capture device

-l audiodevice list available audio capture device

-l window list available application window

Below is sample output of -l device:

rtspserver -l device

```
Available video capture device :
index : 0, name : FaceTime HD Camera (Built-in)

Available audio capture device :
index : 0, name : Headset Microphone (Apple Audio Device)
index : 1, name : Internal Digital Microphone (Apple Audio Device)
```

Note: The demo version has the following limitations:

Maximum support four concurrent sessions.

Chapter 11 Multiple capture devices support

If your system has multiple audio capture device, you can use:

rtsp://[serverip]:[rtsp_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 has multiple video capture device, you can use:

rtsp://[serverip]:[rtsp_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 has multiple monitors, you can use:

rtsp://[serverip]:[rtsp_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 from the second monitor

The audio index or video index represents which device can run the following command to view:

```
rtspserver -l device
```

videodevice or audiodevice can also specify the device name, such as:

rtsp://[serverip]:[rtsp_port]/videodevice=testvideo

Run the following command to get the device name:

```
rtspserver -l device
```

Note: There can be no spaces in the device name, if the device name contains spaces, you need to use %20 instead of spaces.

If the device name is “FaceTime HD Camera”, the rtsp stream address is:

rtsp://[serverip]:[rtsp_port]/videodevice=FaceTime%20HD%20Camera

Chapter 12 Capture application window

The rtsp server supports capturing application windows, you can use the following command to list valid application windows:

rtspserver -l window

Below is a sample output of the command:

```
Available window name :

C:\Windows\system32\cmd.exe - RtspServer.exe  -l window
user manual.doc - WPS Office
Rtsp-server Project - Source Insight - [Main.cpp]
RtspServer
```

You can use the following URL to capture the specified application window:

rtsp://[serverip]:[rtsp_port]/window=[window title]

Note: window title case insensitive

Such as:

rtsp://[serverip]:[rtsp_port]/window=rtspserver

Note: There can be no spaces in the window title, if the window title contains spaces, you need to use %20 instead of spaces. Such as:

rtsp://[serverip]:[rtsp_port]/window=user%20manual.doc%20-%20WPS%20Office

Chapter 13 Support URL parameters

RTSP server supports URL parameters, with the following format:

Taking playing the test.mp4 file as an example:

rtsp://[serverip]:[rtsp_port]/test.mp4?param1=value1&param2=value2

Pararm1 and param2 represent URL parameters, while value1 and value2 represent the values of param1 and param2.

Note: The parameter value specified through the URL has a higher priority than the parameters configured in the configuration file.

The RTSP server supports the following parameters:

srtp: Whether to enable SRTP.

Possible values

0 - disable srtp

1 - enable srtp

t: Specify the transmission method.

Possible values:

unicast

multicast

p: Specify transmission protocol.

Possible values:

udp : rtp over udp

tcp : rtp over tcp

ve: Specify video encoding.

Possible values:

H264

H265

JPEG

MP4

fps：Specify video frame rate.

w: Specify video width.

h: Specify video height.

vb：Specify video bitrate, unit is kb/s.

ae：Specify audio encoding.

Possible values:

PCMU：g711 ulaw

PCMA: g711 alaw

G726

G722

OPUS

AAC

sr: Specify audio samplerate.

ch：Specify the number of audio channels

ab：Specify audio bitrate.

Example:

Using unicast RTP over UDP mode, output video encoding as H264, resolution as 1920 * 1080, frame rate as 30, bit rate as 4000K, audio as AAC, sampling rate as 44100, stereo RTSP stream:

rtsp://[serverip]:[rtsp_port]/test.mp4?t=unicast&p=udp&ve=h264&w=1920&h=1080&fps=30&vb=4000&ae=aac&sr=44100&ch=2

Chapter 14 Support SRTP

Happytime rtsp server supports Secure Real-time Transport Protocol (secure RTP or SRTP) for publishing and playback. The encryption keys can be passed in the SDP data. The open-source tools FFmpeg and FFplay can be used for SRTP testing.

14.1 SRTP for publishing

Happytime RTSP Pusher supports SRTP publishing. Modify the RTSP Pusher configuration file to set <SRTP> under the <Pusher> tag to 1, and RTSP Pusher will use SRTP for data publishing:

RTSP Pusher uses SRTP to publish configuration examples:

```
<pusher>
    <src>test.mp4</src>
    <transfer>
        <mode>RTSP</mode>
        <rtspurl>rtsp://192.168.3.36/myapp/live</rtspurl>
        <user>admin</user>
        <pass>admin</pass>
    </transfer>
    <video>
      ……
    </video>
    <audio>
      ……
    </audio>
    <metadata>0</metadata>
    <srtp>1</srtp>
</pusher>
```

You will see SDP information in the ANNOUNCE request message of rsp pusher. It should look something like this:

```
v=0
o=- 0 0 IN IP4 192.168.3.36
c=IN IP4 192.168.3.36
s=session
```

```
t=0 0
a=control:*
m=video 0 RTP/SAVP 96
a=rtpmap:96 H264/90000
a=fmtp:96
packetization-mode=1;profile-level-id=42801F;sprop-parameter-sets=Z0KAH5ZSAKALdJQEBAUAAAMAAQAAAwAyhA==,aMuNSA==
a=control:realvideo
a=crypto:1 AES_CM_128_HMAC_SHA1_80 inline:KSO+hOFs1q5SkEnx8bvp67Om2zyHDD6ZJF4NHAa3
m=audio 0 RTP/SAVP 97
a=rtpmap:97 MPEG4-GENERIC/44100/2
a=fmtp:97
streamtype=5;profile-level-id=1;mode=AAC-hbr;sizelength=13;indexlength=3;indexdeltalength=3;config=121056E500
a=control:realaudio
a=crypto:1 AES_CM_128_HMAC_SHA1_80 inline:R96zEk3IQ7uLph8DWn0JOCUfXdTL/Jb1RTsTDYkK
```

14.2 SRTP for playback

The RTSP server supports the URL parameter srtp to specify whether SRTP playback is enabled or not.

Example:

Enable SRTP to play test.mp4 files:

rtsp://[serverip]:[rtsp_port]/test.mp4?srtp=1

Use FFplay or Happytime rtsp client to test playback.

Chapter 15 Support RTSPS

For encrypted communication, RTSP can be tunneled over a secure transport layer, most commonly using TLS (Transport Layer Security). This secured version is often referred to as RTSPS and typically uses port 322.

Happytime rtsp server supports RTSPS. Modify the configuration file rtspserver.cfg, set <rtsps_enable> to 1, set <rtsps_port> to specify the RTSPS service port, and set <rtsps_cert> and <rtsps_key> to specify the RTSPS certificate and private key:

```
<config>
    <serverip></serverip>
    <rtsp_port>554</rtsp_port>
    <rtsps_enable>1</rtsps_enable>
    <rtsps_port>322</rtsps_port>
    <rtsps_cert>ssl.ca</rtsps_cert>
    <rtsps_key>ssl.key</rtsps_key>
    ……
</config>
```

Note: On Linux systems, ports below 1024 are reserved by the system and require root privileges to use.

Note: The certificate file ssl.ca and key file ssl.key provided by default are self-signed local hosts certificates, only for testing purposes (browsers may pop up untrusted certificate warnings) and cannot be used in formal deployment environments.

Run the rtsp server, and you will see the RTSPS playback address in the output window of the rtsp server.

To test RTSPS streams, you can use ffplay, Happytime rtsp client, or Happytime media client for testing.