
User manual

(Portable RTC)

HAPPYTIME SOFTWARE CO., LIMITED

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

Table of Contents

| | |
|---|-----------|
| Chapter 1 Introduction | 4 |
| Chapter 2 Key features | 5 |
| Chapter 3 Configuration | 6 |
| 3.1 Configuration Templates | 6 |
| 3.2 sConfiguring Node Description | 8 |
| 3.2.1 <i>System parameters</i> | 8 |
| 3.2.2 <i>User node</i> | 10 |
| 3.2.3 <i>application node</i> | 11 |
| Chapter 4 Run Portable RTC | 14 |
| Chapter 5 Multiple capture devices support | 17 |
| Chapter 6 Capture application window | 19 |

Chapter 1 Introduction

Happytime portable rtc is a WebRTC solution implemented in C language, designed to provide users with compact and easily portable real-time communication (RTC) capabilities. The project not only implements the core functions of WebRTC, but also extends a variety of media transmission and sharing capabilities, including desktop sharing, application window sharing, audio and video media file transmission, camera video transmission, and rtsp/rtmp/srt streaming transmission.

Happytime portable rtc is developed based on C language and adopts the core technology stack of WebRTC, including SRTP/SRTCP, DTLS, ICE and other protocols, to ensure the security and real-time performance of communication. At the same time, the project has undergone a lot of optimization and expansion, making it more compact, portable, and supporting more media transmission and sharing functions.

Happytime portable rtc is a powerful and easily portable WebRTC solution that supports simultaneous access by multiple browser clients and provides rich media transmission and sharing capabilities. Whether it is for remote education, remote meetings, live streaming sharing, or medical consultations, portable rtc can provide efficient and stable real-time communication services.

Chapter 2 Key features

1. Multi-browser client support

Happytime portable rtc supports simultaneous access by multiple web browser clients, enabling real-time communication across platforms and browsers.

2. Desktop and application window sharing

Users can send live video streams of their desktop or specific application windows to multiple browser clients for remote presentations, teaching, and collaboration.

3. Audio and video media file transfer

In addition to real-time audio and video streaming, Happytime portable rtc also supports sending pre-recorded audio and video media files to multiple browser clients, enriching the content and form of communication.

4. Camera video transmission

Users can easily send the video stream of the local camera to multiple browser clients in real time, enabling video calls, video conferencing, and other functions.

5. rtsp/rtmp/srt streaming

In addition to the protocols natively supported by WebRTC, Happytime portable rtc also supports sending video streams of streaming media protocols such as RTSP, RTMP, and SRT to multiple browser clients, enabling the utilization and sharing of more types of video sources.

6. Two-way audio intercom

Happytime portable rtc supports two-way audio communication, that is, real-time audio calls, providing a high-quality voice communication experience.

7. Small and lightweight

Because Happytime portable rtc is written in C language, it is small and lightweight, and can be easily deployed and run on various hardware and software platforms. This makes it particularly useful in embedded systems, mobile devices, or resource-constrained environments.

Chapter 3 Configuration

If no configuration file is specified when starting portable RTC, the default configuration file `portablertc.cfg` will be used.

3.1 Configuration Templates

```
<?xml version="1.0" encoding="utf-8"?>
<config>
  <http_enable>1</http_enable>
  <http_server_ip></http_server_ip>
  <http_port>80</http_port>
  <https_enable>1</https_enable>
  <https_server_ip></https_server_ip>
  <https_port>443</https_port>
  <cert_file>ssl.ca</cert_file>
  <key_file>ssl.key</key_file>
  <http_max_user>16</http_max_user>
  <need_auth>0</need_auth>
  <ipv6_enable>1</ipv6_enable>
  <loop_nums>-1</loop_nums>
  <log_enable>1</log_enable>
  <log_level>1</log_level>
  <ws_signal_server>ws://192.168.3.36:8080</ws_signal_server>
  <wss_signal_server>wss://192.168.3.36:5443</wss_signal_server>

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

  <application>
    <name>myapp</name>
```

```
<output>
  <url></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>
```

```
<proxy>
  <suffix>proxy</suffix>
  <url></url>
  <user></user>
  <pass></pass>
  <transfer>TCP</transfer>
  <ondemand>1</ondemand>
  <output>
    <video>
      <codec>H264</codec>
      <width></width>
      <height></height>
      <framerate></framerate>
      <bitrate></bitrate>
    </video>
    <audio>
      <codec>G711A</codec>
```

```
        <samplerate>8000</samplerate>
        <channels>1</channels>
        <bitrate></bitrate>
    </audio>
</output>
</proxy>

</application>
</config>
```

3.2 sConfiguring Node Description

3.2.1 System parameters

<http_enable>

Whether to enable http server, 0-disable, 1-enable.

<http_server_ip>

Specify the http server ip, if not specified, it will listen on all interfaces.

<http_port>

Specify the http server port, default is 80.

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

<https_enable>

Whether to enable HTTPS server, 0-disable, 1-enable.

<https_server_ip>

Specify the https server ip, if not specified, it will listen on all interfaces.

<https_port>

Specify the https server port, default is 443.

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

<cert_file>

Specify the https server certificate file.

<key_file>

Specify the https server 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.

<http_max_user>

http max user connection.

<need_auth>

Enable authentication, 0--disable, 1--enable.

<ipv6_enable>

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

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

<loop_nums>

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

<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 |

<ws_signal_server>

The signal server address starts with ws://.

If accessing the portable RTC stream with http://, the portablertc will to use the ws signaling service address to register.

<wss_signal_server>

The security signal server address starts with wss://.

If accessing the portable RTC stream with https://, the portablertc will to use the wss secure signaling service address to register.

Note: If the portable RTC stream is accessed using https:// and the WSS secure signaling server address is not configured, some browsers may fail to access the portable RTC stream.

Note: Before starting portable RTC, the signal server needs to be started first. The Signal server startup interface will output the signaling server address.

3.2.2 User node

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

<username>

The login username

<password>

The login password

3.2.3 application node

<application> : it can configure multiple nodes

<name>: Application Name

The stream address is :

http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/FILENAME

https://[https_server_ip]:[https_port]/portablertc?streamid=[application-na
me]/FILENAME

The [application-name] is name tag value.

3.2.3.1 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 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

<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

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)

Note : G711A and G711U only support 8000 samplerate

<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)

<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.

3.2.3.2 Proxy node

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

<suffix>

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

http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]/[suffix]

https://[https_server_ip]:[https_port]/portablertc?streamid=[application-name]/[suffix]

<url>

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

<user> **<pass>**

Specify the original rtsp/rtmp/srt 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 RTP 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 stream. If it appears and the configured parameters are inconsistent with the parameters of the original RTSP/RTMP/SRT stream, then the transcode output is performed.

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

Chapter 4 Run Portable RTC

The portable RTC is a console application.

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

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

Note: Before starting portable RTC, it need to start the STUN server and signal server.

Note: If the portable RTC is on the same network as the browser client, the signal server does not need to configure ice servers.

The signal server provide registration and connection establishment services.

STUN service is used to assist in connection establishment and NAT penetration.

After decompressing the download package, open the happytime-signal-server directory, which contains the signal server execution program.

We have tested the stunman and coturn STUN servers.

You can download the stunman or coturn server and run it as STUN server.

Stunman server download address:

<https://www.stunprotocol.org/>

Coturn server download address:

<https://github.com/coturn/coturn>

The portable rtc supports the following command line options:

`-c config` specify the configuration file

`-c` option specifies the configuration file, if not specified, the default configuration portablertc.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:

```
portablertc -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)
```

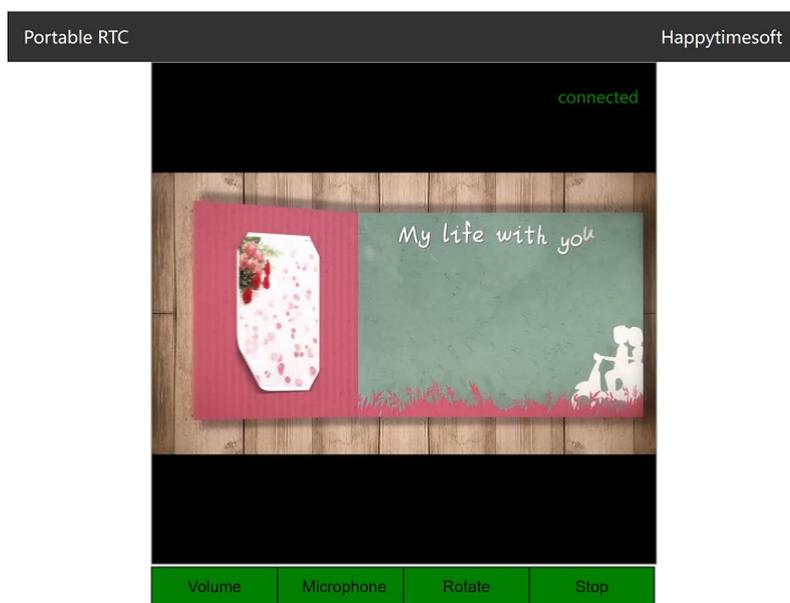
```
index : 1, name : Internal Digital Microphone (Apple Audio Device)
```

Note : The demo version has the following limitations:

Maximum support four concurrent sessions.

After starting portable RTC, it will output the stream address in the console window.

Open the portable RTC stream address in the browser, and the interface is as follows:



Note: The browser client computer needs to turn off the firewall, otherwise the client may not receive the audio and video data sent by portable RTC through the UDP port.

If the stream being played has audio, clicking the "Volume" button will play the audio, and clicking the "Volume" button again will stop playing the audio.

If a secure streaming address (starting with https://) is opened, click the "Microphone" button and the browser will ask if the microphone is enabled, click confirm to enable the audio two-way talk function.

Note: Non secure stream address (starting with http://), cannot enable microphone. For browser security reasons, HTTP streaming does not allow the microphone to be turned on.

Note: Press F12 on the browser page (most browsers have a shortcut key for entering debug mode) to view the console output of the page. If the peer-to-peer connection is unsuccessful, you can check the reason for the connection failure.

Chapter 5 Multiple capture devices support

1. If your system have multiple audio capture device, you can use following url to play:

```
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/audiodeviceN
```

```
https://[https_server_ip]:[https_port]/portablertc?streamid=[application-na
me]/audiodeviceN
```

The N to specify the audio capture device index, start from 0, such as:

http://192.168.3.36/portablertc?streamid=myapp/audiodevice ; stream audio from the first audio device

http://192.168.3.36/portablertc?streamid=myapp/audiodevice1 ; stream audio from the second audio device

2. If your system have multiple video capture device, you can use

```
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/videodeviceN
```

```
https://[https_server_ip]:[https_port]/portablertc?streamid=[application-na
me]/videodeviceN
```

The N to specify the video capture device index, start from 0, such as:

http://192.168.3.36/portablertc?streamid=myapp/videodevice ; stream video from the first video device

http://192.168.3.36/portablertc?streamid=myapp/videodevice1 ; stream video from the second video device

3. If your system have multiple monitors, you can use

```
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/screenliveN
```

```
https://[https_server_ip]:[https_port]/portablertc?streamid=[application-na
me]/screenliveN
```

The N to specify the monitor index, start from 0, such as:

http://192.168.3.36/portablertc?streamid=myapp/screenlive ; stream living screen from the first monitor

http://192.168.3.36/portablertc?streamid=myapp/screenlive1 ; stream living screen the second monitor

The audio index or video index represents which device can execute *portablertc -l device* to view.

videodevice or audiodevice can also specify the device name, such as:
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/videodevice=testvideo

https://[https_server_ip]:[https_port]/portablertc?streamid=[application-name]
/videodevice=testvideo

Execute the *portablertc -l device* command to get the device name.

Note that 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 (Built-in)”, the stream address is:
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/videodevice=FaceTime%20HD%20Camera%20(Built-in)

https://[https_server_ip]:[https_port]/portablertc?streamid=[application-name]
/videodevice=FaceTime%20HD%20Camera%20(Built-in)

Chapter 6 Capture application window

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

```
portablertc -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:

```
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/window=[window title]
```

```
https://[https_server_ip]:[https_port]/portablertc?streamid=[application-na
me]/window=[window title]
```

Note : window title case insensitive

Such as :

```
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/window=rtspserver
```

```
https://[https_server_ip]:[https_port]/portablertc?streamid=[application-na
me]/window=rtspserver
```

Note that 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:

```
http://[http_server_ip]:[http_port]/portablertc?streamid=[application-name]
/window=user%20manual.doc%20-%20WPS%20office
```

```
https://[https_server_ip]:[https_port]/portablertc?streamid=[application-na
me]/window=user%20manual.doc%20-%20WPS%20office
```