rtspserver

RtspServerLive555 C++ library

v1.0.0

Table of contents

Overview

The **RtspServerLive555** is a C++ library provides RTSP server. The library based on popular opensource library **Live555** library (source code not included, just linked). The **RtspServerLive555** library provides simple interface to include in any C++ project. The library allows you to create an RTSP server (multicast by default) without the complexity of the **Live555** library. User can set general RTSP server parameters: RTSP server port, IP, stream name, user name and password. Server works frame-by-frame: user put video frame (encoded by H264 and JPEG). The RTSP server provided by library compatible with all popular RTSP clients: **ffmpeg**, **gstreamer**, **VLC**, **Milestone** etc. The library depends on **Frame** class (source code included) and **Live555** (linked in CMake and must be installed in the OS). The library tested on **Linux OS**.

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	20.11.2023	First version which supports H264 and JPEG codecs.

RtspServerLive555 class description

Class declaration

RtspServerLive555 class declared in **RtspServerLive555.h** file. Class declaration:

```cpp
class RtspServerLive555
{
public:

    /// Class constructor.
    RtspServerLive555();

    /// Class destructor.
    ~RtspServerLive555();

    /// Get class version string.
    static std::string getVersion();

    /// Init RTSP server.
    bool init(int port,
              std::string streamName = "unicast",
              std::string ip = "",
              std::string userName = "",
              std::string password = "");

    /// Put video frame into server for streaming.
    bool sendFrame(cr::video::Frame& frame);

    /// Close RTSP server.
    void close();
};
```

getVersion method

getVersion() method return string of current class version. Method declaration:

```cpp
static std::string getVersion();
```

Method can be used without **RtspServerLive555** class instance:

```cpp
cout << "RtspServerLive555 version: " << RtspServerLive555::getVersion() << endl;
```

init method

init(...) method designed to initialize RTSP server. If server already initialized the method will reinitialize server. If RTPS server parameters changed user can call init(...) method to reinitialize server. Method declaration:

```
bool init(int port,
          std::string streamName = "unicast",
          std::string ip = "",
          std::string userName = "",
          std::string password = "");
```

Parameter	Value
port	RTSP server port. Must have values from 0 to 65535. Recommended value from 1024 to 65535.
streamName	RTSP stream name. For example "unicast", "multicast", "live" etc.
ip	RTSP IP address.
userName	Username for RTSP server authentication. Must be initialized with the "password".
password	Password for RTSP server authentication. Must be initialized with the "userName".

Returns: TRUE if initialization is done or FALSE if not.

sendFrame method

sendFrame(...) method designed to send Frame to RTSP server. Method declaration:

```
bool sendFrame(cr::video::Frame& frame);
```

Parameter	Value
frame	Frame object to send RTSP stream. H264 and JPEG frame formats are supported.

Returns: TRUE if frame is sent or FALSE if not.

close method

close() method closes RTSP server if it is open. Method declaration:

```
void close();
```

Example

Test application shows how to create RTSP server and send frames.

```cpp
#include <iostream>
#include <string>
#include "RtspServerLive555.h"

int main(void)
{
    // Init RTSP server.
    cr::rtsp::RtspServerLive555 server;
    if(!server.init(8554, "unicast", "127.0.0.1", "user", "password"))
        return -1;

    // Create H264 frame.
    cr::video::Frame frameH264(640, 480, cr::video::Fourcc::H264);

    // Main loop.
    while(1)
    {
        // --------------------------------
        // Prepare h264 frame data ...
        // --------------------------------

        // Send frame to RTSP server.
        if(!server.sendFrame(frameH264))
            cout << "Can not send frame to rtsp server" << endl;
    }
    return 1;
}
```

Dependencies

The **RtspServerLive555** is a C++ library uses the **Live555** library, which in turn relies on OpenSSL for secure network communication. To effectively utilize the **RtspServerLive555** library, it is essential to ensure that both Live555 and OpenSSL are correctly installed and configured on your system.

How to install OpenSSL

- Run command :

  ```
  sudo apt-get install libssl-dev
  ```

How to install Live555 by using package manager

- Run command :

```
sudo apt-get install liblivemedia-dev
```

If you get error from package manager, go to next step and install live555 from source code.

Important Note : There will be a compilation error if your system has live555 from source code installation and package manager installation at the same time. In order to solve problem you should remove one of them. For example:

```
sudo apt-get remove liblivemedia-dev
```

How to install Live555 from source code

- First you need to download Live555 from **official website**.
- Extract .tar.gz file.

```
tar -xz live.(version).tar.gz
```

- Run **genMakefiles** command with proper OS name. You check supported OS in live folder see files config.OSname .

```
cd live
```

```
./genMakefiles <os-platform>
```

for example:

```
./genMakefiles linux-64bit
```

if you are getting compile error. Add -std=c++20 -DNO_STD_LIB flags to CPLUSPLUS_FLAGS param in **config.os-platform** for example **config.linux-64-bit**:

```
COMPILE_OPTS =      $(INCLUDES) -fPIC -I/usr/local/include -I. -O2 -
DSOCKLEN_T=socklen_t -D_LARGEFILE_SOURCE=1 -D_FILE_OFFSET_BITS=64
C =         c
C_COMPILER =        cc
C_FLAGS =       $(COMPILE_OPTS)
CPP =           cpp
CPLUSPLUS_COMPILER =    c++
CPLUSPLUS_FLAGS =   $(COMPILE_OPTS) -Wall -DBSD=1 -Wno-deprecated -std=c++20 -
DNO_STD_LIB
OBJ =           o
LINK =          c++ -o
LINK_OPTS =     -L.
CONSOLE_LINK_OPTS = $(LINK_OPTS)
LIBRARY_LINK =      ar cr
```

```
LIBRARY_LINK_OPTS =
LIB_SUFFIX =            a
LIBS_FOR_CONSOLE_APPLICATION = -lssl -lcrypto
LIBS_FOR_GUI_APPLICATION =
EXE =
```

- Run make command

```
make
```

- Install to system

```
sudo make install
```

Build and connect to your project

Typical commands to build **RtspServerLive555** library:

```
git clone https://github.com/ConstantRobotics-Ltd/RtspServerLive555.git (if you have
access)
cd RtspServerLive555
mkdir build
cd build
cmake ..
make
```

If you want connect RtspServerLive555 library to your CMake project as source code you can make follow. For example, if your repository has structure:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
```

You can add repository **RtspServerLive555** as submodule by command:

```
cd <your respository folder>
git submodule add https://github.com/ConstantRobotics-Ltd/RtspServerLive555.git
3rdparty/RtspServerLive555
```

In you repository folder will be created folder **3rdparty/RtspServerLive555** which contains files of **RtspServerLive555** repository. New structure of your repository:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    RtspServerLive555
```

Create CMakeLists.txt file in **3rdparty** folder. CMakeLists.txt should contain:

```
cmake_minimum_required(VERSION 3.13)

################################################################################
## 3RD-PARTY
## dependencies for the project
################################################################################
project(3rdparty LANGUAGES CXX)

################################################################################
## SETTINGS
## basic 3rd-party settings before use
################################################################################
# To inherit the top-level architecture when the project is used as a submodule.
SET(PARENT ${PARENT}_YOUR_PROJECT_3RDPARTY)
# Disable self-overwriting of parameters inside included subdirectories.
SET(${PARENT}_SUBMODULE_CACHE_OVERWRITE OFF CACHE BOOL "" FORCE)

################################################################################
## CONFIGURATION
## 3rd-party submodules configuration
################################################################################
SET(${PARENT}_SUBMODULE_RTSP_SERVER_LIVE555             ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_RTSP_SERVER_LIVE555)
    SET(${PARENT}_RTSP_SERVER_LIVE555                   ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_RTSP_SERVER_LIVE555_TEST              OFF CACHE BOOL "" FORCE)
endif()

################################################################################
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
################################################################################
if (${PARENT}_SUBMODULE_RTSP_SERVER_LIVE555)
    add_subdirectory(RtspServerLive555)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **RtspServerLive555** to your project and excludes test application from compiling. Your repository new structure will be:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    CMakeLists.txt
    RtspServerLive555
```

Next you need include folder 3rdparty in main **CMakeLists.txt** file of your repository. Add string at the end of your main **CMakeLists.txt**:

```
add_subdirectory(3rdparty)
```

Next you have to include RtspServerLive555 library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} RtspServerLive555)
```

Done!