Developer manual

（GB28181 Device）

Happytimesoft Technology Co., LTD

Declaration

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。

Table of Contents

Chapter 1 Build

1.1 Dependent libraries

The GB28181 device depend on the following libraries:

ffmpeg 4.0 (Embedded platform, if you do live streaming, it do not need ffmpeg library)

ALSA 1.1.6 (stream from audio device on linux platform)

SRT version 1.5.0

Openssl version 1.1.1g

ZLIB version 1.2.11

XCB 1.9 and above (stream from living screen on linux platform)

1.2 Windows Platform

Use VS2017 or later open GB28181Device.sln to build

1.3 Linux Platform

make clean

make

1.4 MAC Platform

make -f mac.mk clean

make -f mac.mk

1.5 IOS Platform

make -f ios.mk clean

make -f ios.mk

1.6 Android Platform

make -f android.mk clean

make -f android.mk

1.7 Build APK for android

Use QtCreator open android.pro to build

1.8 Embedded Platform

Modify embed.mk to specify cross-compiler, execute the following command to compile:

make -f embed.mk clean

make -f embed.mk

1.9 Compilation parameter

EPOLL : Enable EPOLL mode for SOCKET multiplexing on Linux

MEDIA_FILE : Support streaming from local media files

MEDIA_DEVICE : Support streaming from devices (camera, microphone, living screen, application windows)

MEDIA_PROXY : Support streaming from other media servers (rtsp server,http mjpeg server)

RTMP_PROXY : Support streaming from RTMP servers (Need to enable MEDIA_PROXY)

SRT_PROXY : Support streaming from SRT servers (Need to enable MEDIA_PROXY)

MEDIA_LIVE : Support streaming from encoded media data (for embeded platform)

OVER_HTTP : Support rtsp over http (for rtsp client of media proxy)

OVER_WEBSOCKET : Support rtsp over websocket (for rtsp client of media proxy)

HTTPS : Support rtsp over https (for rtsp client of media proxy)

Chapter 2 Build ALSA library on linux platform

If you enable MEDIA_DEVICE compiler macro on linux platform, you need to build ALSA library, please execute the following commands to build:

cd third
tar jxf alsa-lib-1.1.6.tar.bz2 cd alsa-lib-1.1.6
./configure --enable-shared=yes make install

On 64bit linux platform:

ln -s /usr/lib64/alsa-lib /usr/lib/alsa-lib

Chapter 3 Install XCB library on linux platform

If you enable MEDIA_DEVICE compiler option on linux platform, you need to install XCB library.

On CentOS system, please run the following commands to install:

yum install libxcb

yum install libxcb-devel

On ubuntu system, please run the following commands to install:

sudo apt install xcb

sudo apt install libxcb-util0-dev

Chapter 4 Embedded developers

4.1 Compile

On embedded platform, please use embed.mk to compile:

make -f embed.mk clean

make -f embed.mk

Please modify the following options in embed.mk file to specify your cross compiler:

CCOMPILE -- Specify the C file compiler

CPPCOMPILE -- Specify the C++ file compiler

LINK -- Specify the object file linker

COMPILEOPTION -- Add compile options

LINKOPTION -- Add link options

INCLUDEDIR -- Add header file search path

LIBDIRS -- Add library file search path

SHAREDLIB -- Add library files that need to be linked

4.2 Feed custom data

In **sua_parse_live_url** function, set p_sua->media_info struct.

In **live_video.cpp** file, implement the following functions:

initCapture()

stopCapture()

captureThread()

If has audio, in **live_audio.cpp** file, implement the following functions:

initCapture()

stopCapture()

captureThread()

Note: Implement captureThread function, call **procData** function after getting audio and video encoded data

CLiveVideo and CLiveAudio are stub class, the purpose is to obtain encoded data from the hardware encoder and send it to the video and audio transmission queue.

When the sip client connects for the first time, call the initCapture interface, where the hardware encoder can be initialized.

When the last sip client disconnects, call stopCapture(), here you can stop the hardware encoder.

captureThread, get the encoded data and then call procData to send the data to the video and audio queue.

If you have your own data capture thread, you can also call **media_live_put_video** and **media_live_put_audio** to send data to the video and audio queue.

4.3 Modify configuration

Modify the configuration file and specify the <channel>.<media_url> as live, as the following:

```
<channel>
  <cid>34020000001310000001</cid>
  <cname>channel1</cname>
  <media_url>live</media_url>
  <output>
    ......
  </output>
</channel>
```

4.4 Support multiple streams

First, in the **sua_parse_live_url** function, use media url to determine which channel's stream is requested, set the media information and the index, such as:

```
if (strncasecmp(p_url "live1", strlen("live1")) == 0)
{
    p_rua->media_info.is_live = 1;

    p_rua->media_info.has_video = 1;
    p_sua->media_info.v_info.framerate = 25;
    p_sua->media_info.v_info.width = 1920;
    p_sua->media_info.v_info.height = 1080;
    p_rua->media_info.v_index = 0;
}
else if (strncasecmp(p_url, "live2", strlen("live2")) == 0)
{
    p_rua->media_info.is_live = 1;

    p_rua->media_info.has_video = 1;
    p_sua->media_info.v_info.framerate = 25;
    p_sua->media_info.v_info.width = 1280;
    p_sua->media_info.v_info.height = 720;
    p_rua->media_info.v_index = 1;
}
```

In the getStreamNums function in live_video.cpp, modify the support stream numbers, such as:

```
CLiveVideo::getStreamNums()
{
   return 2;
}
```

in the **initCapture** and **captureThread** function, use the Member variables **m_nStreamIndex** to init and capture the streams.

The corresponding configuration file channel configuration is as follows:

```
<channel>
  <cid>34020000001310000001</cid>
  <cname>channel1</cname>
  <media_url>live1</media_url>
  <output>
    ......
  </output>
</channel>
<channel>
  <cid>34020000001310000002</cid>
  <cname>channel2</cname>
  <media_url>live2</media_url>
  <output>
    ......
  </output>
</channel>
```

Chapter 5 Dll files

On windows platform, after the successful compilation of the project, if running or debugging, there is no dynamic library file, please download our demo version from the following link:

x86 version:

https://happytimesoft.com/downloads/happytime-gb28181-device.zip

x64 version:

https://www.happytimesoft.com/downloads/happytime-gb28181-device-x 64.zip

Then copy all dll files from the installed path to your project bin path.

Chapter 6 Development

1. In **gb28181.cpp** file, the following functions need attention:

sip_call_cb

Call status callback

sip_audio_cb

Audio data reception callback

sip_call_in

Called when a call is received

sip_audio_call_out

Initiate an audio call

sip_ntf_cb

SIP event callback

2. In **gb28181_msg.cpp** file, , the following functions need attention:

gb28181_record_info_res_get

Get records information, need to fill the GB28181_RECORD_INFO_RES structure

gb28181_device_status_res_get

Get device status, need to fill the GB28181_DEVICE_STATUS_RES structure

gb28181_device_info_res_get

Get device information, need to fill the GB28181_DEVICE_INFO_RES structure

gb28181_catalog_res_get

Get catalog information, need to fill the GB28181_CATALOG_RES structure

gb28181_config_res_get

Get device configurations, need to fill the GB28181_CONFIG_RES structure

gb28181_preset_res_get

Get presets, need to fill the GB28181_PRESET_RES structure

gb28181_device_control

Implement device control operations

gb28181_device_config

Set device configuration

gb28181_broadcast_notify

Audio broadcast

Chapter 7 Android development

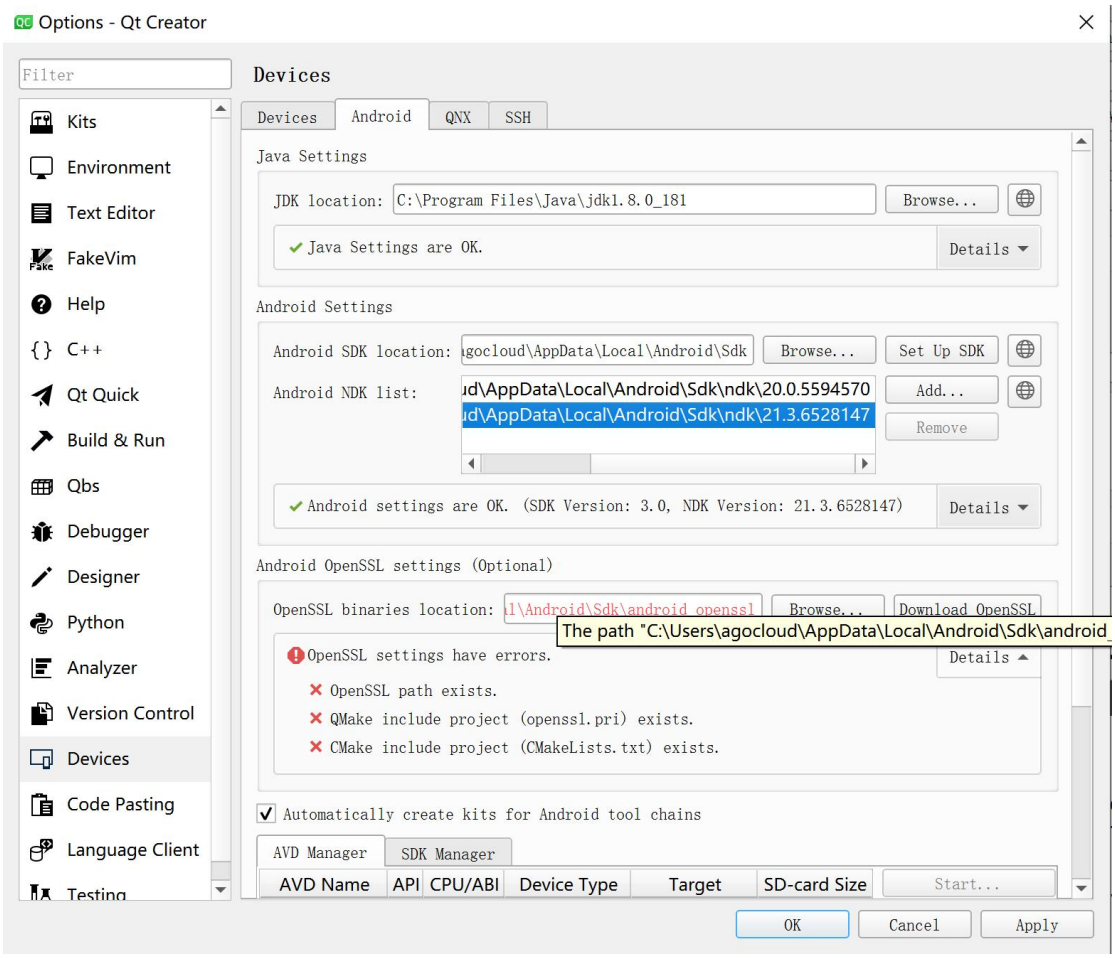
7.1 Set the Android compilation environment

1. Install QT 5.15.0 for Android:

https://download.qt.io/official_releases/online_installers/qt-unified-windows-x86-online.exe

2. Install android studio, configure Android SDK, NDK

3. Start QTCreator, Open "Tools->Options" menuitem, select "devices", set the "JDK Location", "Android SDK Location" and "Android NDK Location", as the following:



4. The Android compile environment setup is complete.

7.2 Configure project

Open *.pro for the first time, and select android for the project

configuration instead of desktop:

Delete *.pro.user in the directory, and then open *.pro, the project configuration window will pop up again.

7.3 Android extra

The android extra code in android/src/org/happytimesoft/gb28181device/GB28181DeviceActivity.java

enableMulticast :

Turn on the multicast reception, android system order to save power, the default does not receive the multicast packets.

disableMulticast :

Turn off the multicast reception.

openFile :

Call the third-party application to open the file

notify :

send android system notification

disableLockScreen :

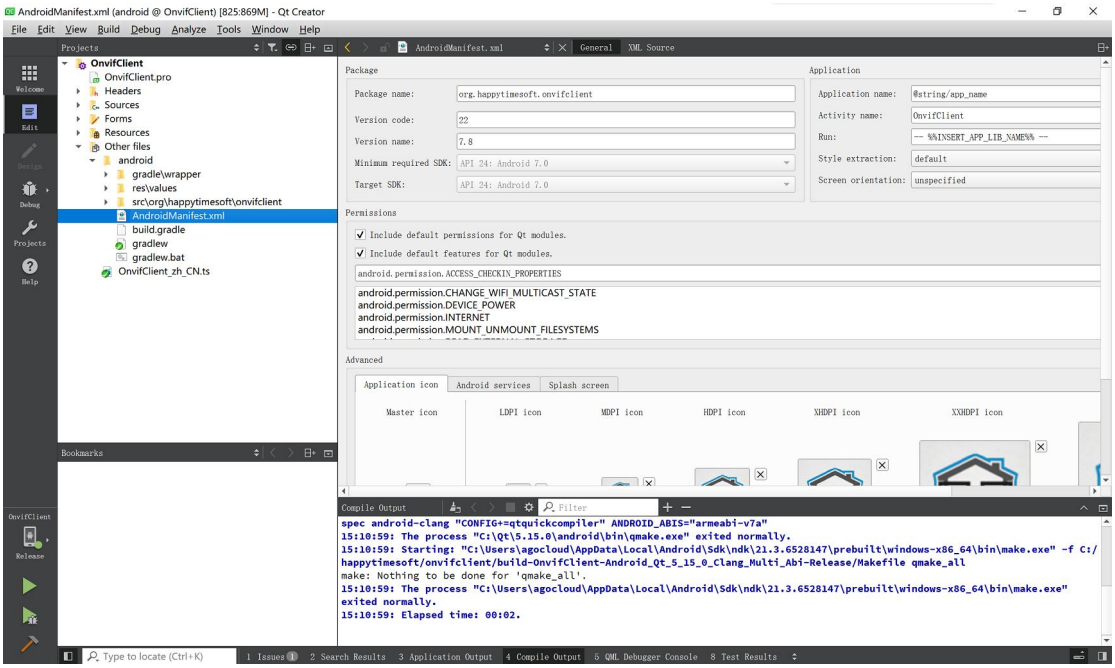
Turn off the automatic lock screen function

enableLockScreen :

Turn on the automatic lock screen function

7.4 Android manifest

Open AndroidManifest.xml in QtCreator, as the following:



You can do the following modifications:

Packet name

Version

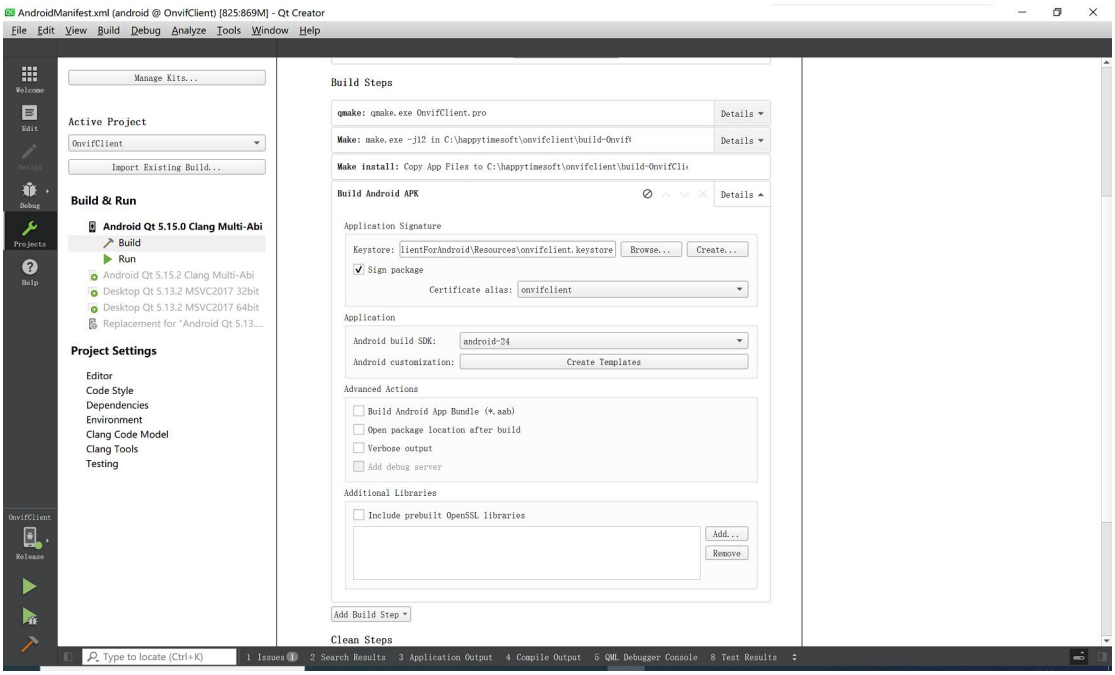
Application name

Application icon

Permissions required

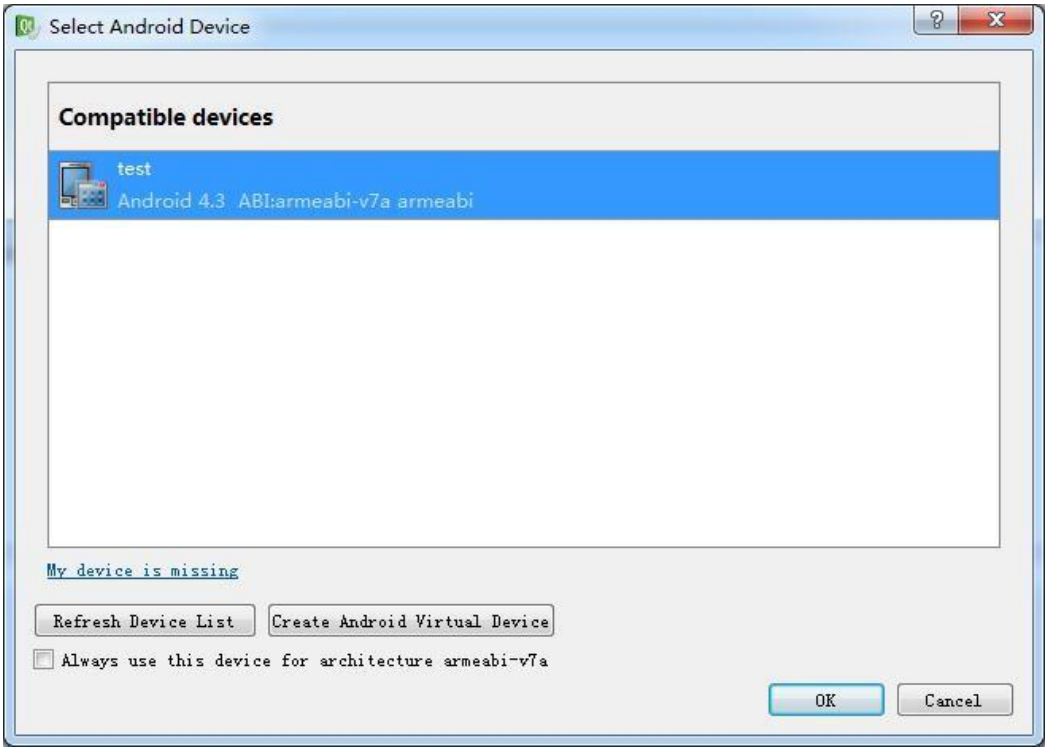
7.5 Build Release APK

Select projects, switch to run configurations page, as the following:



Click “Create...” button to create a sign packet, the check the “Sign packet”.

Click  button, pop up the following dialog:



Select the android device, then click “OK” button.

Note : you can connect your real device or create a virtual device

7.6 Android Studio Project

We provide qt for android project, but not android studio project. However, you can follow the steps below to create an android studio project.

Install Android NDK on Linux and use android-lib.mk to compile the gb28181 device android library.

1. Modify NDK directory and API level in android-lib.mk file.
2. Execute the following command to compile:

make -f android-lib.mk clean

make -f android-lib.mk

To create an android studio project and compile the JNI interface using NDK, you need to use the following files:

android/native.cpp

android/native.h

android/java/com/happytime/lcc/Gb28181Device.java

Then you can call the interfaces in Gb28181Device.java in your Android project.

gb28181Start : start gb28181 device, for parameters, please refer to the user manual about the configuration file.

gb28181Stop: stop gb28181 device.

sendVideoData : Send video data, default is H264 encoding, devid is 0.

Note : I-frames need to be sent together with SPS and PPS.

sendAudioData : Send audio data, default is G711A encoding, 8k sample and mono channel.