Developer manual

（Media Client）

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 media client depend on the following libraries:

FFMPEG version 4.0

Direct X version 10.0 (for video render on windows platform)

Qt Library version 6.5.1

SDL version 2.0.14 (for video render on linux platform)

SRT version 1.5.0

Openssl version 1.1.1g

ZLIB version 1.2.11

ALSA 1.1.6 (capture audio on linux platform for audo back channel)

1.2 Build for Windows

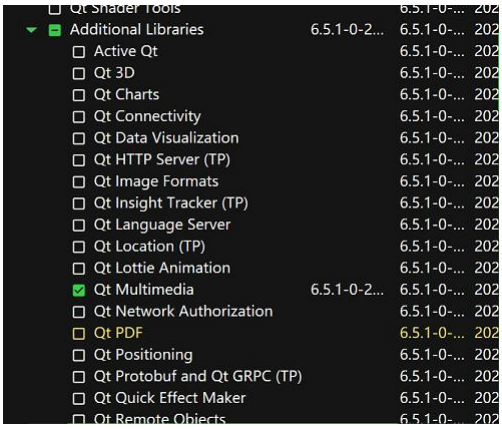
1.2.1 Prepare

Install QT 6.5.1 for windows, install QT VS Add-in for QT6.

https://download.qt.io/official_releases/online_installers/qt-unified-windows-x64-online.exe

https://download.qt.io/archive/vsaddin/2.9.1/qt-vsaddin-msvc2019-2.9.1-rev.06.vsix

When selecting the QT component, you need to select the "QT multimedia" component under "Additional Libraries", as shown in the following figure:



1.2.2 Build

Use VS2019 to open MediaClient.sln to build.

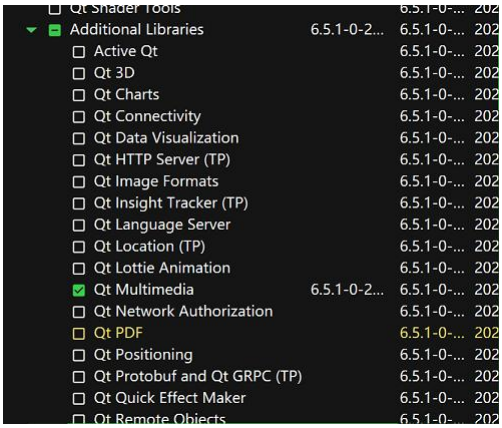
1.3 Build for Linux

1.3.1 Prepare

1. Install QT 6.5.1 for linux.

https://download.qt.io/official_releases/online_installers/qt-unified-linux-x64-online.run

When selecting the QT component, you need to select the "QT multimedia" component under "Additional Libraries", as shown in the following figure:



2. Run the following command in the MediaClient directory to generate dynamic library soft links.

./mklinks.sh

3. If you enable BACKCHANNEL compile option, please reference Build ALSA library on linux platform to Build ALSA library.

1.3.2 Build

Use QtCreator to open MediaClient.pro to compile.

1.4 Build for MAC

1.4.1 Prepare

1. Install QT 6.5.1 for MAC.

https://download.qt.io/official_releases/online_installers/qt-unified-mac-x64-online.dmg

2.Run the following command to install the ffmpeg, x264, x265, and sdl libraries.

brew install ffmpeg

brew install x264

brew install x265

brew install sdl2

brew install openssl

brew install srt

1.4.2 Build

Use QtCreator to open MediaClient.pro to compile.

1.5 Build for IOS

1.5.1 Prepare

1. Install xcode,setting up the IOS development environment
2. Install QT 6.5.1 for IOS.

https://download.qt.io/official_releases/online_installers/qt-unified-mac-x64-online.dmg

1.5.2 Build

Use QtCreator open MediaClientForMobile/MediaClient.pro to build

1.6 Build Android APK

Use QtCreator open MediaClientForMobile/MediaClient.pro to build, please reference Android development

Chapter 2 Build RTSP Client Library

2.1 Windows Platform

Use VS2019 open MediaClient.sln to build

2.2 Linux Platform

make -f RtspClientLibrary.mk clean

make -f RtspClientLibrary.mk (Compile the dynamic library)

make -f RtspClientLibrary-static.mk clean

make -f RtspClientLibrary-static.mk (Compile the static library)

2.3 Android Platform

make-f RtspClientLibrary-android.mk clean

make-f RtspClientLibrary-android.mk

Note : Need to install android build environment

2.4 MAC Platform

make -f RtspClientLibrary-mac.mk clean

make -f RtspClientLibrary-mac.mk

2.5 IOS Platform

make -f RtspClientLibrary-ios.mk clean

make -f RtspClientLibrary-ios.mk

Note : Need to install ios build environment

2.6 Embedded Platform

Modify RtspClientLibrary.mk to specify cross-compiler, then make

Please modify the following options in RtspClientLibrary.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

2.7 Compilation parameter

EPOLL : Enable EPOLL mode for socket multiplexing on Linux

BACKCHANNEL : Support RTSP audio back channel, about rtsp audio back channel, please reference the following link:

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

REPLAY : Support audio and video replay, about audio and video replay, please reference the following link:

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

OVER_HTTP : Support RTSP over HTTP

Note: If the RTSP url starts with http://, RTSP over HTTP connection is used.

OVER_WEBSOCKET : Support RTSP over Websocket

Note: If the RTSP url starts with ws://, RTSP over Websocket connection is used.

If the RTSP url starts with wss://, RTSP over Websocket connection on TLS/SSL (need to enable HTTPS) is used.

HTTPS : Support RTSP over HTTPS

Note: If the RTSP url starts with https://, RTSP over HTTPS connection is used.

Chapter 3 Build RTMP Client Library

3.1 Windows Platform

Use VS2019 open MediaClient.sln to build

3.2 Linux Platform

make -f RtmpClientLibrary.mk clean

make -f RtmpClientLibrary.mk (Compile the dynamic library)

make -f RtmpClientLibrary-static.mk clean

make -f RtmpClientLibrary-static.mk (Compile the static library)

3.3 Android Platform

make-f RtmpClientLibrary-android.mk clean

make-f RtmpClientLibrary-android.mk

Note : Need to install android build environment

3.4 MAC Platform

make -f RtmpClientLibrary-mac.mk clean

make -f RtmpClientLibrary-mac.mk

3.5 IOS Platform

make -f RtmpClientLibrary-ios.mk clean

make -f RtmpClientLibrary-ios.mk

Note : Need to install ios build environment

3.6 Embedded Platform

Modify RtmpClientLibrary.mk to specify cross-compiler, then make

Please modify the following options in RtmpClientLibrary.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

Chapter 4 Build HTTP-FLV Client Library

4.1 Windows Platform

Use VS2019 open MediaClient.sln to build

4.2 Linux Platform

make -f HttpFlvClientLibrary.mk clean

make -f HttpFlvClientLibrary.mk (Compile the dynamic library)

make -f HttpFlvClientLibrary-static.mk clean

make -f HttpFlvClientLibrary-static.mk (Compile the static library)

4.3 Android Platform

make-f HttpFlvClientLibrary-android.mk clean

make-f HttpFlvClientLibrary-android.mk

Note : Need to install android build environment

4.4 MAC Platform

make -f HttpFlvClientLibrary-mac.mk clean

make -f HttpFlvClientLibrary-mac.mk

4.5 IOS Platform

make -f HttpFlvClientLibrary-ios.mk clean

make -f HttpFlvClientLibrary-ios.mk

Note : Need to install ios build environment

4.6 Embedded Platform

Modify HttpFlvClientLibrary.mk to specify cross-compiler, then make

Please modify the following options in HttpFlvClientLibrary.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

Chapter 5 Build HTTP-Mjpeg Client Library

5.1 Windows Platform

Use VS2019 open MediaClient.sln to build

5.2 Linux Platform

make -f HttpMjpegClientLibrary.mk clean

make -f HttpMjpegClientLibrary.mk (Compile the dynamic library)

make -f HttpMjpegClientLibrary-static.mk clean

Make -f HttpMjpegClientLibrary-static.mk (Compile the static library)

5.3 Android Platform

make-f HttpMjpegClientLibrary-android.mk clean

make-f HttpMjpegClientLibrary-android.mk

Note : Need to install android build environment

5.4 MAC Platform

make -f HttpMjpegClientLibrary-mac.mk clean

make -f HttpMjpegClientLibrary-mac.mk

5.5 IOS Platform

make -f HttpMjpegClientLibrary-ios.mk clean

make -f HttpMjpegClientLibrary-ios.mk

Note : Need to install ios build environment

5.6 Embedded Platform

Modify HttpMjpegClientLibrary.mk to specify cross-compiler, then make

Please modify the following options in HttpMjpegClientLibrary.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

Chapter 6 Build SRT Client Library

6.1 Windows Platform

Use VS2019 open MediaClient.sln to build

6.2 Linux Platform

make -f SrtClientLibrary.mk clean

make -f SrtClientLibrary.mk (Compile the dynamic library)

make -f SrtClientLibrary-static.mk clean

make -f SrtClientLibrary-static.mk (Compile the static library)

6.3 Android Platform

make -f SrtClientLibrary-android.mk clean

make -f SrtClientLibrary-android.mk

Note : Need to install android build environment

6.4 MAC Platform

make -f SrtClientLibrary-mac.mk clean

make -f SrtClientLibrary-mac.mk

6.5 IOS Platform

make -f SrtClientLibrary-ios.mk clean

make -f SrtClientLibrary-ios.mk

Note : Need to install ios build environment

6.6 Embedded Platform

Modify SrtClientLibrary.mk to specify cross-compiler, then make

Please modify the following options in SrtClientLibrary.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

Chapter 7 Build ALSA library on linux platform

If you enable BACKCHANNEL 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 8 Build openssl

For windows, Linux, and mac platforms, please refer to the openssl install document.

Build openssl for Android, please refer the openssl-android-build.txt.

Build openssl for IOS, please use openssl-ios-build.sh.

The source code openssl/lib/android folder include the android openssl library is arm version, The source code openssl/lib/ios folder include the IOS openssl library is arm64 version.

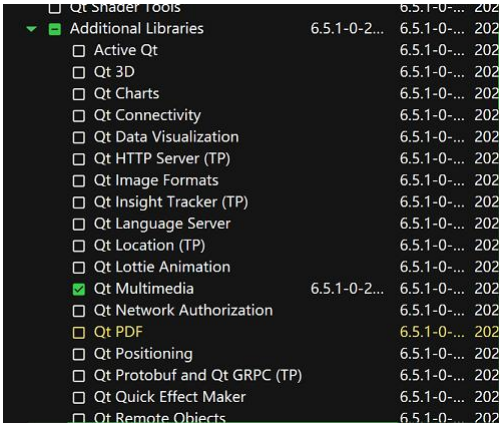
If you need to compile other openssl versions, according to the compilation instructions to compile.

Chapter 9 Setup QT for VS

9.1 Install QT

Download and run the QT install application, select the QT for VS2019 64 bit, then finish the installation.

When selecting the QT component, you need to select the "QT multimedia" component under "Additional Libraries", as shown in the following figure:

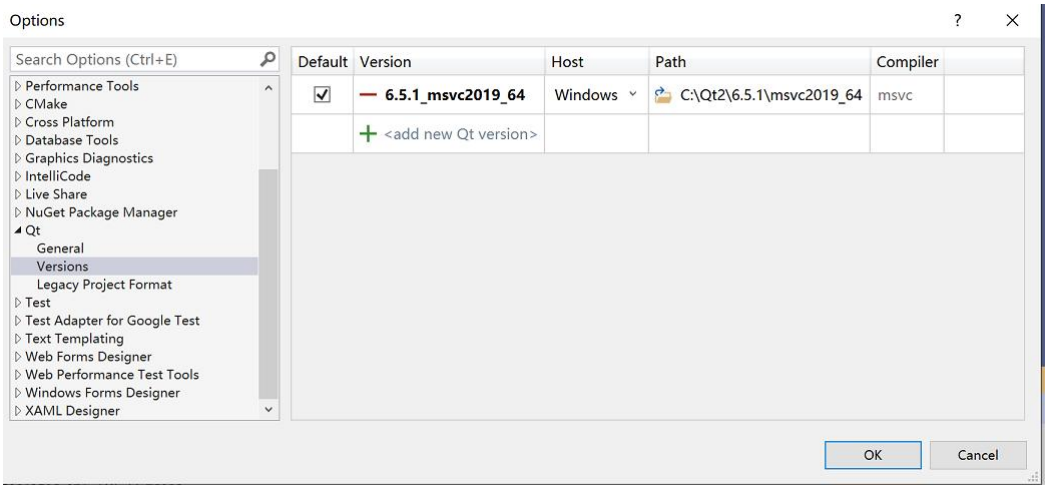


9.2 Install QT VS addin

Download and run qt-vsaddin-msvc2019, install the QT VS addin for VS2019.

9.3 Set the QT environment for VS

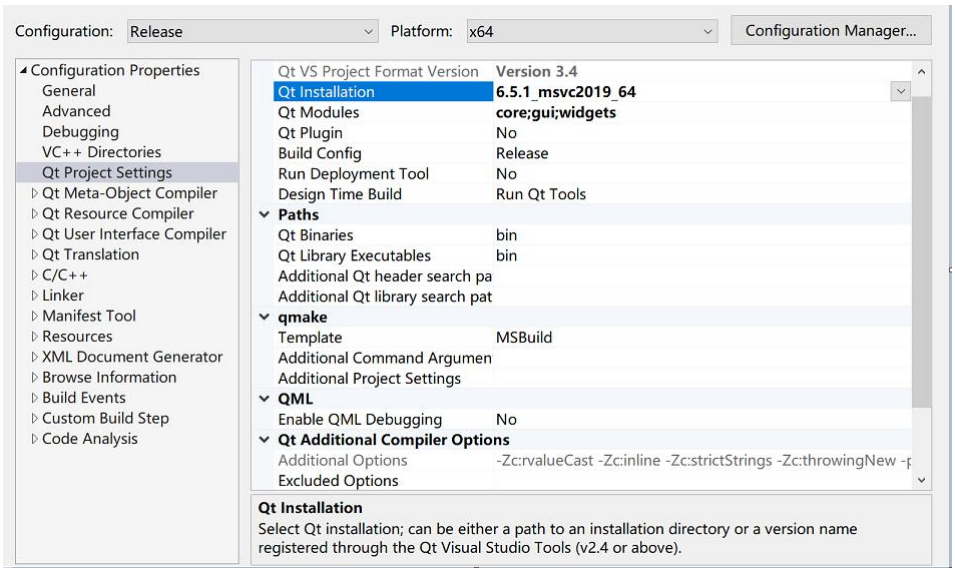
Run VS2019, select the "Extensions->Qt VS TOOL->QT Version" menu item, as the following:



Click “+” to add new QT Version.

9.4 Set QT for MediaClient project

Open MediaClient VS2019 project(MediaClient.sln), select the MediaClient project, select “Extensions->Qt VS TOOL->QT Project Settings”, as the following:



Chapter 10 Dll files

Please download the demo version from :

https://happytimesoft.com/downloads/happytime-media-client-x64-setup.exe

Install it, then copy all the required files to you bin folder from the install folder.

Do not copy the QT dynamic library, please copy the QT dynamic library files from your QT installation path.

Alternatively, you can run QT tool windeployqt6 to deploy the qt dynamic library to your bin folder.

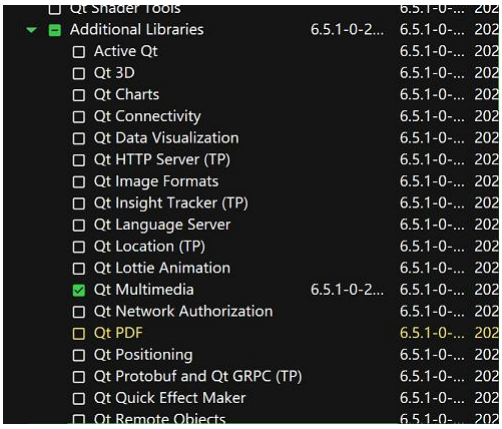
Chapter 11 Android development

11.1 Set the Android compilation environment

1. Install QT 6.5.1 for Android:

https://download.qt.io/official_releases/online_installers/qt-unified-windows-x64-online.exe

When selecting the QT component, you need to select the "QT multimedia" component under "Additional Libraries", as shown in the following figure:



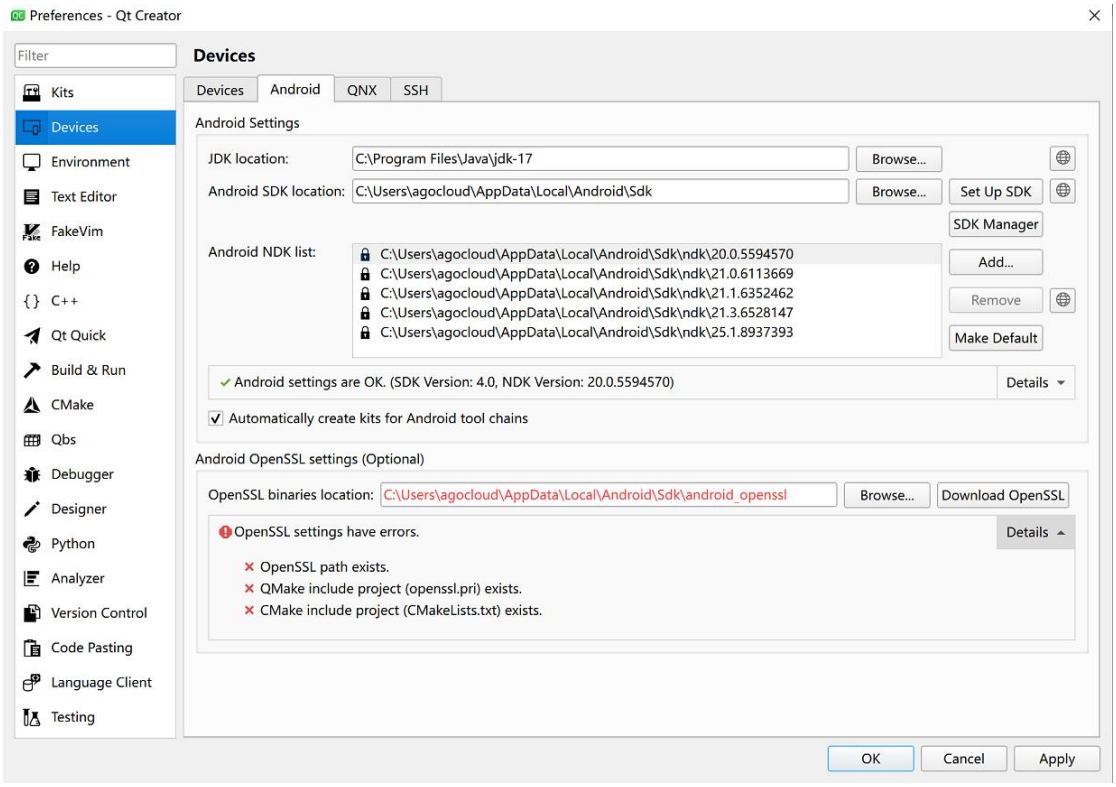
2. Install android studio, configure Android SDK, NDK

The Android app has the following requirements:

JDK 17

Android platform SDK 33

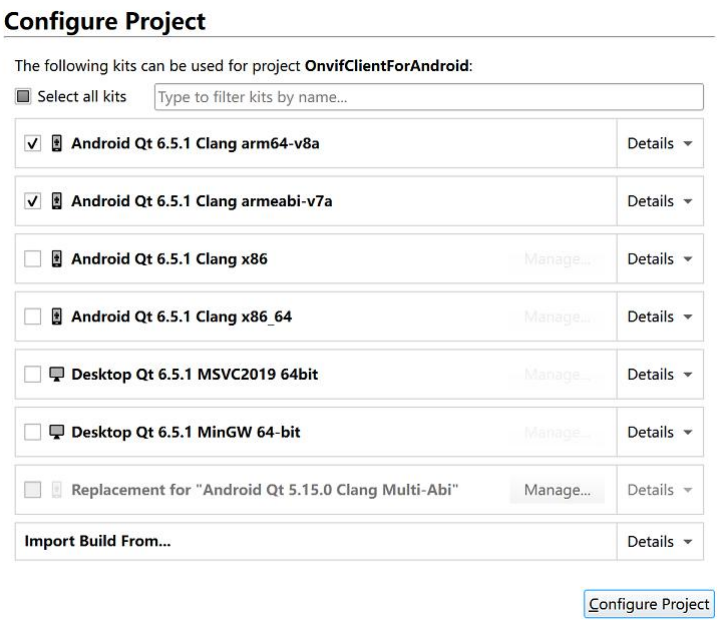
3. Start QTCreator, Open "Edit->Preferences" menuitem, select "devices", set the "JDK Location", "Android SDK Location" and "Android NDK Location", as the following:



4. The Android compile environment setup is complete.

11.2 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.

11.3 Android extra

The android extra code in

android/src/org/happytimesoft/util/HtUtil.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

disableLockScreen :

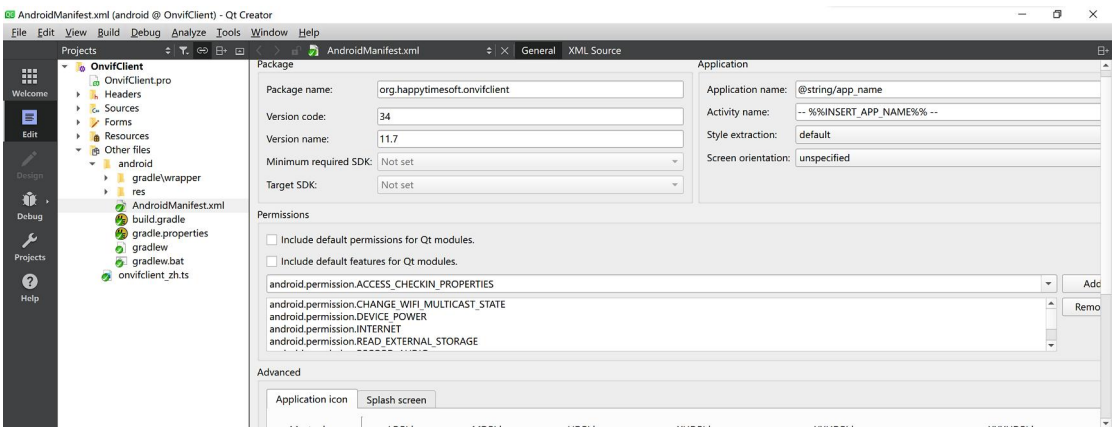
Turn off the automatic lock screen function

enableLockScreen :

Turn on the automatic lock screen function

11.4 Android manifest

Open AndroidManifest.xml in QtCreator, as the following:



You can do the following modifications:

Packet name

Version

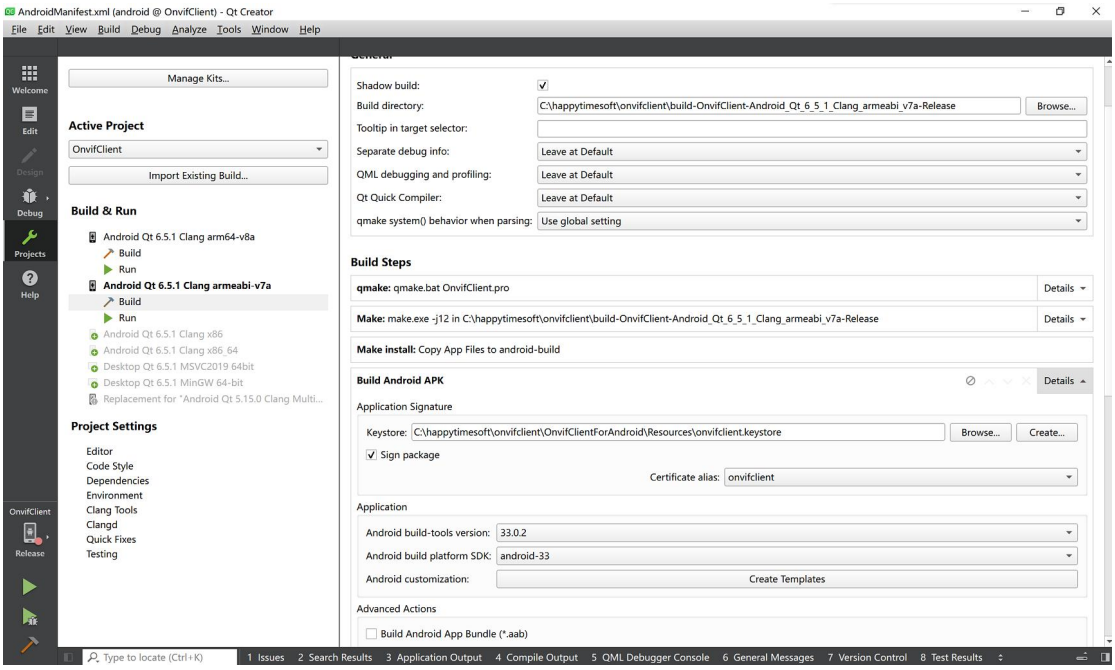
Application name

Application icon

Permissions required

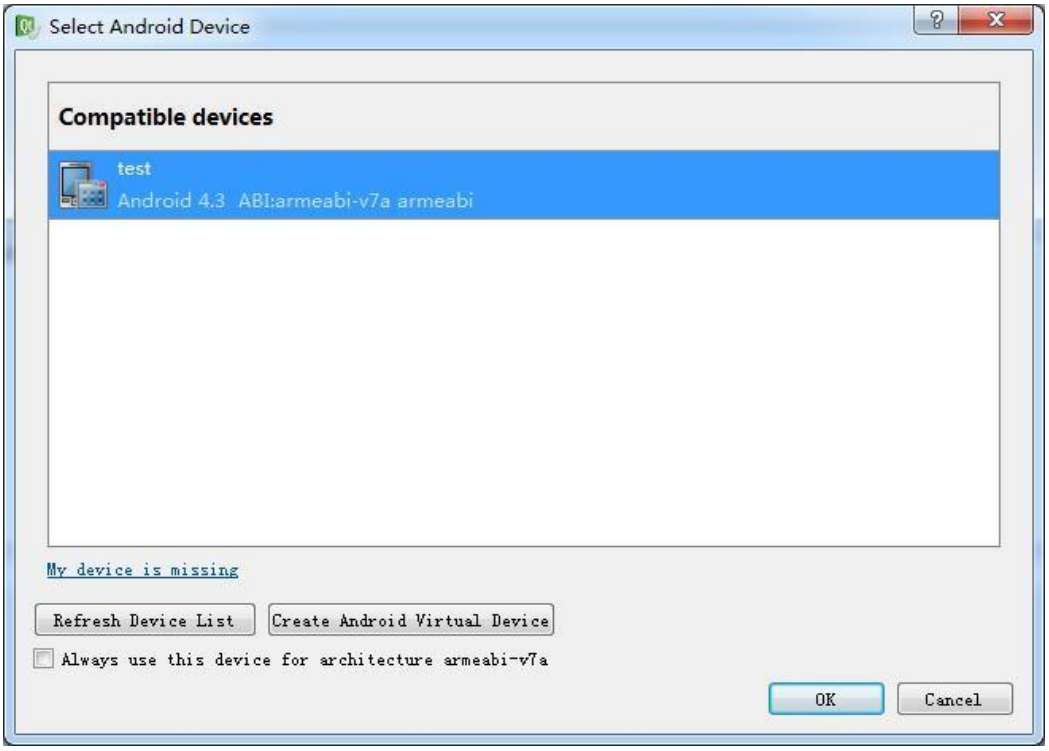
11.5 Build Release APK

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