# VCodecImsdk C++ library

**v4.0.1**

# Table of contents



# Overview

**VCodecImsdk** C++ library provides hardware video **encoding/decoding** (H264, HEVC and JPEG) for Intel HD Graphics based on Intel Media SDK. VCodecImsdk class inherits interface and data structures from open source VCodec library. The library provides simple interface to be implemented in different C++ projects. It is written with C++11 standard. The library is supplied as source code only in form of CMake project.

**Encoding** time for 11th Gen Intel(R) Core(TM) **i5-1145G7E** on **Ubuntu 22.04 LTS**:

| codec / resolution | 2560x1440 | 1920x1080 | 1280x720 | 640x512 |
|---|---|---|---|---|
| **H264** | 11.6 msec | 8.6 msec | 4.4 msec | 2.6 msec |
| **HEVC** | 23.4 msec | 15.2 msec | 9.3 msec | 5.2 msec |
| **JPEG** | 8.2 msec | 4.8 msec | 2.5 msec | 1.2 msec |

**Decoding** time for 11th Gen Intel(R) Core(TM) **i5-1145G7E** on **Ubuntu 22.04 LTS**:

| codec / resolution | 2560x1440 | 1920x1080 | 1280x720 | 640x512 |
|---|---|---|---|---|
| **H264** | 7.3 msec | 4.8 msec | 2.5 msec | 1.3 msec |
| **HEVC** | 7.4 msec | 4.3 msec | 2.2 msec | 1.3 msec |
| **JPEG** | 8.7 msec | 5.2 msec | 2.6 msec | 1.3 msec |

# Versions

**Table 1** - Library versions.

| Version | Release date | What's new |
|---|---|---|
| 1.0.1 | 19.05.2022 | First version. |
| 2.0.0 | 27.06.2022 | - new interface |
| 3.0.0 | 27.01.2023 | - Added JPEG encoding support.<br>- Added test application . |
| 4.0.0 | 15.09.2023 | - Interface changes to VCodec.<br>- Test application updated.<br>- Added example application. |
| 4.0.1 | 10.01.2024 | - VCodec interface library updated.<br>- Examples updated.<br>- Documentation updated. |

# Library files

The library is supplied only by source code. The user is given a set of files in the form of a CMake project (repository). The repository structure is shown below:

```
CMakeLists.txt --------------- Main CMake file of the library.
3rdparty --------------------- Folder with third-party libraries.
    CMakeLists.txt ----------- CMake file which includes third-party libraries.
    IMSDK_LINUX -------------- Implimentation of IMSDK on Linux.
    IMSDK_WIN ---------------- Implimentation of IMSDK on Windows.
    VCodec ------------------- Source code of VCodec library.
example ---------------------- Folder with simple example of VCodecImsdk usage.
    CMakeLists.txt ----------- CMake file for example application.
    main.cpp ----------------- Source code file of example application.
test ------------------------- Folder with codec test application.
    CMakeLists.txt ----------- CMake file for codec test application.
    main.cpp ----------------- Source code file of codec test application.
src -------------------------- Folder with source code of the library.
    Backend ------------------ Folder with bakend files.
    MediaSamples ------------- Folder with service samples.
    CMakeLists.txt ----------- CMake file of the library.
```

```
    VCodecImsdk.cpp ---------- Source code file of the library.
    VCodecImsdk.h ------------ Header file which includes VCodecImsdk class declaration.
    VCodecImsdkVersion.h ----- Header file which includes version of the library.
    VCodecImsdkVersion.h.in -- CMake service file to generate version file.
```

# VCodecImsdk class description

## VCodecImsdk class declaration

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

```
class VCodecImsdk : public VCodec
{
public:

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

    /// Class constructor.
    VCodecImsdk();

    /// Class destructor.
    ~VCodecImsdk();

    /// Set parameter value.
    bool setParam(VCodecParam id, float value) override;

    /// Get parameter value.
    float getParam(VCodecParam id) override;

    /// Encode/Decode video frame.
    bool transcode(Frame& src, Frame& dst) override;

    /// Execute command.
    bool executeCommand(VCodecCommand id) override;
};
```

## getVersion method

**getVersion()** method returns string of current version of **VCodecImsdk** class. Method declaration:

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

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

```
cout << "VCodecImsdk class version: " << VCodecImsdk::getVersion() << endl;
```

Console output:

```
VCodecImsdk class version: 4.0.1
```

# transcode method

**transcode(...)** method intended to encode and decode video frame (Frame class). Video codec encodes/decodes video frames frame-by-frame. Method declaration:

```
bool transcode(Frame& src, Frame& dst) override;
```

| Parameter | Value |
|---|---|
| src | Source video frame (see Frame class description). To encode video **src** frame must have raw pixel format such as **NV12**. To decode video data **src** frame must have compressed pixel format (field **fourcc** of Frame class): **HEVC**, **JPEG**, **H264**. |
| dst | Result video frame (see Frame class description). To encode video data **dst** frame must have compressed pixel format (field **fourcc** of Frame class): **HEVC**, **JPEG**, **H264**. In contrary, src frame must be **NV12**. |

**Returns:** TRUE if frame was encoded/decoded or FALSE if not.

# setParam method

**setParam(...)** method designed to set new video codec parameters value. Method declaration:

```
bool setParam(VCodecParam id, float value) override;
```

| Parameter | Description |
|---|---|
| id | Video codec parameter ID according to **VCodecParam** enum. |
| value | Video codec parameter value. |

**Returns:** TRUE is the parameter was set or FALSE if not.

**VCodec.h** file of VCodec library defines IDs for parameters (**VCodecParam** enum) and IDs for commands (**VCodecCommand** enum). VCodecParam declaration:

```
enum class VCodecParam
{
    /// [read/write] Log level:
    /// 0-Disable, 1-Console, 2-File, 3-Console and file.
    LOG_LEVEL = 1,
    /// [read/write] Bitrate, kbps. For H264 and H265 codecs.
    BITRATE_KBPS,
    /// [read/write] Quality 0-100%. For JPEG codecs.
    QUALITY,
```

```
    /// [read/write] FPS. For H264 and H265 codecs.
    FPS,
    /// [read/write] GOP size. For H264 and H265 codecs.
    GOP,
    /// [read/write] H264 profile: 0 - Baseline, 1 - Main, 2 - High.
    H264_PROFILE,
    /// [read/write] Codec type. Depends on implementation.
    TYPE,
    /// Custom 1. Depends on implementation.
    CUSTOM_1,
    /// Custom 2. Depends on implementation.
    CUSTOM_2,
    /// Custom 3. Depends on implementation.
    CUSTOM_3
};
```

**Table 2** - Video codec params description. Some params not supported by VCodecOneVpl library.

| Parameter | Access | Description |
|---|---|---|
| LOG_LEVEL | read / write | **Not supported** by VCodecImsdk library. |
| BITRATE_KBPS | read / write | Bitrate, kbps. For H264 encoding only. According to this value, FPS and GOP size video codec calculate parameter for H264 encoding. |
| QUALITY | read / write | **Not supported** by VCodecImsdk library. |
| FPS | read / write | FPS. For H264 encoding only. According to this value, FPS and GOP size video codec calculate parameter for H264 encoding. |
| GOP | read / write | GOP size (Period of key frames) for H264 encoding. Value: 1 - each output frame is key frame, 20 - each 20th frame is key frame etc. |
| H264_PROFILE | read / write | H264 profile for H264 encoding: 0 - Baseline, 1 - Main, 2 - High. |
| TYPE | read / write | **Not supported** by VCodecImsdk library. |
| CUSTOM_1 | read / write | **Not supported** by VCodecImsdk library. |
| CUSTOM_2 | read / write | **Not supported** by VCodecImsdk library. |
| CUSTOM_3 | read / write | **Not supported** by VCodecImsdk library. |

# getParam method

**getParam(...)** method designed to obtain video codec parameter value. Method declaration:

```
float getParam(VCodecParam id);
```

| Parameter | Description |
|---|---|
| id | Video codec parameter ID according to **VCodecParam** enum (see Table 2). |

**Returns:** parameter value or -1 if the parameters not supported.

# executeCommand method

**executeCommand(...)** method designed to execute video codec command. Version 4.0.1 doesn't support commands. Method will return FALSE. Method declaration:

```
bool executeCommand(VCodecCommand id);
```

| Parameter | Description |
|---|---|
| id | Video codec command ID according to **VCodecCommand** enum. |

**Returns:** method returns FALSE in any case.

**VCodec.h** file of VCodec library defines IDs for parameters (**VCodecParam** enum) and IDs for commands (**VCodecCommand** enum). VCodecCommand declaration:

```
enum class VCodecCommand
{
    /// Reset.
    RESET = 1,
    /// Generate key frame. For H264 and H265 codecs.
    MAKE_KEY_FRAME
};
```

**Table 3** - Video codec commands description. Some commands maybe unsupported by particular video codec class.

| Command | Description |
|---|---|
| RESET | **Not supported** by VCodecImsdk library. |
| MAKE_KEY_FRAME | **Not supported** by VCodecImsdk library. |

# Build and connect to your project

If you are using Linux you have to install libraries according to Installation on Linux. For Windows you don't need install additional libraries. Typical commands to build **VCodecImsdk** library:

```
cd VCodecImsdk
git submodule update --init --recursive
mkdir build
cd build
cmake ..
make
```

If you want connect **VCodecImsdk** 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
```

Create folder **3rdparty** in your repository. Copy repository folder **VCodecImsdk** to **3rdparty** folder. New structure of your repository:

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

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(${PARENT}_SUBMODULE_VCODEC_IMSDK                    ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_VCODEC_IMSDK)
    SET(${PARENT}_VCODEC_IMSDK                          ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_IMSDK_TEST                     OFF CACHE BOOL "" FORCE)
    SET(${PARENT}_VCODEC_IMSDK_EXAMPLE                  OFF CACHE BOOL "" FORCE)
endif()

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

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

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

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 VCodecImsdk library in your **src/CMakeLists.txt** file:

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

Done!

# Installation on Linux

There are several steps to launching VCodecImsdk on Linux (tested on Ubuntu 22.04 LTS):

1. Install ubuntu with last updates:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

2. Install additional dependencies:

```
sudo apt-get install wayland-protocols libx11-xcb-dev libxcb-present-dev libxcb-dri3-dev ffmpeg build-essential cmake git libopencv-dev vainfo v4l-utils net-tools openssh-client openssh-server ninja-build libmfx1 libmfx-tools libva-drm2 libva-x11-2 libva-wayland2 libva-glx2 libva-dev libmfx-dev autoconf libtool libdrm-dev xorg xorg-dev openbox libx11-dev libgl1-mesa-glx libgl1-mesa-dev
```

3. Install gmmlib:

```
cd Downloads
git clone https://github.com/intel/gmmlib.git
cd gmmlib
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j4
sudo make install
```

4. Install intel media driver

```
cd Downloads
git clone https://github.com/intel/media-driver.git
cd media-driver
nano CMakeLists.txt
Replace default CMAKE_INSTALL_PREFIX to "/usr/lib/x86_64-linux-gnu" (insert line "set(CMAKE_INSTALL_PREFIX "/usr/lib/x86_64-linux-gnu")" on 37 line )
Save changes: ctrl + S and then ctrl + X
cd ..
mkdir build_media
cd build_media
cmake -DCMAKE_BUILD_TYPE=Release ../media-driver
make -j4
sudo make install
```

5. Change driver according CPU version. **Additional information**

```
export LIBVA_DRIVER_NAME=iHD
sudo apt-get install intel-media-va-driver-non-free
```

# Simple example

Example application generates image color pattern with moving rectangle and writes compressed data to binary file **"out.hevc"**. Example shows how to create codec objects and how to encode video frames:

```
#include <iostream>
#include "VCodecImsdk.h"

int main(void)
{
    // Create codec and set parameters.
```

```
cr::video::VCodec* videoCodec = new cr::video::VCodecImsdk();
videoCodec->setParam(cr::video::VCodecParam::BITRATE_KBPS, 7500);
videoCodec->setParam(cr::video::VCodecParam::GOP, 30);
videoCodec->setParam(cr::video::VCodecParam::FPS, 30);

// Create NV12 frame and fill by random values.
const int width = 1280;
const int height = 720;
cr::video::Frame frameNv12(width, height, cr::video::Fourcc::NV12);
for (uint32_t i = 0; i < frameNv12.size; ++i)
    frameNv12.data[i] = (uint8_t)i;

// Create output HEVC frame.
cr::video::Frame frameHEVC(width, height, cr::video::Fourcc::HEVC);

// Create output file.
FILE *outputFile = fopen("out.hevc", "w+b");

// Params for moving object.
int objectWidth = 128;
int objectHeight = 128;
int directionX = 1;
int directionY = 1;
int objectX = width / 4;
int objectY = height / 2;

// Encode and record 200 frames.
for (uint32_t n = 0; n < 200; ++n)
{
    // Draw moving object.
    memset(frameNv12.data, 128, width * height);
    for (int y = objectY; y < objectY + objectHeight; ++y)
        for (int x = objectX; x < objectX + objectHeight; ++x)
            frameNv12.data[y * width + x] = 255;
    objectX += directionX;
    objectY += directionY;
    if (objectX >= width - objectWidth - 5 || objectX <= objectWidth + 5)
        directionX = -directionX;
    if (objectY >= height - objectHeight - 5 || objectY <= objectHeight + 5)
        directionY = -directionY;

    // Encode.
    if (!videoCodec->transcode(frameNv12, frameHEVC))
    {
        std::cout << "Can't encode frame" << std::endl;
        continue;
    }

    // Write to file.
    fwrite(frameHEVC.data, frameHEVC.size, 1, outputFile);
}

// Close file.
fclose(outputFile);
```

```
    return 1;
}
```