
# Eyecloud OpenNCC
# Software Development Kit (SDK) API

**Specification**

---

**June 2021**

**Revision 3.0.1**

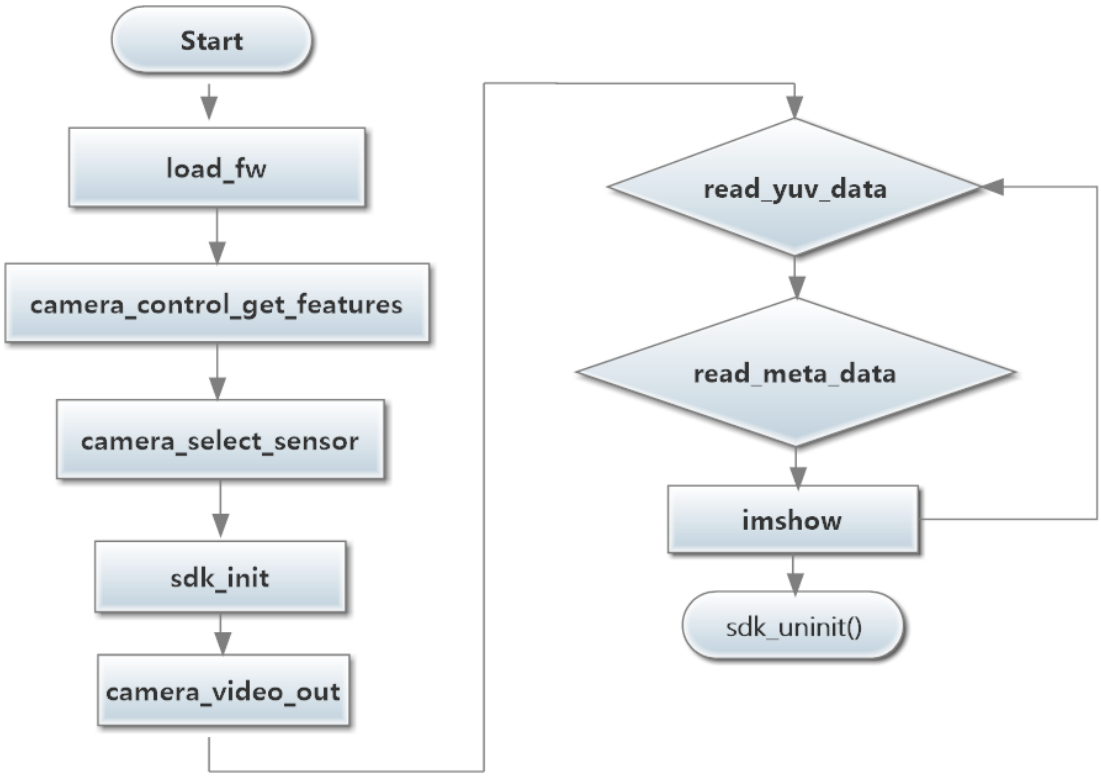
# Revision

| Version | Date | Editor | Description |
|---|---|---|---|
| 1.0.0 | 2020/1/10 | Joshua | Initial version |
| 1.0.1 | 2020/3/16 | Yang.W | Optimized version |
| 2.0.0 | 2020/4/7 | Duke | Add python interface |
| 3.0.0 | 2020/10/14 | Duke | 1:Add interface suppored for dual model sdk_net2_init()and related structures.<br>2: Add 64 bytes in metadata<br>3:Remove interface of infrared and depth data . |
| 3.0.1 | 2021/6/30 | Zed | Optimize document structure |

# Directory

# 1 C/C++ SDK Interface Description

The interface files are primarily contained in 3 files: **sdk.h**, **cameraCtrl.h**, and **Fp16Convert**.




**OpenNCC SDK Video Processing Flowchart**

## 1.1 Device Initialization Related Interfaces

### 1.1.1 Load Device Firmware

| Interface Name | Interface Parameters | Description |
|---|---|---|
| load_fw() | const char* bootExe | Path to USB boot program |
| | const char* firmware | Path to firmware file |

**Example**:

- load_fw("./moviUsbBoot","./fw/flicRefApp.mvcmd");

Return:

0: Success; -1: Failure

**Description**:

Automatically loads device firmware, device boots, host (PC) opens USB device.

### 1.1.2 Get Connected USB Information

| Interface Name | Interface Parameters | Description |
|---|---|---|
| get_usb_version() | void | N/A |

**Example**:

- get_usb_version();

Return:30:usb3.0; 20:usb2.0

**Description**:

Returns USB Version Information (Port and USB Cable) connected to the device.

### 1.1.3 Initializing AI Camera Parameters

| Interface Name | Interface Parameters | Description |
|---|---|---|
| sdk_init() | vscRecvCb cb | Callback |
| | void* param | Callback function parameters |
| | const char *blob_path | Path to AI Model (.blob) |
| | CameraInfo*cam | Camera Configuration Parameters (See below) |
| | int cam_Len | Camera Configuration Structure Length |

There are two ways to get media and metadata.

1: Passively obtained through callback function.

2: Actively obtained through read_XXX_data() without setting the callback function and callback parameters.

```
typedef struct{
    int imageWidth;          //image width
    int imageHeight;         //image height
    int startX;              //Ai calculation start X
    int startY;              //Ai calculation start Y
    int endX;                //Ai calculation end X
    int endY;                //Ai calculation end Y
```

```
    int inputDimWidth;
  /* input width after resize,if <=0,get from xml automatically */
    int inputDimHeight;
/* input height after resize,if <=0,get from xml automatically */
    IMAGE_FORMAT inputFormat;
/*model input fomat,only support RGB/RGB_PLANAR/BGR/BGR_PLANAR */
    float meanValue[3];
/* Second pretreatment; If inputFormat is RGB:
                        R = (R-meanValue[0])/stdValue
                        G = (G-meanValue[0])/stdValue
                        B = (B-meanValue[0])/stdValue */
    float   stdValue;
    int     isOutputYUV;            //enable control 1 : open   0 :close
    int     isOutputH26X;           //enable control 1 : open   0 :close
    int     isOutputJPEG;           // enable control 1 : open   0 :close
    encodeMode mode;            /* H264/H265 */
} CameraInfo;
```

**Example**:

- sdk_init(NULL,  NULL,
  (char*) "./blob/face-detection.blob",
  &cam_info,
  sizeof(cam_info));

Return:

0: Success; -1: Failure

**Description**:

Specifies the AI Vision model file and calculation parameters, initializes the device algorithm model, camera function switch selection, sets the video encoding parameters (if the function switch is turned on). Video output is controlled by camera_video_out()

Attention:This interface only supports one AI model with single-input.

### 1.1.4 Initializing AI Camera 2 modes Parameters

| Interface Name | Interface Parameters | Description |
|---|---|---|

| | | |
|---|---|---|
| sdk_net2_init() | vscRecvCb cb | Callback |
| | void* param | Callback function parameters |
| | const char *blob_path | AI Mode 1 file(blob) |
| | Network1Par* par1 | Mode1 param |
| | int par1_Len | Param1 length |
| | const char *blob2_path | AI Mode 2 file(blob) |
| | Network2Par* par2 | Mode2 param |
| | int par2_Len | Param2 length |

There are two ways to get media and metadata.

1. Passively obtained through callback function.

2. Actively obtained through read_XXX_data() without setting the callback function and callback parameters.

```
/* first module process setting */
typedef struct{
    int imageWidth;
    int imageHeight;
    int startX;
    int startY;
    int endX;
    int endY;
    int inputDimWidth;
    int inputDimHeight;
    IMAGE_FORMAT inputFormat;
    /* input image mode,only RGB/RGB_PLANAR/BGR/BGR_PLANAR */
    float meanValue[3];          /* inputFormat RGB:
                                    R = (R-meanValue[0])/stdValue
                                    G = (G-meanValue[0])/stdValue
                                    B = (B-meanValue[0])/stdValue */
    float stdValue;
    int     isOutputYUV;
    int     isOutputH26X;
    int     isOutputJPEG;
    encodeMode mode;                /* H264/H265 */

  char extInputs[MAX_EXTINPUT_SIZE];          /* second model input */
```

```
    int     modelCascade;                   /* linked next model */
    int     inferenceACC;                   /*  Accelerating    inference  0:close
1:open */
} Network1Par;


/* second module param */
typedef struct{
        int startXAdj;
        int startYAdj;
        int endXAdj;
        int endYAdj;
        char    labelMask[MAX_LABEL_SIZE];
        /* mask label, bit equal to 1    will be useful */
        float minConf;                  /* conf value from first model */
        int inputDimWidth;
        int inputDimHeight;
        IMAGE_FORMAT inputFormat;
        /* input image mode,only RGB/RGB_PLANAR/BGR/BGR_PLANAR */
        float meanValue[3];
        /* inputFormat RGB:
            R = (R-meanValue[0])/stdValue
            G = (G-meanValue[0])/stdValue
            B = (B-meanValue[0])/stdValue */
        float stdValue;
        char extInputs[MAX_EXTINPUT_SIZE];      /* second model input    */
        int     modelCascade;           /*linked next model    for third model in future*/
} Network2Par;
```

**Example**:

- char *blob    = "./blob/vehicle-license-plate-detection-barrier-0106/vehicle-license-plate-detection-barrier-0106.blob";
- char *blob2 = "./blob/license-plate-recognition-barrier-0001/license-plate-recognition-barrier-0001.blob";

- ret = sdk_net2_init(0,0,\
                blob,    &cnn1PrmSet, sizeof(cnn1PrmSet), \
                blob2,    &cnn2PrmSet, sizeof(cnn2PrmSet));

Return:

0: Success; -1: Failure

**Description**:

Specifies the 2 AI Vision model files and calculation parameters, initializes the device algorithm model, camera function switch selection, sets the video encoding parameters (if the function switch is turned on). Video output is controlled by camera_video_out().

### 1.1.5 Get Metadata Size

| Interface Name | Interface Parameters | Description |
|---|---|---|
| get_meta_size() | void | N/A |

**Example**: Omitted.

Return: Size of the CNN calculation result's metadata

**Description**:

Turn off the camera, reload the model, and call before changing the model.it only support one AI mode net now.

### 1.1.6 Uninitialize SDK

| Interface Name | Interface Parameters | Description |
|---|---|---|
| sdk_uninit() | void | N/A |

**Example**:

- sdk_uninit();

Return: N/A

**Description**:

Turn off the camera, reload the model, and call before changing the model.

### 1.1.7 Get SDK Version Information

| Interface Name | Interface Parameters | Description |
|---|---|---|
| get_sdk_version() | char* version | Version Information |

**Example**:

- char version[100];

- get_sdk_version(version);

Return:void

**Description**:

Gets SDK version information.

## 1.2 Video Streaming Related Interfaces

### 1.2.1 Get YUV Data

| Interface Name | Interface Parameters | Description |
|---|---|---|
| read_yuv_data() | char* pbuf | Receive Buffer |
| | int * size | Input and output parameters. Input is the size of the input buffer , output is the size of the returned video data. |
| | int blocked | 0: If there is no data, return immediately. |

**Example**:

- read_yuv_data(data_yuv, &size,1)

Return:

0: Success; -1: Failure

**Description**:

Gets a YUV data stream from the device.

Content: struct frameSpecOut + YUV(NV12)

### 1.2.2 Get H.264 or H.265 Data

| Interface Name | Interface Parameters | Description |
|---|---|---|
| read_26x_data() | char* pbuf | Receive Buffer |
| | int * size | Input and output parameters. Input is the size of the input buffer , output is the size of the returned video data. |
| | int blocked | 0: If there is no data, return immediately. |

**Example**:

- read_26x_data(data_26x, &size,1)

Return:

0: Success; -1: Failure

**Description**:

Gets a H.264 or H.265 data stream from the device.

Content: struct frameSpecOut + H26X data.

### 1.2.3 Get JPEG data

| Interface Name | Interface Parameters | Description |
|---|---|---|
| read_jpg_data() | char* pbuf | Receive Buffer |
| | int * size | Input and output parameters. Input is the size of the input buffer area, output is the size of the returned video data. |
| | int blocked | 0: If there is no data, return immediately. |

**Example**:

- read_jpg_data(yuv420p, &size,1)

Return:

0: Success; -1: Failure

**Description**:

Gets a JPEG data stream from the device.

Content: struct frameSpecOut + MJPEG data.

### 1.2.4 Get the output of the AI Network algorithm

| Interface Name | Interface Parameters | Description |
|---|---|---|
| read_meta_data() | char* pbuf | Receive Buffer |
| | int * size | Input and output parameters. Input is the size of the input buffer area, output is the size of the returned video data. |
| | int blocked | 0: If there is no data, return immediately. |

**Example**:

- read_meta_data(data_mate,&size,1)

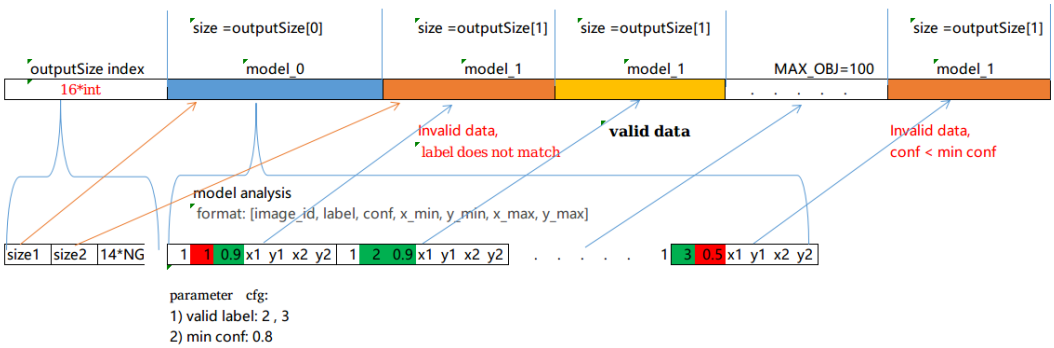
Return:

0:Success; -1: Failure

**Description**:

Get the number of operations from the device's AI Network

.Content: struct frameSpecOut + AI data. and AI data format is below::




## 1.3 Camera Control Related Interfaces

### 1.3.1 Obtain Camera Module Information

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_get_features() | SensorModesConfig * | Device information structure pointer |

**Example**:

- SensorModesConfig cameraCfg;
- camera_control_get_features(&cameraCfg);

Return:

0: Success; -1: Failure

cameraCfg.moduleName
cameraCfg.camWidth //image width
cameraCfg.camHeight //image height
cameraCfg.camFps //frame rate
cameraCfg.AFmode //focus mode
cameraCfg.maxEXP // maximum time of exposure,unit: us
cameraCfg.minGain //minimum gain
cameraCfg.maxGain //maximum gain

**Description**:

Obtain information about the mode of the camera. Some cameras will support multiple video modes, which can be selected through camera_select_sensor().

### 1.3.2 Select Module's Working Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_select_sensor() | int sensorid | camera_control_get_features() obtains the array of information of supported camera modes. sensorid is the serial number of the array |

**Example**:

- camera_select_sensor(0);

Return:

0: Success; -1: Failure

**Description**:

Sets the working mode of the camera's visible light module.

### 1.3.3 Control Camera's Video Output

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_video_out() | int video_type | YUV Data output mode |
| | camera_ctrl_VIDEO_out mode | Disabled, Single (For photos), Continuous |

```
typedef enum
{
    VIDEO_OUT_DISABLE,          /* Output  Disabled */
    VIDEO_OUT_SINGLE,           /* Single Output */
    VIDEO_OUT_CONTINUOUS,       /* Continuous Output */
}camera_ctrl_video_out;
```

**Example**:

- camera_video_out(YUV420p, VIDEO_OUT_CONTINUOUS);

Return:

0: Success; -1:Failure

**Description**:

Sets the device to output video data. This works for YUV420p, H26X, JPEG. H26X does not support single output..

### 1.3.4 Set Camera's Focus Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_af_mode() | camera_ctrl_af_mode af_mode | CAMERA_CONTROL__AF_MODE_OFF : Manual Focus<br>CAMERA_CONTROL__AF_MODE_AUTO :Automatic Focus |

**Example**:

- camera_control_af_mode(CAMERA_CONTROL__AF_MODE_OFF);

Return:

0: Success; - 1: Failure

**Description**:

Sets the camera to manual focus. Using camera_control_get_features() one can check if the camera supports manual focusing (cameraCfg.AFmode). If not supported, the setting is invalid and the camera defaults to automatic.

### 1.3.5 Set Camera's Lens Distance

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_lens_move() | uint32_t lens_position | Range of distances(1-100) |

**Example**:

- camera_control_lens_move(10);

Return:

0: Success; -1: Failure

**Description**:

Used when focusing manually, greater value is a greater distance.

### 1.3.6 Trigger Single Focus

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_focus_trigger() | N/A | |

**Example**:

- camera_control_focus_trigger();

Return:

0 :Success; -1: Failure

**Description**:

Single focus

### 1.3.7 Set Camera's Exposure Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_ae_mode() | camera_ctrl_ae_mode<br>flash_mode | Manual or Automatic |

**Example**:

- camera_control_ae_mode(CAMERA_CONTROL__AE_AUTO__FLASH_MODE__AUTO);

Return:

0: Success; -1: Failure

**Description**:

Sets exposure mode.

### 1.3.8 Set Exposure Time

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_ae_set_exp() | uint32_t<br>exp_compensation | Exposure duration in microsecond (μs) range (1-1 / fps) |

**Example**:

- camera_control_ae_set_exp(20000);

Return:

0: Success; -1: Failure

**Description**:

Sets the exposure time for the manual exposure mode.

### 1.3.9 Set Camera Gain

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_ae_set_gain() | uint32_t iso_val | Gain value |

**Example**:

- camera_control_lens_move(100);

Return:

0: Success; -1: Failure

**Description**:

Sets the gain in manual exposure mode. Min/max gain values can be retrieved through camera_control_get_features() and set manually.

### 1.3.10 Set Camera White Balance Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_awb_mode() | camera_ctrl_awb_mode awb_mode | Manual or Auto |

**Example**:

- camera_control_awb_mode(CAMERA_CONTROL__AWB_MODE__AUTO);

Return:

0 :Success; -1: Failure

**Description**:

Sets camera to automatic white balance mode.

### 1.3.11 Float Conversion

| Interface Name | Interface Parameters | Description |
|---|---|---|
| f16Tof32() | unsigned int x | 16-bit Data |

**Example**:

- Float f=f16Tof32(100);

Return: float

**Description**:

Converts 16-bit short data to a floating point number. Used for metadata calculations and analysis.

# 2 Python SDK Interface Documentation

Starting from version 2.0.X onwards, the API will support Python. The SDK Interface can be found in the openncc.py file. To use the module, import it using: import openncc as ncc.

## 2.1. Device Initialization Related Interfaces

### 2.1.1 Get SDK Version

| Interface Name | Interface Parameters | Description |
|---|---|---|
| get_sdk_version() | void | N/A |

**Example**:

- print("get usb %d sdk versin %s" % (ncc.get_usb_version() ,ncc.get_sdk_version()))

Return:SDK Version

**Description**:

Gets the SDK version.

### 2.1.2 Get Version of Connected USB Device

| Interface Name | Interface Parameters | Description |
|---|---|---|
| get_usb_version() | void | N/A |

Return: 30 :usb3.0; 20: usb2.0

**Example**:

- print("get usb %d sdk versin %s" % (ncc.get_usb_version() ,ncc.get_sdk_version()))

**Description**:

Returns USB Version Information (Port and USB Cable) connected to the device

### 2.1.3 Load Device Firmware

| Interface Name | Interface Parameters | Description |
|---|---|---|
| load_fw() | bootExe | Path to USB boot program |
| | firmware | Path to firmware file |

Return:

0: Success; -1: Failure

**Example**:

- res = ncc.load_fw("./moviUsbBoot","fw/flicRefApp.mvcmd")

  if res<0:

```
printf('load firmware error!')
sys.exit(1)
```

**Description**:

Automatically loads device firmware, device boots, host (PC) opens USB device..

### 2.1.4 Initializing Camera Parameters

| Interface Name | Interface Parameters | Description |
|---|---|---|
| sdk_init() | vscRecvCb cb | Callback |
| | param | Callback function parameters |
| | Blob1_path | Path to AI Model (.blob) |
| | cam | Camera Configuration Parameters (See below) |
| | Cam_len | Camera configuration structure length |

There are two ways to get Media and Metadata. 1: Passively obtained through callback function, 2: Actively obtained through read_XXX_data() without setting the callback function and callback parameters..

**Example**:

```
cam_info=ncc.CameraInfo()
cam_info.inputFormat=ncc.IMG_FORMAT_BGR_PLANAR
cam_info.stdValue=1

cam_info.isOutputYUV=1
cam_info.isOutputH26X=1
cam_info.isOutputJPEG=1

cam_info.imageWidth    = cameraCfg.camWidth
cam_info.imageHeight = cameraCfg.camHeight
cam_info.startX          = 0
cam_info.startY          = 0
cam_info.endX             = cameraCfg.camWidth
cam_info.endY             = cameraCfg.camHeight
cam_info.inputDimWidth =0
cam_info.inputDimHeight =0
```

ncc.SetMeanValue(cam_info,0.0,0.0,0.0)

- ret = ncc.sdk_init(None, None, "./blob/face-detection-retail-0004-fp16.blob",cam_info, struct.calcsize("13I4f"))

- print("xlink_init ret=%d " % ret)
  if (ret<0):
  return

**Description**:

Specifies the AI Vision model file and calculation parameters, initializes the device algorithm model, camera function switch selection, sets the video encoding parameters (if the function switch is turned on). Video output is controlled by camera_video_out().

### 2.1.5 Initializing Camera with 2 AI Parameters

| Interface Name | Interface Parameters | Description |
|---|---|---|
| sdk_net2_init() | vscRecvCb cb | Callback |
| | param | Callback function parameters |
| | blob1_path | AI Mode 1 file(blob) |
| | par1 | Mode1 param |
| | par1_Len | Param1 length |
| | blob2_path | AI Mode 2 file(blob) |
| | par2 | Mode2 param |
| | par2_Len | Param2 length |

There are two ways to get Media and Metadata. 1: Passively obtained through callback function, 2: Actively obtained through read_XXX_data() without setting the callback function and callback parameters..

**Description**:

Specifies the AI Vision model file and calculation parameters, initializes the device algorithm model, camera function switch selection, sets the video encoding parameters (if the function switch is turned on). Video output is controlled by camera_video_out().

### 2.1.6 Uninitialize SDK

| Interface Name | Interface Parameters | Description |
|---|---|---|
| sdk_uninit() | N/A | N/A |

**Example**:

- sdk_uninit();

Return:N/A

**Description**:

Turn off the camera, reload the model, and call before changing the model.

## 2.2 Video Streaming Related Interfaces

### 2.2.1 Get YUV Data

| Interface Name | Interface Parameters | Description |
|---|---|---|
| GetYuvData() | yuvbuf | Bytearray receive buffer |

**Example**:

- metasize=ncc.get_meta_size()
  offset=struct.calcsize(media_head)
  yuvsize=cameraCfg.camWidth*cameraCfg.camHeight*2
  yuvbuf = bytearray(yuvsize+offset)
  metabuf = bytearray(metasize+offset)
- size = ncc.GetYuvData(yuvbuf)

Return:Size of the YUV data.

**Description**:

Gets a YUV data stream from the device.

Content: struct frameSpecOut + YUV(NV12) data.

### 2.2.2 Get H.264 or H.265 Data

| Interface Name | Interface Parameters | Description |
|---|---|---|
| GetH26xData() | databuf | Bytearray receive buffer |

**Example**:

Same as 2.2.1

**Description**:

Gets a H.264 or H.265 data stream from the device.

Content: struct frameSpecOut + H26X data..

### 2.2.3 Get JPEG data

| Interface Name | Interface Parameters | Description |
|---|---|---|
| GetJpegData() | databuf | Bytearray receive buffer |

**Example**:

Same as 2.2.1

**Description**:

Gets a JPEG data stream from the device.

Content: struct frameSpecOut + MJPEG data.

### 2.2.4 Get the results of the AI Network inference

| Interface Name | Interface Parameters | Description |
|---|---|---|
| GetMetaData() | databuf | Bytearray receive buffer |

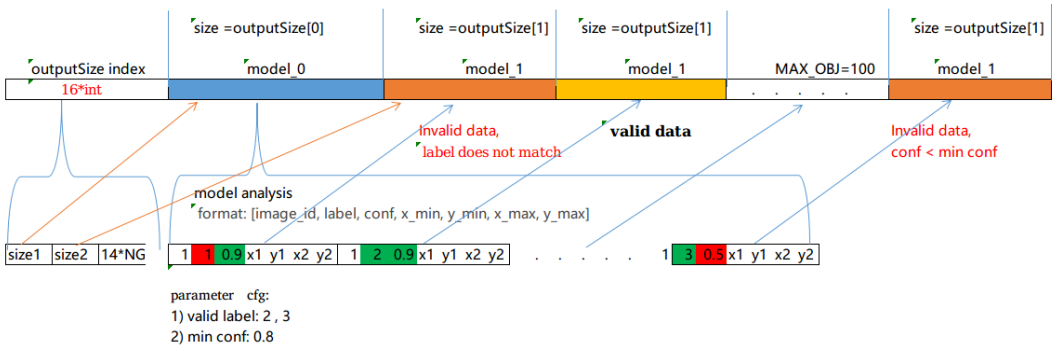
**Example**:

Same as 2.2.1

**Description**:

Get the number of operations from the device's AI Network.

Content: struct frameSpecOu t+ AI data and AI data format is below:




## 2.3 Camera Control Related Interfaces

### 2.3.1 Obtain Camera Module Information

| Interface Name | Interface Parameters | Description |
|---|---|---|
| CameraSensor class | GetFirstSensor(),GetNextSensor() | |

**Example**:

- sensors=ncc.CameraSensor()
- sensor1 = ncc.SensorModesConfig()
- if sensors.GetFirstSensor(sensor1)==0:

print("camera: %s, %dX%d@%dfps, AFmode:%d,
maxEXP:%dus,gain[%d, %d]\n" % (
sensor1.moduleName, sensor1.camWidth, sensor1.camHeight, sensor1.camFps,
sensor1.AFmode, sensor1.maxEXP, sensor1.minGain, sensor1.maxGain))

- sensor2 = ncc.SensorModesConfig()
- while sensors.GetNextSensor(sensor2)==0:
  print("camera: %s, %dX%d@%dfps, AFmode:%d,
maxEXP:%dus,gain[%d, %d]\n" % (
sensor2.moduleName, sensor2.camWidth, sensor2.camHeight, sensor2.camFps,
sensor2.AFmode, sensor2.maxEXP,sensor2.minGain, sensor2.maxGain))

**Description**:

Obtains information about the mode of the camera. Some cameras will support multiple video modes, which can be selected through camera_select_sensor().

### 2.3.2 Select Module's Working Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_select_sensor() | sensorid | camera_control_get_features() obtains the array of information of supported camera modes. sensorid is the serial number of the array. |

**Example**:

- ncc.camera_select_sensor(0)

Return:

0: Success; -1:Failure

**Description**:

Sets the working mode of the camera's visible light module.

### 2.3.3 Control the Camera's Video Output

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_video_out() | video_type | Video data type |
| | out mode | Disabled, Single (For photos), or Continuous |

**Example**:

- ncc.camera_video_out(ncc.YUV420p,ncc.VIDEO_OUT_CONTINUOUS)

Return:

0: Success ;-1: Failure

**Description**:

Sets the device to output video data. This works for YUV420p, H26X, JPEG. H26X does not support single output.

### 2.3.4 Set Camera's Focus Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_af_mode() | camera_ctrl_af_mode af_mode | CAMERA_CONTROL__AF_MODE_OFF: Manual Focus<br>CAMERA_CONTROL__AF_MODE_AUTO :Automatic Focus |

**Example**:

- ncc.camera_control_af_mode(ncc.CAMERA_CONTROL__AF_MODE_AUTO);

Return:

0: Success; -1: Failure

**Description**:

Sets the camera to manual focus. Using camera_control_get_features() one can check if the camera supports manual focusing (cameraCfg.AFmode). If not supported, the setting is invalid and the camera defaults to automatic focusing.

### 2.3.5 Set the Camera's Lens Distance

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_lens_move() | lens_position | Range of distances(1-100) |

**Example**:

- ncc.camera_control_lens_move(10);

Return:

0: Success; -1:Failure

**Description**:

Used when focusing manually, greater value is a greater distance.

### 2.3.6 Trigger Single Focus

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_focus_trigger() | void | N/A |

**Example**:

- camera_control_focus_trigger();

Return:

0: Success; -1: Failure

**Description**:

Focuses the camera once.

### 2.3.7 Set Camera's Exposure Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_ae_mode() | camera_ctrl_ae_mode<br>flash_mode | Manual or Automatic |

**Example**:

- ncc.camera_control_ae_mode( ncc.CAMERA_CONTROL__AE_AUTO__FLASH_MODE__AUTO);

Return:

0 :Success; -1: Failure

**Description**:

Sets exposure mode.

### 2.3.8 Set Exposure Time

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_ae_set_exp() | exp_compensation | Exposure duration in microsecond (μs) range (1-1 / fps) |

**Example**:

- ncc.camera_control_ae_set_exp(20000);

Return:

0: Success; -1: Failure

**Description**:

Sets the exposure time for the manual exposure mode.

### 2.3.9 Set Camera Gain

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_ae_set_gain() | iso_val | Gain value |

**Example**:

- ncc.camera_control_lens_move(100);

Return:

0: Success; -1: Failure

**Description**:

Sets the gain in manual exposure mode. Min/max gain values can be retrieved through camera_control_get_features() and set manually..

### 2.3.10 Set Camera White Balance Mode

| Interface Name | Interface Parameters | Description |
|---|---|---|
| camera_control_awb_mode() | camera_ctrl_awb_mode awb_mode | Manual or Automatic |

**Example**:

- ncc.camera_control_awb_mode(ncc.CAMERA_CONTROL__AWB_MODE__AUTO);

Return:

0: Success; -1: Failure

**Description**:

Sets camera to automatic white balance mode..

### 2.3.11 Float Conversion

| Interface Name | Interface Parameters | Description |
|---|---|---|
| f16Tof32() | x | 16-bit Data |

**Example**:

- f=f16Tof32(100);

Return: Float

**Description**:

Converts 16-bit short data to a floating point number. Used for metadata calculations and analysis.