Machine Vision Camera SDK (Android)

Developer Guide

Legal Information

Contents

Chapter 1 Overview

Machine vision camera SDK (MvCameraSDK) contains API definitions, objects, camera driver and so on. It is compatible with standard protocols, and currently, GigEVision and USB3Vision protocols are supported.

1.1 Introduction

This manual mainly introduces the MvCameraSDK based on Java language, which provides several APIs to implement the functions of image acquisition, parameter configuration, file access, and so on.

Parameter configuration and image acquisition are two basic functions, see details below:

- Parameter configuration: Get and set all parameters of cameras, such as image width, height, exposure time, which are realized by the general configuration API.
- Image acquisition: When the camera sends image data to Android devices, the image data will be saved to the SDK. SDK provides two methods for getting the image, including search method and callback method. The two methods cannot be adopted at same time, you should choose one method according to actual application.

Caution

All the APIs are defined in the class MvCameaControl, and all the structures are defined in the class MvCameraControlDefines.

1.2 Development Environment

The development environment of MvCameraSDK is shown in the table below.

Item	Required
Hardware Environment	RK3399, RK3128
Software Environment	Android 7.1
Dependent Library	GenICam library, MvCameraControl, MVGigEVisionSDK, libMvUsb3vTL

1.3 Update History

The update history shows the summary of changes in MvCameraSDK (Android) with different versions.

Summary of Changes in Version 1.0.0_Jan., 2020

New document.

Chapter 2 Typical Applications

2.1 Connect Device

Before operating the device to implement the functions of image acquisition, parameter configuration, and so on, you should connect the device (open device).

Steps

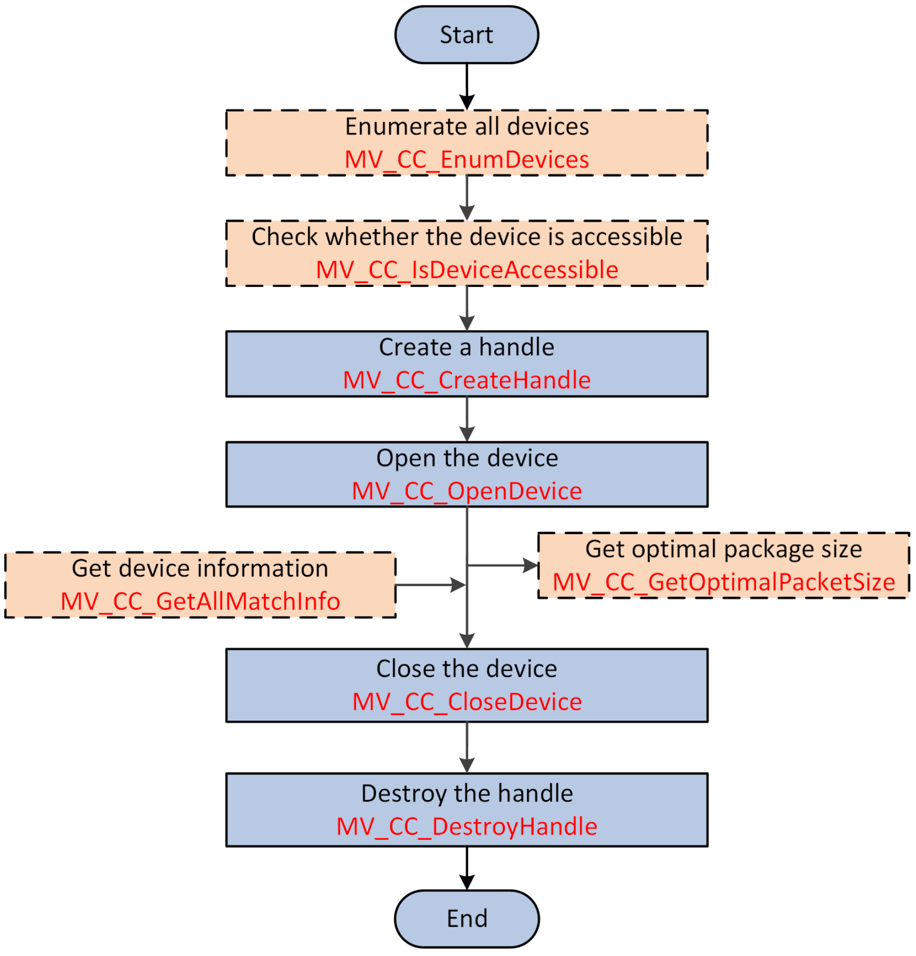


Figure 2-1 Programming Flow of Connecting Device

1. **Optional:** Call ***MV_CC_EnumDevices*** to enumerate all devices.

   The information of found devices is returned in the structure ***MV_CC_DEVICE_INFO*** by **nTLayerType**.
2. **Optional:** Call ***MV_CC_IsDeviceAccessible*** to check whether the specified device is accessible before opening it.
3. Call ***MV_CC_CreateHandle*** to create a device handle.
4. Call ***MV_CC_OpenDevice*** to open the device.

5. **Optional:** Perform the following operation(s) after connecting device.

Get Device Information	Call ***MV_CC_GetAllMatchInfo***
Get Optimal Package Size	Call ***MV_CC_GetOptimalPacketSize***

6. Call ***MV_CC_CloseDevice*** to close the device.
7. Call ***MV_CC_DestroyHandle*** to destroy the handle and release resources.

2.2 Get Image Directly

You can directly get the image after starting getting stream, or adopt asynchronous mode (thread or timer) to get the image.

Steps

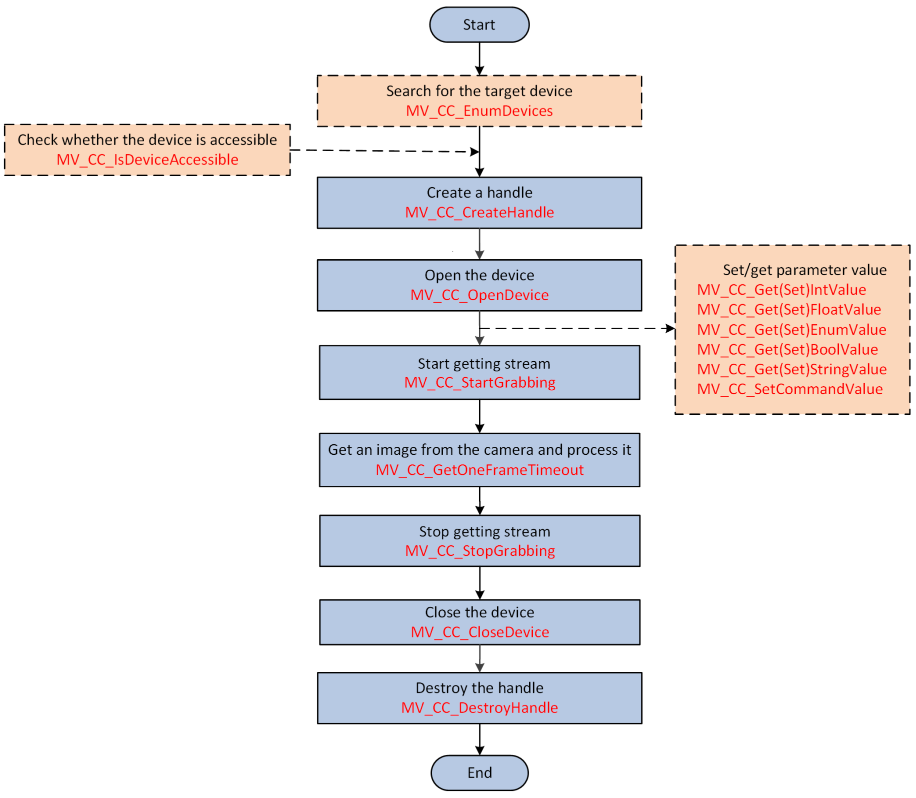


Figure 2-2 Programming Flow of Getting Image Directly

Note

All the camera properties can be referred to the node sheet " ***MvCameraNode*** .xlsx" under the installation directory. The name, definition, data type, value range, and access mode of each node are described in the sheet.

1. Call ***MV_CC_EnumDevices*** to enumerate all devices.

The information of found devices is returned in the structure ***MV_CC_DEVICE_INFO*** by **nTLayerType**.

2. **Optional:** Call ***MV_CC_IsDeviceAccessible*** to check if the specified device is accessible before opening it.
3. Call ***MV_CC_CreateHandle*** to create a device handle.
4. Call ***MV_CC_OpenDevice*** to open the device.
5. **Optional:** Perform one or more of the following operations to get/set different types parameters.

Parameters of type int	Call ***MV_CC_GetIntValue*** / ***MV_CC_SetIntValue***
Parameters of type float	Call ***MV_CC_GetFloatValue*** / ***MV_CC_SetFloatValue***
Parameters of type enum	Call ***MV_CC_GetEnumValue*** / ***MV_CC_SetEnumValue***
Parameters of type boolean	Call ***MV_CC_GetBoolValue*** / ***MV_CC_SetBoolValue***
Parameters of type string	Call ***MV_CC_GetStringValue*** / ***MV_CC_SetStringValue***
Parameters of type command	Call ***MV_CC_SetCommandValue***

6. Acquire images.
   1) **Optional:** Call ***MV_CC_SetImageNodeNum*** to set the number of image cache nodes.

      When the number of obtained images is larger than this number, the earliest image data will be discarded automatically.
   2) Call ***MV_CC_StartGrabbing*** to start getting streams.

      Note

      For original image data, you can call ***MV_CC_ConvertPixelType*** to convert the image pixel format, or you can call ***MV_CC_SaveImage*** to convert the image to JPEG or BMP format and save as a file.
   3) Call ***MV_CC_GetOneFrameTimeout*** repeatedly in the application layer to get the frame data with specified pixel format.
7. Call ***MV_CC_StopGrabbing*** to stop the acquisition.
8. Call ***MV_CC_CloseDevice*** to close the device.
9. Call ***MV_CC_DestroyHandle*** to destroy the handle and release resources.

2.3 Get Image in Callback Function

The API MV_CC_RegisterImageCallBackEx is provided for registering callback function. You can customize the callback function and the obtained image will automatically called back. This method can simplify the application logic.

Steps

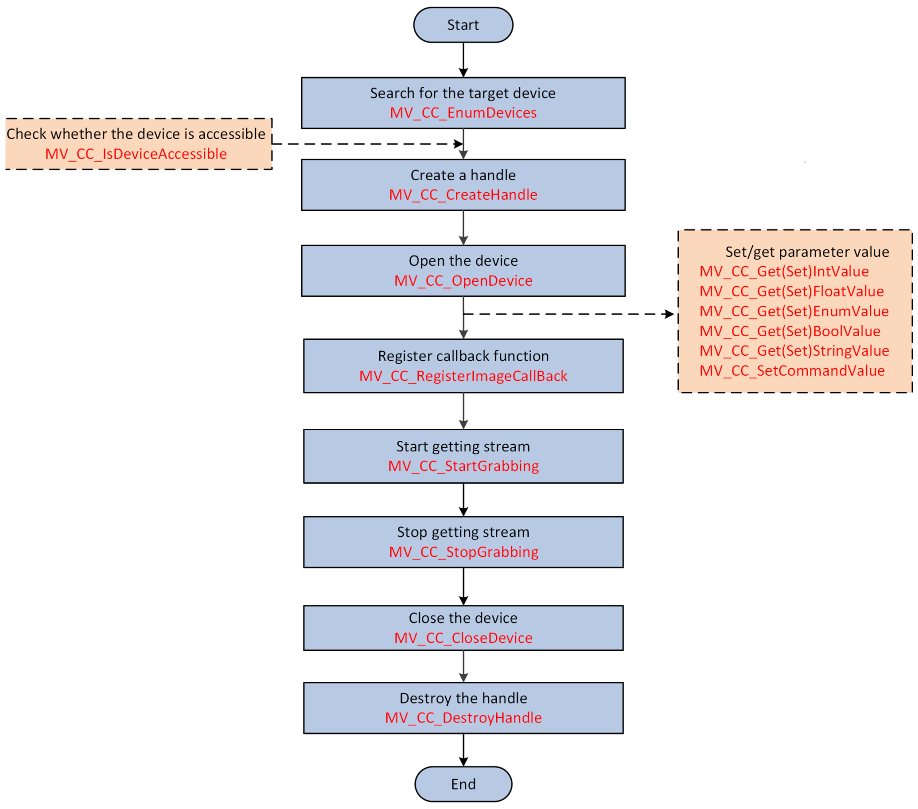


Figure 2-3 Programming Flow of Getting Image in Callback Function

Note

All the open properties of the camera can be referred to the node sheet " ***MvCameraNode*** .xlsx" under the installation directory. The name, definition, data type, value range, and access mode of each node are described in the sheet.

1. Call ***MV_CC_EnumDevices*** to enumerate all devices.

   The information of found devices is returned in the structure ***MV_CC_DEVICE_INFO*** by **nTLayerType**.
2. **Optional:** Call ***MV_CC_IsDeviceAccessible*** to check if the specified device is accessible before opening it.
3. Call ***MV_CC_CreateHandle*** to create a device handle.
4. Call ***MV_CC_OpenDevice*** to open the device.
5. **Optional:** Perform one or more of the following operations to get/set different types parameters.

Parameters of type int	Call ***MV_CC_GetIntValue*** / ***MV_CC_SetIntValue***
Parameters of type float	Call ***MV_CC_GetFloatValue*** / ***MV_CC_SetFloatValue***
Parameters of type enum	Call ***MV_CC_GetEnumValue*** / ***MV_CC_SetEnumValue***

Parameters of type boolean	Call ***MV_CC_GetBoolValue*** / ***MV_CC_SetBoolValue***
Parameters of type string	Call ***MV_CC_GetStringValue*** / ***MV_CC_SetStringValue***
Parameters of type command	Call ***MV_CC_SetCommandValue***

6. Acquire images.
   1) **Optional:** Call ***MV_CC_SetImageNodeNum*** to set the number of image cache nodes.

      When the number of obtained images is larger than this number, the earliest image data will be discarded automatically.
   2) Call ***MV_CC_RegisterImageCallBack*** to register image data callback function.
   3) Call ***MV_CC_StartGrabbing*** to start the acquisition.

      Note

      For original image data, you can call ***MV_CC_ConvertPixelType*** to convert the image pixel format, or you can call ***MV_CC_SaveImage*** to convert the image to JPEG or BMP format and save as a file.

7. Call ***MV_CC_StopGrabbing*** to stop the acquisition.
8. Call ***MV_CC_CloseDevice*** to close the device.
9. Call ***MV_CC_DestroyHandle*** to destroy the handle and release resources.

Chapter 3 API Reference

3.1 General APIs

3.1.1 MV_CC_GetSDKVersion

Get the SDK version No.

API Definition

```
public native static String MV_CC_GetSDKVersion(
);
```

Return Value

Return SDK version No. and compilation date for success.

Example

The following sample code is for reference only.

```
public void GetSDKVersion(){
  String version = MvCameraControl.MV_CC_GetSDKVersion();
}
```

3.1.2 MV_CC_EnumDevices

Enumerate devices.

API Definition

```
public native static ArrayList<MV_CC_DEVICE_INFO> MV_CC_EnumDevices(
  int       nTLayerType
);
```

Parameters

nTLayerType

[IN] Transport layer protocol type

Return Value

Return device list for success, and return *NULL* or exception information for failure.

Example

The following sample code is for reference only.

```
public void EnumDevices() {
  try {
     ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList
= MvCameraControl.MV_CC_EnumDevices(MvCameraControlDefines.MV_GIGE_DEVICE |
MvCameraControlDefines.MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
     Log.e("CameraControl", e.errCode + e.errMsg);
  }
}
```

3.1.3 MV_CC_EnumerateTls

Get supported transport layers.

API Definition

```
public native static int MV_CC_EnumerateTls(
);
```

Return Value

Return supported device types. e.g., "nTLayerType = MyCamera.MV_GIGE_DEVICE | MyCamera.MV_USB_DEVICE" indicates that GigE device and USB3.0 device are both supported. Available protocol types are shown below:

Macro Definition	Description
MV_GIGE_DEVICE	GigE Device
MV_USB_DEVICE USB	USB Device

Example

The following sample code is for reference only.

```
public void EnumerateTls() {
  int nTransLayers = MvCameraControl.MV_CC_EnumerateTls();
  if (nTransLayers == MvCameraControlDefines.MV_GIGE_DEVICE) {
     Log.e("CameraControl", "GigeDevice");
  } else if (nTransLayers == MvCameraControlDefines.MV_USB_DEVICE) {
     Log.e("CameraControl", "UsbDevice");
  } else if (nTransLayers == (MvCameraControlDefines.MV_GIGE_DEVICE +
MvCameraControlDefines.MV_USB_DEVICE)) {
     Log.e("CameraControl", "GigeDevice and UsbDevice");
  }
}
```

3.1.4 MV_CC_IsDeviceAccessible

Check whether the specified device is accessible.

API Definition

```
public native static boolean MV_CC_IsDeviceAccessible(
  MV_CC_DEVICE_INFO    deviceInfo,
  int                  accessMode
);
```

Parameters

devInfo

[IN] Device information, see ***MV_CC_DEVICE_INFO*** for details.

accessMode

[IN] Access mode

Return Value

Return *true* to indicate that the device is accessible, and return *false* to indicate that the device is not accessible or offline. Return false if the device does not support the modes MV_ACCESS_ExclusiveWithSwitch, MV_ACCESS_ControlWithSwitch, or MV_ACCESS_ControlSwitchEnableWithKey.

Remarks

Currently the device does not support the preemption modes, neither do the devices of other manufacturers.

Example

The following sample code is for reference only.

```
public void IsDeviceAccessible() {
  try {
      ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList =
MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
      if (devicesList == null) {
    Log.e("CameraControl", "error: EnumDevices fail");
    return;
      }
      if (devicesList.size() > 0) {
    //The first device is selected by default
         Boolean falg = MvCameraControl.MV_CC_IsDeviceAccessible(devicesList.get(0),
MvCameraControlDefines.MV_ACCESS_Exclusive);
    Log.e("CameraControl", "isDeviceAccessible:" + falg);
      }

  } catch (CameraControlException e) {
      e.printStackTrace();
      Log.e("CameraControl", "error: EnumDevices fail nRet" + e.errCode);
  }
}
```

3.1.5 MV_CC_CreateHandle

Create a handle and check whether to enable logging.

API Definition

```
public native static Handle MV_CC_CreateHandle(
  MV_CC_DEVICE_INFO     devInfo,
  boolean               logFlag
);
```

Parameters

devInfo

[IN] Device information, see ***MV_CC_DEVICE_INFO*** for details.

logFlag

[IN] Whether to enable logging: "true"-enable logging, "false"-disable logging

Return Value

Return a handle for success, and return ***Error Code*** for failure.

Remarks

Create the required resources in the library and initialize the internal modules according to the device information.

3.1.6 MV_CC_CreateHandle

Create a handle and log files will be automatically generated.

API Definition

```
public native static Handle MV_CC_CreateHandle(
  MV_CC_DEVICE_INFO     devInfo
);
```

Parameters

devInfo

[IN] Device information, see ***MV_CC_DEVICE_INFO*** for details.

Return Value

Return a handle for success, and return ***Error Code*** for failure.

Remarks

Create the required resources in the library and initialize the internal modules according to the device information.

Example

The following sample code is for reference only.

```
public void CreateHandle() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
    } catch (CameraControlException e) {
      e.printStackTrace();
    }
  if (devicesList == null ) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
    }
        If(devicesList.size() == 0){
    Return;
  }
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  MvCameraControlDefines.Handle handle = null;
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      Log.e("CameraControl", e.errCode + e.errMsg);
      return;
  }
  int nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.1.7 MV_CC_DestroyHandle

Destroy handle and release resources.

API Definition

```
public native static int MV_CC_DestroyHandle(
  Handle    handle
);
```

Parameters

handle

[IN] Device handle

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void DestroyHandle() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
      e.printStackTrace();
  }
  if (devicesList == null) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      Log.e("CameraControl", "error: CreateHandle fail:" + e.errCode);
      return;
  }
  int nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.1.8 MV_CC_OpenDevice

Open the device and set the device access mode.

API Definition

```
public native static int MV_CC_OpenDevice(
  Handle        handle,
  int           accessMode,
  short         switchoverKey
);
```

Parameters

handle

[IN] Device handle

accessMode

[IN] Device access mode, which is exclusive mode by default.

switchoverKey

[IN] Key for switching permissions

Return Value

Return *MV_OK* for success, and return ***Error Code*** on failure.

3.1.9 MV_CC_OpenDevice

Open the device and the device access mode is exclusive mode by default.

API Definition

```
public native static int MV_CC_OpenDevice(
  Handle       handle
);
```

Parameters

handle

[IN] Device handle

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void OpenDevice() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }

  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: OpenDevice fail");
      return;
  }

  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: CloseDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.1.10 MV_CC_CloseDevice

Close the device.

API Definition

```
public native static int MV_CC_CloseDevice(
  Handle    handle
);
```

Parameters

handle

[IN] Device handle

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void CloseDevice() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
      e.printStackTrace();
  }
  if (devicesList == null) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }

  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: OpenDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: CloseDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.2 Parameter Settings APIs

3.2.1 MV_CC_SetBoolValue

Set the parameter value of type boolean.

API Definition

```
public native static int MV_CC_SetBoolValue(
  Handle    handle,
  String    strKey,
  boolean   boolValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

boolValue

[IN] Parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void SetBoolValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_SetBoolValue(handle, "Gain", false);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: SetBoolValue fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
     return;
  }
}
```

3.2.2 MV_CC_GetBoolValue

Get the parameter value of type boolean.

API Definition

```
public native static int MV_CC_GetBoolValue(
  Handle    handle,
  String    strKey,
  Boolean   boolValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

boolValue

[OUT] Obtained parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void GetBoolValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
```

```
 }
 MvCameraControlDefines.Handle handle = null;
 MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
 try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
 } catch (CameraControlException e) {
     e.printStackTrace();
     return;
 }
 int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
 if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail" + nRet);
     return;
 }
 Boolean booleanvalue = new Boolean(false);
 nRet = MvCameraControl.MV_CC_GetBoolValue(handle, "Gain", booleanvalue);
 if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: GetBoolValue fail" + nRet);
     return;
 }
 nRet = MvCameraControl.MV_CC_CloseDevice(handle);
 if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail" + nRet);
     return;
 }
 nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
 if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
     return;
 }
}
```

3.2.3 MV_CC_SetEnumValue

Set the parameter value of type enum.

API Definition

```
public native static int MV_CC_SetEnumValue(
 Handle      handle,
 String      strKey,
 int         value
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

value

[IN] Parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void SetEnumValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_SetEnumValue(handle, "GainAuto", 0);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: SetEnumValue fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
     return;
  }
}
```

3.2.4 MV_CC_GetEnumValue

Get the parameter value of type enum.

API Definition

```
public native static int MV_CC_GetEnumValue(
  Handle       handle,
  String       strKey,
  Integer      enumValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

pEnumValue

[OUT] Obtained parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void GetEnumValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: OpenDevice fail" + nRet);
      return;
  }
  MvCameraControlDefines.MVCC_ENUMVALUE enumvalue = new MvCameraControlDefines.MVCC_ENUMVALUE();
  nRet = MvCameraControl.MV_CC_GetEnumValue(handle, "GainAuto", enumvalue);

  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: GetEnumValue fail" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: CloseDevice fail" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
      return;
  }
}
```

3.2.5 MV_CC_GetEnumValue

Get the parameter value of type enum.

API Definition

```
public native static int MV_CC_GetEnumValue(
  Handle          handle,
  String          strKey,
  MVCC_ENUMVALUE     enumValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

pEnumValue

[OUT] Obtained parameter value, see ***MVCC_ENUMVALUE*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.2.6 MV_CC_SetFloatValue

Set the parameter value of type float.

API Definition

```
public native static int MV_CC_SetFloatValue(
  Handle    handle,
  String    strKey,
  float     floatValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

floatValue

[IN] Parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void SetFloatValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
        e.printStackTrace();
    }
    if (devicesList == null) {
        Log.e("CameraControl", "error: EnumDevices fail");
        return;
    }
    MvCameraControlDefines.Handle handle = null;
    MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
    try {
        handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
        if (handle == null) {
      Log.e("CameraControl", "error: CreateHandle fail");
      return;
        }
    } catch (CameraControlException e) {
        e.printStackTrace();
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_SetFloatValue(handle, "Gain", 0f);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: SetFloatValue fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
        return;
    }
}
```

3.2.7 MV_CC_GetFloatValue

Get the parameter value of type float.

API Definition

```
public native static int MV_CC_GetFloatValue(
  Handle      handle,
  String      strKey,
 Float       floatValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

floatValue

[OUT] Obtained parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void GetFloatValue() {
 ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
 try {
    devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
 } catch (CameraControlException e) {
    e.printStackTrace();
 }
 if (devicesList == null) {
    Log.e("CameraControl", "error: EnumDevices fail");
    return;
 }
 MvCameraControlDefines.Handle handle = null;
 MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
 try {
    handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
    if (handle == null) {
  Log.e("CameraControl", "error: CreateHandle fail");
  return;
    }
 } catch (CameraControlException e) {
    e.printStackTrace();
    return;
 }
 int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
 if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: OpenDevice fail" + nRet);
    return;
 }
 MvCameraControlDefines.MVCC_FLOATVALUE floatvalue = new MvCameraControlDefines.MVCC_FLOATVALUE();
 nRet = MvCameraControl.MV_CC_GetFloatValue(handle, "Gain", floatvalue);
 if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: GetFloatValue fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
        return;
    }
}
```

3.2.8 MV_CC_GetFloatValue

Get parameters of type float.

API Definition

```
public native static int MV_CC_GetFloatValue(
  Handle           handle,
  String           strKey,
  MVCC_FLOATVALUE  floatValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name

floatValue

[OUT] Obtained parameter value, see ***MVCC_FLOATVALUE*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.2.9 MV_CC_SetIntValue

Set the parameter value of type int.

API Definition

```
public native static int MV_CC_SetIntValue(
  Handle    handle,
  String    strKey,
  long      value
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

value

[IN] Parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void SetIntValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail" + nRet);
```

```
        return;
    }
    nRet = MvCameraControl.MV_CC_SetIntValue(handle, "Width", 1000);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: SetIntValue fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
        return;
    }
}
```

3.2.10 MV_CC_GetIntValue

Get the parameter value of type int.

API Definition

```
public native static int MV_CC_GetIntValue(
  Handle            handle,
  String            strKey,
  MVCC_INTVALUE     intValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name

intValue

[OUT] Obtained parameter value, see ***MVCC_INTVALUE*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.2.11 MV_CC_GetIntValue

Get parameters of type int.

API Definition

```
public native static int MV_CC_GetIntValue(
  Handle     handle,
  String     strKey,
  Integer    intValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

intValue

[OUT] Obtained parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void GetIntValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
```

```
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail" + nRet);
        return;
    }
    MvCameraControlDefines.MVCC_INTVALUE intvalue = new MvCameraControlDefines.MVCC_INTVALUE();
    nRet = MvCameraControl.MV_CC_GetIntValue(handle, "Width", intvalue);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: GetIntValue fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
        return;
    }
}
```

3.2.12 MV_CC_SetStringValue

Set the parameter value of type String.

API Definition

```
public native static int MV_CC_SetStringValue(
  Handle       handle,
  String      strKey,
  String      stringValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

stringValue

[IN] Parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void SetStringValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_SetStringValue(handle, "DeviceUserID", "hikCamera");
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: SetStringValue fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
     return;
  }
}
```

3.2.13 MV_CC_GetStringValue

Get the parameter value of type String.

API Definition

```
public native static int MV_CC_GetStringValue(
 Handle              handle,
 String              strKey,
 MVCC_STRINGVALUE    stringValue
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

stringValue

[OUT] Obtained parameter value, see ***MVCC_STRINGVALUE*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void GetStringValue() {
 ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
 try {
    devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
 } catch (CameraControlException e) {
    e.printStackTrace();
 }
 if (devicesList == null) {
    Log.e("CameraControl", "error: EnumDevices fail");
    return;
 }
 MvCameraControlDefines.Handle handle = null;
 MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
 try {
    handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
    if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
    }
 } catch (CameraControlException e) {
    e.printStackTrace();
```

```
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail" + nRet);
        return;
    }
    MvCameraControlDefines.MVCC_STRINGVALUE stringvalue = new MvCameraControlDefines.MVCC_STRINGVALUE();
    nRet = MvCameraControl.MV_CC_GetStringValue(handle, "DeviceUserID", stringvalue);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: GetStringValue fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail" + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
        return;
    }
}
```

3.2.14 MV_CC_SetEnumValueByString

Set the parameter value of type String.

API Definition

```
public native static int MV_CC_SetEnumValueByString(
    Handle      handle,
    String      strKey,
    String      value
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

value

[IN] Parameter value

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.2.15 MV_CC_SetCommandValue

Set the parameter value of type Command.

API Definition

```
public native static int MV_CC_SetCommandValue(
  Handle     handle,
  String     strKey
);
```

Parameters

handle

[IN] Device handle

strKey

[IN] Parameter name, see ***MvCameraNode*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
void SetCommandValue() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
      e.printStackTrace();
  }
  if (devicesList == null) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
    return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: OpenDevice fail" + nRet);
    return;
  }
  nRet = MvCameraControl.MV_CC_SetCommandValue(handle, "DeviceReset");
  if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: SetCommandValue fail" + nRet);
    return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: CloseDevice fail" + nRet);
    return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
    return;
  }
   }
```

3.2.16 MV_XML_GetGenICamXML

Get the camera description file in XML format.

API Definition

```
public native static int MV_XML_GetGenICamXML(
  Handle      handle,
  byte[]      data,
  int         dataLen
);
```

Parameters

handle

[IN] Device handle

pata

[OUT] The XML file buffer address

dataLen

[OUT] The XML file size

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void GetGenICamXML() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
     return;
  }
  int MAX_XML_FILE_SIZE = (1024 * 1024 * 3);
  byte[] data = new byte[MAX_XML_FILE_SIZE];
  Integer nXMLDataLen = new Integer(0);
  nRet = MvCameraControl.MV_XML_GetGenICamXML(handle, data, nXMLDataLen);
  if (MvCameraControlDefines.MV_OK != nRet || nXMLDataLen > MAX_XML_FILE_SIZE) {
     Log.e("CameraControl", "error: GetGenICamXML failed! " + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail " + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail " + nRet);
     return;
  }
}
```

3.3 Functional APIs

3.3.1 MV_CC_GetAllMatchInfo

Get camera information of all types.

API Definition

```
public native static int MV_CC_GetAllMatchInfo(
  Handle              handle,
  MV_ALL_MATCH_INFO     info
);
```

Parameters

handle

[IN] Device handle

info

[OUT] Camera information structure, see ***MV_ALL_MATCH_INFO*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void GetAllMatchInfo() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
```

```
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StartGrabbing fail");
        return;
    }
    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    MvCameraControlDefines.MV_ALL_MATCH_INFO info = new MvCameraControlDefines.MV_ALL_MATCH_INFO();
    nRet = MvCameraControl.MV_CC_GetAllMatchInfo(handle, info);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: GetAllMatchInfo fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "Operation: Stop Grabbing......failed!");
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail " + nRet);
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail " + nRet);
        return;
    }
}
```

3.3.2 MV_CC_GetDeviceInfo

Get camera information after opening the camera.

API Definition

```
public native static int MV_CC_GetDeviceInfo(
  Handle              handle,
  MV_CC_DEVICE_INFO     info
);
```

Parameters

handle

[IN] Device handle

info

[OUT] Camera information structure, see ***MV_CC_DEVICE_INFO*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Remarks

For GigE cameras, it is not recommended to call this API when getting stream as the danger of blocking.

3.3.3 MV_CC_SetImageNodeNum

Set the number of image nodes.

API Definition

```
public native static int MV_CC_SetImageNodeNum(
  Handle    handle,
  int       num
);
```

Parameters

handle

[IN] Device handle

num

[IN] The number of image nodes, the default value is "1"; range: [1, 30].

Return Value

Return *MV_OK* on success, and return ***Error Code*** on failure.

Example

The following sample code is for reference only.

```
public void SetImageNodeNum() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
        Log.e("CameraControl", "error: EnumDevices fail");
        return;
    }
    MvCameraControlDefines.Handle handle = null;
    MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
    try {
        handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
        if (handle == null) {
      Log.e("CameraControl", "error: CreateHandle fail");
      return;
        }
    } catch (CameraControlException e) {
        e.printStackTrace();
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_SetImageNodeNum(handle, 1);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: SetImageNodeNum fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StartGrabbing fail");
        return;
    }
    MvCameraControlDefines.MVCC_INTVALUE intvalue = new MvCameraControlDefines.MVCC_INTVALUE();
    nRet = MvCameraControl.MV_CC_GetIntValue(handle, "PayloadSize", intvalue);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: PayloadSize fail");
        return;
    }
    byte[] datas = new byte[(int) intvalue.curValue + 2048];
    MvCameraControlWrapper.MvCameraControlDefines.MV_FRAME_OUT_INFO info = new
MvCameraControlDefines.MV_FRAME_OUT_INFO();
    for (int i = 0; i < 50; i++) {
        nRet = MvCameraControl.MV_CC_GetOneFrameTimeout(handle, datas, info, 1000);
        if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: GetOneFrameTimeout fail");
      return;
        }
    }
    nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StopGrabbing fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
```

```
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.3.4 MV_CC_RegisterExceptionCallBack

Register exception message callback.

API Definition

```
public native static int MV_CC_RegisterExceptionCallBack(
  Handle                    handle,
  CameraExceptionCallBack     callBack
);
```

Parameters

handle

[IN] Device handle

fExceptionCallBack

[IN] Callback function to receive exception messages, see ***CameraExceptionCallBack*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Remarks

Call this API after opening device.

Example

The following sample code is for reference only.

```
public void RegisterExceptionCallBack() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_RegisterExceptionCallBack(handle, new CameraExceptionCallBack() {
     @Override
     public int OnExceptionCallBack(int i) {
   return 0;
     }
  });
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: RegisterExceptionCallBack fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: StartGrabbing fail");
     return;
  }
  MvCameraControlDefines.MVCC_INTVALUE intvalue = new MvCameraControlDefines.MVCC_INTVALUE();
  nRet = MvCameraControl.MV_CC_GetIntValue(handle, "PayloadSize", intvalue);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: PayloadSize fail");
     return;
  }
  byte[] datas = new byte[(int) intvalue.curValue + 2048];
  MvCameraControlWrapper.MvCameraControlDefines.MV_FRAME_OUT_INFO info = new
MvCameraControlDefines.MV_FRAME_OUT_INFO();
  for (int i = 0; i < 50; i++) {
     nRet = MvCameraControl.MV_CC_GetOneFrameTimeout(handle, datas, info, 1000);
     if (nRet != MvCameraControlDefines.MV_OK) {
   Log.e("CameraControl", "error: GetOneFrameTimeout fail");
   return;
     }
  }
  nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: StopGrabbing fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: CloseDevice fail");
    return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.3.5 MV_CC_RegisterEventCallBack

Register event callback function.

API Definition

```
public native static int MV_CC_RegisterEventCallBack(
  Handle                handle,
  CameraEventCallBack   callBack
);
```

Parameters

handle

[IN] Device handle

callBack

[OUT] Callback function for receiving event information. See ***CameraEventCallBack*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Remarks

Call this API after opening the camera.

Example

The following sample code is for reference only.

```
public void RegisterEventCallBack() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
    devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
    e.printStackTrace();
  }
  if (devicesList == null) {
    Log.e("CameraControl", "error: EnumDevices fail");
    return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
    handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
```

```
        if (handle == null) {
      Log.e("CameraControl", "error: CreateHandle fail");
      return;
        }
    } catch (CameraControlException e) {
        e.printStackTrace();
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_RegisterEventCallBack(handle, "ExposureEnd", new CameraEventCallBack() {
        @Override
        public int OnEventCallBack(MvCameraControlDefines.MV_EVENT_OUT_INFO mv_event_out_info) {
      return 0;
        }
    });
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: RegisterEventCallBack fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
        return;
    }
}
```

3.3.6 MV_CC_FeatureSave

Save the camera's configuration files.

API Definition

```
public native static int MV_CC_FeatureSave(
  Handle  handle,
  String  fileName
);
```

Parameters

handle

[IN] Device handle

fileName

[IN] Configuration file name

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void FeatureSave() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
      e.printStackTrace();
  }
  if (devicesList == null) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: OpenDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_FeatureSave(handle, "FeatureFile.ini");
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "Save Feature fail! nRet" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_FeatureLoad(handle, "FeatureFile.ini");
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "Load Feature fail! nRet" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "ClosDevice fail! nRet " + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "Destroy Handle fail! nRet " + nRet);
     return;
  }
}
```

3.3.7 MV_CC_FeatureLoad

Import configuration files to the camera.

API Definition

```
public native static int MV_CC_FeatureLoad(
  Handle     handle,
  String     fileName
);
```

Parameters

handle

[IN] Device handle

fileName

[IN] Configuration file name

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void FeatureLoad () {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
```

```
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: OpenDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_FeatureSave(handle, "FeatureFile.ini");
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "Save Feature fail! nRet" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_FeatureLoad(handle, "FeatureFile.ini");
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "Load Feature fail! nRet" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "ClosDevice fail! nRet " + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "Destroy Handle fail! nRet " + nRet);
      return;
  }
}
```

3.3.8 MV_CC_FileAccessRead

Read files from the camera.

API Definition

```
public native static int MV_CC_FileAccessRead(
  Handle              handle,
  MV_CC_FILE_ACCESS   fileAccess
);
```

Parameters

handle

[IN] Device handle

fileAccess

[IN] File access object, see ***MV_CC_FILE_ACCESS*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void FileAccessRead() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
     return;
  }
  MvCameraControlDefines.MV_CC_FILE_ACCESS file_access = new MvCameraControlDefines.MV_CC_FILE_ACCESS();
  file_access.userFileName = "UserSet1.txt";
  file_access.devFileName = "UserSet1";
  nRet = MvCameraControl.MV_CC_FileAccessRead(handle, file_access);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: FileAccessRead fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
```

```
    return;
  }
}
```

3.3.9 MV_CC_FileAccessWrite

Write files to the camera.

API Definition

```
public native static int MV_CC_FileAccessWrite(
  Handle              handle,
  MV_CC_FILE_ACCESS    fileAccess
);
```

Parameters

handle

[IN] Device handle,

fileAccess

[IN] File access object, see ***MV_CC_FILE_ACCESS*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void FileAccessWrite() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
    devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
    e.printStackTrace();
  }
  if (devicesList == null) {
    Log.e("CameraControl", "error: EnumDevices fail");
    return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
    handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
    if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
    }
  } catch (CameraControlException e) {
    e.printStackTrace();
```

```
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail");
        return;
    }
    MvCameraControlDefines.MV_CC_FILE_ACCESS file_access = new MvCameraControlDefines.MV_CC_FILE_ACCESS();
    file_access.userFileName = "UserSet1.txt";
    file_access.devFileName = "UserSet1";
    nRet = MvCameraControl.MV_CC_FileAccessWrite(handle, file_access);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: FileAccessRead fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
        return;
    }
}
```

3.3.10 MV_CC_GetFileAccessProgress

Get file access progress.

API Definition

```
public native static int MV_CC_GetFileAccessProgress(
  Handle                        handle,
  MV_CC_FILE_ACCESS_PROGRESS     fileAccessProgress
);
```

Parameters

handle

[IN] Device handle

fileAccessProgress

[IN] file access progress, see ***MV_CC_FILE_ACCESS_PROGRESS*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.11 MV_GIGE_ForceIp

Force changes of camera network parameters, including IP address, subnet mask, and default gateway.

API Definition

```
public native static MV_GIGE_ForceIp(
  Handle      handle,
  String      iP,
  String      subNetMask,
  String      defaultGateWay
);
```

Parameters

handle

[IN] Device handle

iP

[IN] IP address

subNetMask

[IN] Subnet mask

defaultGateWay

[IN] Default gateway

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void ForceIp() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
```

```
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }
  String nIP = "10.15.6.54";
  String nSubNetMask = "255.255.255.0";
  String nDefaultGateWay = "10.15.6.254";
  int nRet = MvCameraControl.MV_GIGE_ForceIp(handle, nIP, nSubNetMask, nDefaultGateWay);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: ForceIpEx fail" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: DestroyHandle fail " + nRet);
      return;
  }
}
```

3.3.12 MV_GIGE_SetIpConfig

Configure IP mode.

API Definition

```
public native static int MV_GIGE_SetIpConfig(
  Handle            handle,
  MVCC_IP_CONFIG     enType
);
```

Parameters

handle

[IN] Device handle

enType

[IN] IP mode, see ***MVCC_IP_CONFIG*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void SetIpConfig() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
      e.printStackTrace();
  }
  if (devicesList == null) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }
  int nRet = MvCameraControl.MV_GIGE_SetIpConfig(handle,
MvCameraControlDefines.MVCC_IP_CONFIG.MV_IP_CFG_LLA);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: SetIpConfig fail" + nRet);
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
      return;
  }
}
```

3.3.13 MV_GIGE_GetNetTransInfo

Get network transmission information, including received data size, and number of lost frames.

API Definition

```
public native static int MV_GIGE_GetNetTransInfo(
  Handle              handle,
  MV_NETTRANS_INFO    info
);
```

Parameters

handle

[IN] Device handle

pstInfo

[OUT] Network transmission information, including received data size, number of lost frames, and so on. See ***MV_NETTRANS_INFO*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void GetNetTransInfo() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: StartGrabbing fail");
     return;
  }
  MvCameraControlDefines.MVCC_INTVALUE intvalue = new MvCameraControlDefines.MVCC_INTVALUE();
  nRet = MvCameraControl.MV_CC_GetIntValue(handle, "PayloadSize", intvalue);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: PayloadSize fail");
     return;
  }
  byte[] datas = new byte[(int) intvalue.curValue + 2048];
  MvCameraControlDefines.MV_FRAME_OUT_INFO info = new MvCameraControlDefines.MV_FRAME_OUT_INFO();
  for (int i = 0; i < 50; i++) {
      nRet = MvCameraControl.MV_CC_GetOneFrameTimeout(handle, datas, info, 1000);
      if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: GetOneFrameTimeout fail");
    return;
      }
  }
  MvCameraControlDefines.MV_NETTRANS_INFO info1 = new MvCameraControlDefines.MV_NETTRANS_INFO();
  nRet = MvCameraControl.MV_GIGE_GetNetTransInfo(handle, info1);
  if (nRet == MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "GetDataSize" + info1.reviceDataSize);
  }
  nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: StopGrabbing fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: CloseDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.3.14 MV_CC_GetOptimalPacketSize

Get the optimal packet size.

API Definition

```
public native static int MV_CC_GetOptimalPacketSize(
  Handle    handle
);
```

Parameters

handle

[IN] Device handle

Return Value

If succeeded, the return value is larger than 0, which refers to the packet size; if failed, the return value is smaller than 0, which refers to the corresponding ***Error Code*** .

Remarks

The optimized packet size is the size of a packet transported via the network. For GigEVision camera, it is the size of SCPs; and for U3V camera, it is the size of packet read from drive each time. The API should be called after calling ***MV_CC_OpenDevice*** and before calling ***MV_CC_StartGrabbing*** .

Example

The following sample code is for reference only.

```
public void GetOptimalPacketSize() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
     return;
  }
  int nPacketSize = MvCameraControl.MV_CC_GetOptimalPacketSize(handle);
  if (nPacketSize > 0) {
     nRet = MvCameraControl.MV_CC_SetIntValue(handle, "GevSCPSPacketSize", nPacketSize);
     if (nRet != MvCameraControlDefines.MV_OK) {
   Log.e("CameraControl", "Warning: Set Packet Size fail nRet" + nRet);
   return;
     }
  } else {
     Log.e("CameraControl", "Warning: Get Packet Size fail nRet" + nPacketSize);
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail" + nRet);
     return;
  }
}
```

3.3.15 MV_GIGE_SetResend

Set whether to enable resending packets.

API Definition

```
public native static int MV_GIGE_SetResend(
  Handle        handle,
  int           enable
);
```

Parameters

handle

[IN] Device handle

enable

[IN] Enable resending packet or not: 0-disable, 1-enable

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.16 MV_GIGE_SetResend

Set parameters for resending packets.

API Definition

```
public native static int MV_GIGE_SetResend(
  Handle        handle,
  int           enable,
  int           maxResendPercent,
  int           resendTimeout
);
```

Parameters

handle

[IN] Device handle

enable

[IN] Enable resending packet or not: 0-Disable, 1-Enable

maxResendPercent

[IN] Maximum packet resending percentage, range: [0,100]

resendTimeout

[IN] Packet resending timeout, unit: ms

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.17 MV_GIGE_GetResendMaxRetryTimes

Get the maximum times one packet can be resent.

API Definition

```
public native static int MV_GIGE_GetResendMaxRetryTimes(
  Handle        handle,
  int           retryTimes
);
```

Parameters

handle

[IN] Device handle

retryTimes

[OUT] The maximum times one packet can be resent.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.18 MV_GIGE_SetResendMaxRetryTimes

Set the maximum times one packet can be resent.

API Definition

```
public native static int MV_GIGE_SetResendMaxRetryTimes(
  Handle        handle,
  int           retryTimes
);
```

Parameters

handle

[IN] Device handle

retryTimes

[IN] The maximum times one packet can be resent.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.19 MV_GIGE_GetResendTimeInterval

Get the packet resending interval.

API Definition

```
public native static int MV_GIGE_GetResendTimeInterval(
 Handle        handle,
 int           retryTimes
);
```

Parameters

handle

[IN] Device handle

retryTimes

[OUT] Packet resending interval

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.20 MV_GIGE_SetResendTimeInterval

Set the packet resending interval.

API Definition

```
public native static int MV_GIGE_SetResendTimeInterval(
 Handle        handle,
 int           retryTimes
);
```

Parameters

handle

[IN] Device handle

retryTimes

[IN] Packet resending interval, unit: millisecond

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.21 MV_GIGE_GetGvspTimeout

Get packet resending timeout.

API Definition

```
public native static int MV_GIGE_getGvspTimeout(
  Handle        handle,
  int           msec
);
```

Parameters

handle

[IN] Device handle

msec

[OUT] Packet resending timeout, unit: millisecond

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.22 MV_GIGE_SetGvspTimeout

Set packet resending timeout.

API Definition

```
public native static int MV_GIGE_SetGvspTimeout(
  Handle        handle,
  int           msec
);
```

Parameters

handle

[IN] Device handle

msec

[IN] Packet resending timeout, unit: millisecond

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.3.23 MV_XML_GetNodeInterfaceType

Get the current node type.

API Definition

```
public native static int MV_XML_GetNodeInterfaceType(
  Handle                  handle,
  String                  name,
  MV_XML_InterfaceType    interfaceType
);
```

Parameters

handle

[IN] Device handle

pstrName

[IN] Node name

pInterfaceType

[OUT] Node type, see ***MV_XML_InterfaceType*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

3.4 Image Acquisition APIs

3.4.1 MV_CC_RegisterImageCallBack

Register image data callback function, and getting chunk information is supported.

API Definition

```
public native static int MV_CC_RegisterImageCallBack(
  Handle          handle,
  ImageCallback   callback
);
```

Parameters

handle

[IN] Device handle

fOutputCallBack

[IN] Image data callback function, see ***CameraImageCallBack*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Remarks

- This API should be called after opening the camera.
- Two image acquisition modes are available, and these two modes cannot be used at same time:
  Mode 1: Call this API to set image callback function, and then call ***MV_CC_StartGrabbing*** to start the acquisition, The collected image will be returned in the configured callback function.
  Mode 2: Call ***MV_CC_StartGrabbing*** to start the acquisition, and then repeatedly call ***MV_CC_GetOneFrameTimeout*** to get the frame data with specific pixel format in the application layer.
  When getting frame data, the upper-layer program should control the frequency of calling this API according to the frame rate.

Example

The following sample code is for reference only.

```
public void RegisterImageCallBack() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
   Log.e("CameraControl", "error: CreateHandle fail");
   return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }

  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
```

```
        return;
    }

    nRet = MvCameraControl.MV_CC_RegisterImageCallBack(handle, new CameraImageCallBack() {
        @Override
        public int OnImageCallBack(byte[] bytes, MvCameraControlDefines.MV_FRAME_OUT_INFO info) {
      Log.e("CameraControl", "width:" + info.width + "height:" + info.height + "");
      return 0;
        }
    });
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: RegisterImageCallBack fail");
        return;
    }

    nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StartGrabbing fail");
        return;
    }

    try {
        Thread.sleep(3000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StopGrabbing fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.4.2 MV_CC_StartGrabbing

Start image acquisition.

API Definition

```
public native static int MV_CC_StartGrabbing(
  Handle     handle
);
```

Parameters

handle

[IN] Device handle

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void StartGrabbing() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
      e.printStackTrace();
  }
  if (devicesList == null) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }

  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: OpenDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: StartGrabbing fail");
      return;
  }


  try {
      Thread.sleep(3000);
  } catch (InterruptedException e) {
      e.printStackTrace();
  }
  nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: StopGrabbing fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.4.3 MV_CC_GetOneFrameTimeout

Get a frame.

API Definition

```
public native static int MV_CC_GetOneFrameTimeout(
  Handle              handle,
  byte[]              data,
  MV_FRAME_OUT_INFO   frameInfo,
  int                 msec
);
```

Parameters

handle

[IN] Device handle

data

[OUT] Buffer address used to save the frame

frameInfo

[OUT] Obtained frame information, see ***MV_FRAME_OUT_INFO*** for details.

msec

[IN] Waiting timeout, which is 1000 by default, unit: millisecond

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Remarks

Before calling this API, you must call ***MV_CC_StartGrabbing*** to start image acquisition. This API is repeatedly called to get frame data, so the upper-layer program should control the frequency of calling this API according to the frame rate. This API supports setting timeout, which can improve the stationarity of getting stream. If timed out, this API will not return until the data is obtained..

Example

The following sample code is for reference only.

```
public void GetOneFrameTimeout() {
 ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
 try {
    devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
 } catch (CameraControlException e) {
    e.printStackTrace();
 }
 if (devicesList == null) {
    Log.e("CameraControl", "error: EnumDevices fail");
    return;
 }
 MvCameraControlDefines.Handle handle = null;
 MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
 try {
    handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
    if (handle == null) {
  Log.e("CameraControl", "error: CreateHandle fail");
  return;
    }
 } catch (CameraControlException e) {
    e.printStackTrace();
    return;
 }
 int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
 if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: OpenDevice fail");
    return;
 }
 nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
 if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: StartGrabbing fail");
    return;
 }
 MvCameraControlDefines.MVCC_INTVALUE intvalue = new MvCameraControlDefines.MVCC_INTVALUE();
 nRet = MvCameraControl.MV_CC_GetIntValue(handle, "PayloadSize", intvalue);
 if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: PayloadSize fail");
    return;
 }
 byte[] datas = new byte[(int) intvalue.curValue + 2048];
 MvCameraControlWrapper.MvCameraControlDefines.MV_FRAME_OUT_INFO info = new
MvCameraControlDefines.MV_FRAME_OUT_INFO();
 for (int i = 0; i < 50; i++) {
    nRet = MvCameraControl.MV_CC_GetOneFrameTimeout(handle, datas, info, 1000);
    if (nRet != MvCameraControlDefines.MV_OK) {
  Log.e("CameraControl", "error: GetOneFrameTimeout fail");
```

```
      return;
      }
  }
  nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: StopGrabbing fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: CloseDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.4.4 MV_CC_GetBitmapTimeout

Get data of a single frame in Bitmap format.

API Definition

```
public native static int MV_CC_GetBitmapTimeout(
  Handle              handle,
  byte[]              data,
  MV_FRAME_OUT_INFO   frameInfo,
  int                 msec
);
```

Parameters

handle

[IN] Device handle

data

[OUT] Buffer address used to save image data

frameInfo

[OUT] Obtained frame information, see ***MV_FRAME_OUT_INFO*** for details.

msec

[IN] Waiting timeout, unit: millisecond

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void GetBitmapTimeout() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
      devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
      e.printStackTrace();
  }
  if (devicesList == null) {
      Log.e("CameraControl", "error: EnumDevices fail");
      return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
      handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
      if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
      }
  } catch (CameraControlException e) {
      e.printStackTrace();
      return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: OpenDevice fail");
      return;
  }
  nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: StartGrabbing fail");
      return;
  }
  MvCameraControlDefines.MVCC_INTVALUE intvalue = new MvCameraControlDefines.MVCC_INTVALUE();
  nRet = MvCameraControl.MV_CC_GetIntValue(handle, "PayloadSize", intvalue);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: PayloadSize fail");
      return;
  }
  byte[] datas = new byte[(int) intvalue.curValue + 2048];
  MvCameraControlWrapper.MvCameraControlDefines.MV_FRAME_OUT_INFO info = new
MvCameraControlDefines.MV_FRAME_OUT_INFO();
  for (int i = 0; i < 50; i++) {
      nRet = MvCameraControl.MV_CC_GetBitmapTimeout(handle, datas, info, 1000);
      if (nRet != MvCameraControlDefines.MV_OK) {
    Log.e("CameraControl", "error: GetOneFrameTimeout fail");
    return;
      }
  }
  nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
      Log.e("CameraControl", "error: StopGrabbing fail");
```

```
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.4.5 MV_CC_StopGrabbing

Stop image acquisition.

API Definition

```
public native static int MV_CC_StopGrabbing(
  Handle     handle
);
```

Parameters

handle

[IN] Device handle

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

```
public void StopGrabbing() {
    ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
    try {
        devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
    } catch (CameraControlException e) {
        e.printStackTrace();
    }
    if (devicesList == null) {
        Log.e("CameraControl", "error: EnumDevices fail");
        return;
    }
    MvCameraControlDefines.Handle handle = null;
    MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
    try {
        handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
        if (handle == null) {
      Log.e("CameraControl", "error: CreateHandle fail");
      return;
        }
    } catch (CameraControlException e) {
        e.printStackTrace();
        return;
    }
    int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: OpenDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StartGrabbing fail");
        return;
    }
    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StopGrabbing fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail");
        return;
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
}
```

3.4.6 MV_CC_ConvertPixelType

Convert pixel format.

API Definition

```
public native static int MV_CC_ConvertPixelType(
  Handle                      handle,
  MV_CC_PIXEL_CONVERT_PARAM      cvtParam
);
```

Parameters

handle

[IN] Device handle

pstCvtParam

[IN] Object about image conversion parameters, see ***MV_CC_PIXEL_CONVERT_PARAM*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void ConvertPixelType() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: StartGrabbing fail" + nRet);
     return;
  }
  MvCameraControlDefines.MVCC_INTVALUE stIntvalue = new MvCameraControlDefines.MVCC_INTVALUE();
  nRet = MvCameraControl.MV_CC_GetIntValue(handle, "PayloadSize", stIntvalue);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "Get PayloadSize failed! nRet" + nRet);
     return;
  }
  int nBufSize = (int) stIntvalue.curValue + 2048; //One frame data size + reserved bytes (handled in SDK)
  byte[] datas = new byte[nBufSize];
  MvCameraControlDefines.MV_FRAME_OUT_INFO info = new MvCameraControlDefines.MV_FRAME_OUT_INFO();
  nRet = MvCameraControl.MV_CC_GetOneFrameTimeout(handle, datas, info, 1000);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "Get GetOneFrameTimeout failed! nRet" + nRet);
     return;
  }
  MvCameraControlWrapper.MvCameraControlDefines.MV_CC_PIXEL_CONVERT_PARAM convert_param = new
MvCameraControlDefines.MV_CC_PIXEL_CONVERT_PARAM();
  convert_param.srcData = datas;
  convert_param.srcDataLen = info.frameLen;
  convert_param.srcPixelType = info.pixelType;
  convert_param.width = info.width;
  convert_param.height = info.height;
  convert_param.dstPixelType = PixelType_Gvsp_RGB8_Packed;
  nRet = MvCameraControl.MV_CC_ConvertPixelType(handle, convert_param);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "Get ConvertPixelType failed! nRet" + nRet);
     return;
  }
  nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: StopGrabbing fail" + nRet);
  }
  nRet = MvCameraControl.MV_CC_CloseDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: CloseDevice fail" + nRet);
  }
  nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: DestroyHandle fail " + nRet);
  }
}
```

3.4.7 MV_CC_SaveImage

Save images.

API Definition

```
public native static int MV_CC_SaveImage(
  MV_SAVE_IMAGE_PARAM    saveParam
);
```

Parameters

saveParam

[IN] Input and output parameters of saving picture data, see ***MV_SAVE_IMAGE_PARAM*** for details.

Return Value

Return *MV_OK* for success, and return ***Error Code*** for failure.

Example

The following sample code is for reference only.

```
public void SaveImage() {
  ArrayList<MvCameraControlDefines.MV_CC_DEVICE_INFO> devicesList = null;
  try {
     devicesList = MvCameraControl.MV_CC_EnumDevices(MV_GIGE_DEVICE | MV_USB_DEVICE);
  } catch (CameraControlException e) {
     e.printStackTrace();
  }
  if (devicesList == null) {
     Log.e("CameraControl", "error: EnumDevices fail");
     return;
  }
  MvCameraControlDefines.Handle handle = null;
  MvCameraControlDefines.MV_CC_DEVICE_INFO stDevInfo = devicesList.get(0);
  try {
     handle = MvCameraControl.MV_CC_CreateHandle(stDevInfo);
     if (handle == null) {
    Log.e("CameraControl", "error: CreateHandle fail");
    return;
     }
  } catch (CameraControlException e) {
     e.printStackTrace();
     return;
  }
  int nRet = MvCameraControl.MV_CC_OpenDevice(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: OpenDevice fail");
     return;
  }
  nRet = MvCameraControl.MV_CC_StartGrabbing(handle);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "error: StartGrabbing fail" + nRet);
     return;
  }
  MvCameraControlDefines.MVCC_INTVALUE stIntvalue = new MvCameraControlDefines.MVCC_INTVALUE();
  nRet = MvCameraControl.MV_CC_GetIntValue(handle, "PayloadSize", stIntvalue);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "Get PayloadSize failed! nRet" + nRet);
     return;
  }
  int nBufSize = (int) stIntvalue.curValue + 2048; //One frame data size + reserved bytes (handled in SDK)
  byte[] datas = new byte[nBufSize];
  MvCameraControlDefines.MV_FRAME_OUT_INFO info = new MvCameraControlDefines.MV_FRAME_OUT_INFO();
  nRet = MvCameraControl.MV_CC_GetOneFrameTimeout(handle, datas, info, 1000);
  if (nRet != MvCameraControlDefines.MV_OK) {
     Log.e("CameraControl", "Get GetOneFrameTimeout failed! nRet" + nRet);
```

```
        return;
    }
    MvCameraControlDefines.MV_SAVE_IMAGE_PARAM param = new
MvCameraControlDefines.MV_SAVE_IMAGE_PARAM();
    param.data = datas;
    param.dataLen = info.frameLen;
    param.pixelType = info.pixelType;
    param.width = info.width;
    param.height = info.height;
    param.imageType = MvCameraControlDefines.MV_SAVE_IAMGE_TYPE.MV_Image_Jpeg;
    nRet = MvCameraControl.MV_CC_SaveImage(handle,param);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: SaveImage fail" + nRet);
    }
    nRet = MvCameraControl.MV_CC_StopGrabbing(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: StopGrabbing fail" + nRet);
    }
    nRet = MvCameraControl.MV_CC_CloseDevice(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: CloseDevice fail" + nRet);
    }
    nRet = MvCameraControl.MV_CC_DestroyHandle(handle);
    if (nRet != MvCameraControlDefines.MV_OK) {
        Log.e("CameraControl", "error: DestroyHandle fail " + nRet);
    }
}
```

Chapter 4 Callback Function

Enter a short description of your concept here (optional).

This is the start of your concept.

4.1 CameraImageCallBack

Stream callback function.

Callback Function Definition

```
public interface CameraImageCallBack{
 int OnImageCallBack(
  byte[]              datas,
  MV_FRAME_OUT_INFO     info);
}
```

Parameters

datas

Image data

info

The output frame information, see ***MV_FRAME_OUT_INFO*** for details.

4.2 CameraExceptionCallBack

Exception callback function.

Callback Function Definition

```
public interface CameraExceptionCallBack{
int OnExceptionCallBack(
  int     msgType);
}
```

Parameters

msgType

Exception information

4.3 CameraEventCallBack

Event callback function.

Callback Function Definition

```
public interface CameraEventCallBack{
int OnEventCallBack(
   MV_EVENT_OUT_INFO     eventInfo);
}
```

Parameters

eventInfo

Event information, see ***MV_EVENT_OUT_INFO*** for details.

4.4 CameraControlException

Callback function of operation exception.

Callback Function Definition

```
public static class CameraControlException extends Exception{
  public  int         errorCode;
  public  String      errorInfo;
  public CameraControlException(){}
 }
```

Parameters

errorCode

Error code, see ***Error Code*** for details.

errorInfo

Error code information, see ***Error Code*** for details.

Chapter 5 Data Structure and Enumeration

5.1 Data Structure

5.1.1 MV_ALL_MATCH_INFO

Object About Different Matching Type Information

Member	Data Type	Description
type	int	Information type to output
matchInfoNetDetect	***MV_MATCH_INFO_NET_DETECT***	Network traffic and packet loss information
matchInfoUsbDetect	***MV_MATCH_INFO_USB_DETECT***	Total number of bytes that the host received from USB3 Vision cameras

5.1.2 MV_CC_DEVICE_INFO

Object About Device Information

Member	Data Type	Description
majorVer	short	Major version No.
minorVer	short	Minor version No.
macAddrHigh	int	High MAC address
macAddrLow	int	Low MAC address
tLayerType	int	Transport layer type
gigEInfo	***MV_GIGE_DEVICE_INFO***	GigE device information
usb3VInfo	***MV_USB3_DEVICE_INFO***	USB device information

5.1.3 MV_CC_FILE_ACCESS

Object About File Access Information

Member	Data Type	Description
userFileName	String	User file name
devFileName	String	Device file name

5.1.4 MV_CC_FILE_ACCESS_PROGRESS

Object About File Access Progress

Member	Data Type	Description
completed	long	Completed size
total	long	Total size

5.1.5 MV_CC_PIXEL_CONVERT_PARAM

Object About Image Conversion Parameters

Member	Data Type	Description
width	short	Image width
height	short	Image height
srcPixelType	***MvGvspPixelType***	Original pixel format
srcData	byte[]	Buffer of input data
srcDataLen	int	Size of input data
dstPixelType	***MvGvspPixelType***	Target pixel format
dstBuffer	byte[]	High timestamp
dstLen	int	Low timestamp

5.1.6 MV_EVENT_OUT_INFO

Object About Event Information

Member	Data Type	Description
eventName	String	Event name
eventID	short	Event ID
streamChannel	short	Stream channel ID
blockIdHigh	int	High frame No.
blockIdLow	int	Low frame No.
timestampHigh	int	High timestamp
timestampLow	int	Low timestamp
eventData	byte[]	Event data
eventDataSize	int	Event data size

5.1.7 MV_FRAME_OUT_INFO

Object About Output Frame Information

Definition	type	Meaning
width	short	Image width
height	short	Image height
pixelType	int	Pixel format
frameNum	int	Frame No.
devTimeStampHigh	long	High 32-bits of timestamp generated by the camera
devTimeStampLow	long	Low 32-bits timestamp generated by the camera
hostTimeStamp	long	Timestamp generated by the host
frameLen	int	Frame size
secondCount	int	Number of seconds, which increases by per second.
cycleCount	int	Number of clock periods, which increases by per 125 us, and is reset in per second.

Definition	type	Meaning
cycleOffset	int	Clock period offset, it will be reset in every 125 us.
gain	float	Sharpness
exposureTime	float	Exposure time
averageBrightness	int	Average brightness
red	int	WB red
green	int	WB green
nBlue	int	WB blue
frameCounter	int	The number of frames
triggerIndex	int	Triggering times
input	int	Line input
output	int	Line output
offsetX	short	X-offset of the ROI region
offsetY	short	Y-offset of the ROI region
chunkWidth	short	Width of the ROI region
chunkHeight	short	Height of the ROI region
lostPacket	int	The number of lost packets in this frame

5.1.8 MV_GIGE_DEVICE_INFO

Object About GigE Device Information

Member	Data Type	Description
ipCfgOption	String	IP configuration parameters
ipCfgCurrent	String	Current IP configuration
currentIp	String	Current device IP address
currentSubNetMask	String	Current subnet mask
defultGateWay	String	Default gateway
manufacturerName	String	Manufacturer name
modelName	String	Model name

Member	Data Type	Description
deviceVersion	String	Device version No.
manufacturerSpecificInfo	String	Manufacturer information
serialNumber	String	Serial No.
userDefinedName	String	Custom name
netExport	String	Network port's IP address

5.1.9 MV_MATCH_INFO_NET_DETECT

Object About Network Flow and Packet Loss Information

Member	Data Type	Description
reviceDataSize	long	Received data size
lostPacketCount	long	The number of lost packets
lostFrameCount	long	The number of lost frames
netRecvFrameCount	long	Reserved.
requestResendPacketCount	long	The number of packets requested for resending
resendPacketCount	long	The number of resent packets

5.1.10 MV_MATCH_INFO_USB_DETECT

Object About the Number of Bytes the Host Received from USB3 Vision Cameras

Member	Data Type	Description
reviceDataSize	long	Received data size
revicedFrameCount	long	The number of received frames
errorFrameCount	long	The number of error frames

5.1.11 MV_NETTRANS_INFO

Object About Network Transport Information

Member	Data Type	Description
reviceDataSize	int	Received data size
throwFrameCount	int	The number of lost frames
netRecvFrameCount	int	Reserved.
requestResendPacketCount	int	The number of packets which request for resending
resendPacketCount	int	The number of resent packets

5.1.12 MV_SAVE_IMAGE_PARAM

Object About Parameters of Image Format Conversion

Member	Data Type	Description
data	byte[]	Original image data
dataLen	int	Size of the original image data
pixelType	***MvGvspPixelType***	Pixel format of the original image data
width	short	Image width
height	shorts	Image height
imageBuffer	byte[]	Output data buffer, used for storing converted pictures
imageLen	int	Size of the output image
imageType	***MV_SAVE_IAMGE_TYPE***	Format of the output image

5.1.13 MV_USB3_DEVICE_INFO

Object About USB3 Device Information

Member	Data Type	Description
crtlInEndPoint	char	Control input endpoint
crtlOutEndPoint	char	Control output endpoint

Member	Data Type	Description
streamEndPoint	char	Stream endpoint
eventEndPoint	char	Event endpoint
idVendor	short	Supplier ID
idProduct	short	Device ID
deviceNumber	int	Device serial No.
deviceGUID	String	Device GUID
vendorName	String	Supplier name
modelName	String	Model
familyName	String	Family name
deviceVersion	String	Device version
manufacturerName	String	Manufacturer name
serialNumber	String	Serial No.
userDefinedName	String	Custom name
bcdUSB	int	Supported USB protocol

5.1.14 MVCC_ENUMVALUE

Object About Parameter Values of Type enum

Member	Data Type	Description
curValue	int	Current value
supportValue	ArrayList<Integer>	Supported parameter list

5.1.15 MVCC_FLOATVALUE

Object About Parameter Values of Type float

Member	Data Type	Description
curValue	float	Current value
max	float	The maximum value
min	float	The minimum value

5.1.16 MVCC_INTVALUE

Object About Parameter Values of Type int

Member	Data Type	Description
curValue	long	Current value
max	long	The maximum value
min	long	The minimum value
inc	long	Increment

5.1.17 MVCC_STRINGVALUE

Object About Parameter Values of Type String

Member	Data Type	Description
curValue	String	Current value

5.2 Enumeration

5.2.1 MvGvspPixelType

Enumerate GigE protocol pixel types

Enumeration Definition

```
public enum MvGvspPixelType {
  PixelType_Gvsp_Undefined(-1),
  // Mono buffer format defines
  PixelType_Gvsp_Mono1p(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(1) | 0x0037),
  PixelType_Gvsp_Mono2p(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(2) | 0x0038),
  PixelType_Gvsp_Mono4p(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(4) | 0x0039),
  PixelType_Gvsp_Mono8(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(8) | 0x0001),
  PixelType_Gvsp_Mono8_Signed(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(8) | 0x0002),
  PixelType_Gvsp_Mono10(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0003),
  PixelType_Gvsp_Mono10_Packed(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x0004),
  PixelType_Gvsp_Mono12(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0005),
  PixelType_Gvsp_Mono12_Packed(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x0006),
  PixelType_Gvsp_Mono14(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0025),
  PixelType_Gvsp_Mono16(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0007),
  // Bayer buffer format defines
```

```
PixelType_Gvsp_BayerGR8(MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(8)
  | 0x0008), PixelType_Gvsp_BayerRG8(MV_GVSP_PIX_MONO
  | MV_PIXEL_BIT_COUNT(8) | 0x0009), PixelType_Gvsp_BayerGB8(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(8) | 0x000A), PixelType_Gvsp_BayerBG8(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(8) | 0x000B), PixelType_Gvsp_BayerGR10(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x000C), PixelType_Gvsp_BayerRG10(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x000D), PixelType_Gvsp_BayerGB10(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x000E), PixelType_Gvsp_BayerBG10(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x000F), PixelType_Gvsp_BayerGR12(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0010), PixelType_Gvsp_BayerRG12(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0011), PixelType_Gvsp_BayerGB12(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0012), PixelType_Gvsp_BayerBG12(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0013), PixelType_Gvsp_BayerGR10_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x0026), PixelType_Gvsp_BayerRG10_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x0027), PixelType_Gvsp_BayerGB10_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x0028), PixelType_Gvsp_BayerBG10_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x0029), PixelType_Gvsp_BayerGR12_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x002A), PixelType_Gvsp_BayerRG12_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x002B), PixelType_Gvsp_BayerGB12_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x002C), PixelType_Gvsp_BayerBG12_Packed(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(12) | 0x002D), PixelType_Gvsp_BayerGR16(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x002E), PixelType_Gvsp_BayerRG16(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x002F), PixelType_Gvsp_BayerGB16(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0030), PixelType_Gvsp_BayerBG16(
  MV_GVSP_PIX_MONO | MV_PIXEL_BIT_COUNT(16) | 0x0031),
// RGB Packed buffer format defines
PixelType_Gvsp_RGB8_Packed(MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(24)
  | 0x0014), PixelType_Gvsp_BGR8_Packed(MV_GVSP_PIX_COLOR
  | MV_PIXEL_BIT_COUNT(24) | 0x0015), PixelType_Gvsp_RGBA8_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(32) | 0x0016), PixelType_Gvsp_BGRA8_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(32) | 0x0017), PixelType_Gvsp_RGB10_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(48) | 0x0018), PixelType_Gvsp_BGR10_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(48) | 0x0019), PixelType_Gvsp_RGB12_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(48) | 0x001A), PixelType_Gvsp_BGR12_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(48) | 0x001B), PixelType_Gvsp_RGB16_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(48) | 0x0033), PixelType_Gvsp_RGB10V1_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(32) | 0x001C), PixelType_Gvsp_RGB10V2_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(32) | 0x001D), PixelType_Gvsp_RGB12V1_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(36) | 0X0034), PixelType_Gvsp_RGB565_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x0035), PixelType_Gvsp_BGR565_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0X0036),
// YUV Packed buffer format defines
PixelType_Gvsp_YUV411_Packed(MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(12)
  | 0x001E), PixelType_Gvsp_YUV422_Packed(MV_GVSP_PIX_COLOR
  | MV_PIXEL_BIT_COUNT(16) | 0x001F), PixelType_Gvsp_YUV422_YUYV_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x0032), PixelType_Gvsp_YUV444_Packed(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(24) | 0x0020), PixelType_Gvsp_YCBCR8_CBYCR(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(24) | 0x003A), PixelType_Gvsp_YCBCR422_8(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x003B), PixelType_Gvsp_YCBCR422_8_CBYCRY(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x0043), PixelType_Gvsp_YCBCR411_8_CBYYCRYY(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(12) | 0x003C), PixelType_Gvsp_YCBCR601_8_CBYCR(
  MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(24) | 0x003D), PixelType_Gvsp_YCBCR601_422_8(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x003E), PixelType_Gvsp_YCBCR601_422_8_CBYCRY(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x0044), PixelType_Gvsp_YCBCR601_411_8_CBYYCRYY(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(12) | 0x003F), PixelType_Gvsp_YCBCR709_8_CBYCR(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(24) | 0x0040), PixelType_Gvsp_YCBCR709_422_8(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x0041), PixelType_Gvsp_YCBCR709_422_8_CBYCRY(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(16) | 0x0045), PixelType_Gvsp_YCBCR709_411_8_CBYYCRYY(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(12) | 0x0042),
  // RGB Planar buffer format defines
  PixelType_Gvsp_RGB8_Planar(MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(24)
    | 0x0021), PixelType_Gvsp_RGB10_Planar(MV_GVSP_PIX_COLOR
    | MV_PIXEL_BIT_COUNT(48) | 0x0022), PixelType_Gvsp_RGB12_Planar(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(48) | 0x0023), PixelType_Gvsp_RGB16_Planar(
    MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(48) | 0x0024),
  // Custom image format
  PixelType_Gvsp_Jpeg(MV_GVSP_PIX_CUSTOM | MV_PIXEL_BIT_COUNT(24)| 0x0001),
  PixelType_Gvsp_Coord3D_ABC32f(MV_GVSP_PIX_COLOR| MV_PIXEL_BIT_COUNT(96) | 0x00C0),
  PixelType_Gvsp_Coord3D_ABC32f_Planar(MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(96) | 0x00C1),
  PixelType_Gvsp_Coord3D_AC32f(MV_GVSP_PIX_COLOR | MV_PIXEL_BIT_COUNT(40)| 0x00C2), // 0x024000C2, /*
3D coordinate A-C 32-bit floating
    // point */
  PixelType_Gvsp_COORD3D_DEPTH_PLUS_MASK(0x82000000 | MV_PIXEL_BIT_COUNT(28) | 0x0001);// 0x82280001
  public int getnValue() {
    return nValue;
  }
  private int nValue;
  private MvGvspPixelType(int nValue) {
    this.nValue = nValue;
  }
}
```

5.2.2 MVCC_IP_CONFIG

Enumerate about GigEVision IP configuration.

Enumeration Definition

```
public enum MVCC_IP_CONFIG {
  // GigEVision IP Configuration
  MV_IP_CFG_STATIC(0x05000000),
  MV_IP_CFG_DHCP(0x06000000),
  MV_IP_CFG_LLA(0x04000000);
  public int getnValue() {
    return nValue;
  }
  private int nValue;
  private MVCC_IP_CONFIG(int nValue) {
  this.nValue = nValue;
  }
}
```

5.2.3 MV_XML_InterfaceType

Enumerate interface types, to which each node corresponds.

Enumeration Definition

```
public enum MV_XML_InterfaceType{
  IFT_IValue,
  IFT_IBase,
  IFT_IInteger,
  IFT_IBoolean,
  IFT_ICommand,
  IFT_IFloat,
  IFT_IString,
  IFT_IRegister,
  IFT_ICategory,
  IFT_IEnumeration,
  IFT_IEnumEntry,
  IFT_IPort
};
```

5.2.4 MV_SAVE_IAMGE_TYPE

Enumerate picture formats.

Enumeration Definition

```
public enum MV_SAVE_IAMGE_TYPE{
  MV_Image_Bmp(1),
  MV_Image_Jpeg(2),
  MV_Image_Png(3),
  MV_Image_Tif(4);
  private int nValue;
  private MV_SAVE_IAMGE_TYPE(int nValue){
    this.nValue = nValue;
  }
  public int getnValue() {
    return nValue;
  }
}
```

Chapter 6 FAQ (Frequently Asked Question)

6.1 Why is there no image when the USB3Vision camera starts getting stream?

Cause

The camera lost frames due to insufficient USB buffer of the Android device, or the camera's resolution was too large to get stream.

Solution 1

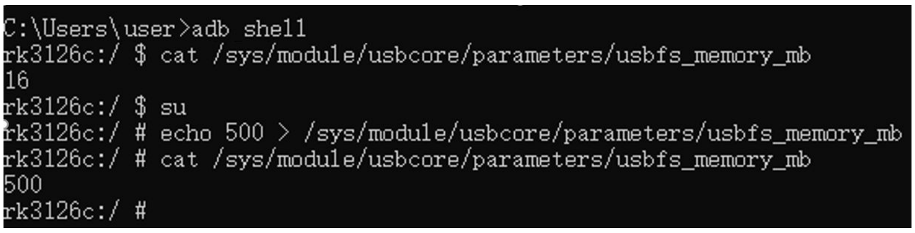
Edit the USB buffer of the Android device via ADB (Android Debug Bridge) as below.

1. Get the current USB buffer.

```
adb shell
cat /sys/module/usbcore/parameters/usbfs_memory_mb
```

2. Set USB buffer.

```
adb
shellsu
echo 500 > /sys/module/usbcore/parameters/usbfs_memory_mb
```

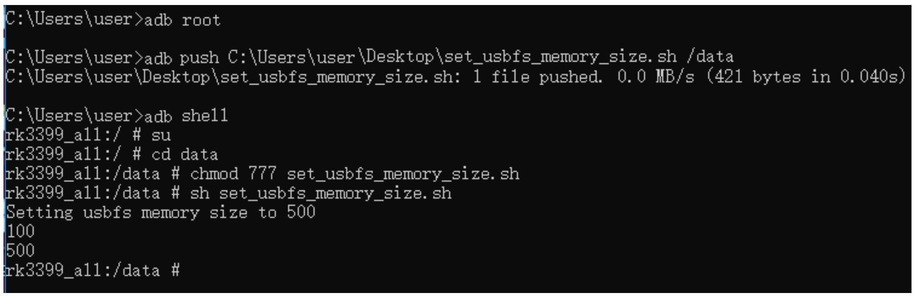


```
C:\Users\user>adb shell
rk3126c:/ $ cat /sys/module/usbcore/parameters/usbfs_memory_mb
16
rk3126c:/ $ su
rk3126c:/ # echo 500 > /sys/module/usbcore/parameters/usbfs_memory_mb
rk3126c:/ # cat /sys/module/usbcore/parameters/usbfs_memory_mb
500
rk3126c:/ #
```

Solution 2

Edit the USB buffer of the Android device via ADB (Android Debug Bridge) as below.

```
adb root
adb push pc(file path) /data/
adb shell
su
cd data/
chmod 777 set_usbfs_memory_size.sh
sh set_usbfs_memory_size.sh
```



```
C:\Users\user>adb root

C:\Users\user>adb push C:\Users\user\Desktop\set_usbfs_memory_size.sh /data
C:\Users\user\Desktop\set_usbfs_memory_size.sh: 1 file pushed. 0.0 MB/s (421 bytes in 0.040s)

C:\Users\user>adb shell
rk3399_all:/ # su
rk3399_all:/ # cd data
rk3399_all:/data # chmod 777 set_usbfs_memory_size.sh
rk3399_all:/data # sh set_usbfs_memory_size.sh
Setting usbfs memory size to 500
100
500
rk3399_all:/data #
```

6.2 Why is the image abnormal or system stuck when the GigE camera starts getting stream?

Cause

The Android device does not support jumbo frame.

Solution

Call the API ***MV_CC_SetIntValue*** to set the parameter **GevSCPSPacketSize** to "1500".

```
int nPacketSize = MvCameraControl.MV_CC_GetOptimalPacketSize(handle);
if (nPacketSize > 0) {
  nRet = MvCameraControl.MV_CC_SetIntValue(handle, "GevSCPSPacketSize", nPacketSize);
   if (nRet != MV_OK) {
     showLog("Warning: Set Packet Size fail nRet" + Integer.toHexString(nRet));
      return;
    }
} else {
    showLog("Warning: Get Packet Size fail nRet" + Integer.toHexString(nPacketSize));
}
```

6.3 Why does the frame loss occur when multiple GigE cameras are connected or the GigE camera's resolution is too high?

Cause

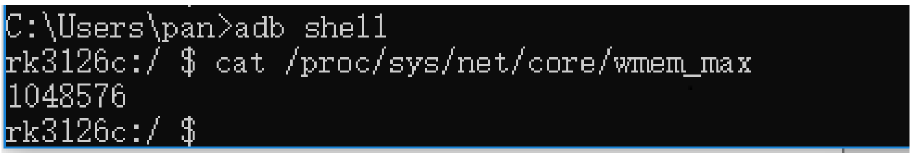
The socket buffer space is insufficient.

Solution 1

Expand the socket buffer space via ADB (Android Debug Bridge) as below.

1. Get the current socket buffer space.

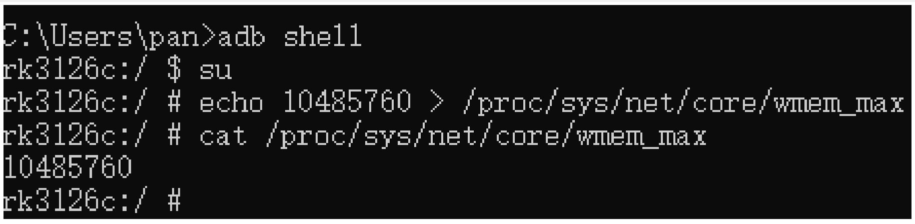
```
adb shell
cat /proc/sys/net/core/wmem_max
```



```
C:\Users\pan>adb shell
rk3126c:/ $ cat /proc/sys/net/core/wmem_max
1048576
rk3126c:/ $
```

2. Expand the socket buffer space.

```
adb shell
su
echo 10485760 > /proc/sys/net/core/wmem_max
```

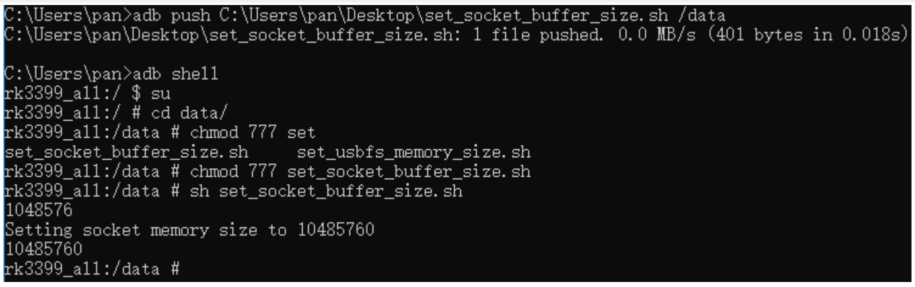


```
C:\Users\pan>adb shell
rk3126c:/ $ su
rk3126c:/ # echo 10485760 > /proc/sys/net/core/wmem_max
rk3126c:/ # cat /proc/sys/net/core/wmem_max
10485760
rk3126c:/ #
```

Solution 2

Expand the socket buffer space via ADB (Android Debug Bridge) as below.

```
adb push pc(file path) /data
adb shell
su
cd data/
chmod 777 set_socket_buffer_size.sh
sh set_socket_buffer_size.sh
```



```
C:\Users\pan>adb push C:\Users\pan\Desktop\set_socket_buffer_size.sh /data
C:\Users\pan\Desktop\set_socket_buffer_size.sh: 1 file pushed. 0.0 MB/s (401 bytes in 0.018s)

C:\Users\pan>adb shell
rk3399_all:/ $ su
rk3399_all:/ # cd data/
rk3399_all:/data # chmod 777 set
set_socket_buffer_size.sh     set_usbfs_memory_size.sh
rk3399_all:/data # chmod 777 set_socket_buffer_size.sh
rk3399_all:/data # sh set_socket_buffer_size.sh
1048576
Setting socket memory size to 10485760
10485760
rk3399_all:/data #
```

6.4 Why can't Android devices enumerate USB3Vision cameras?

Cause

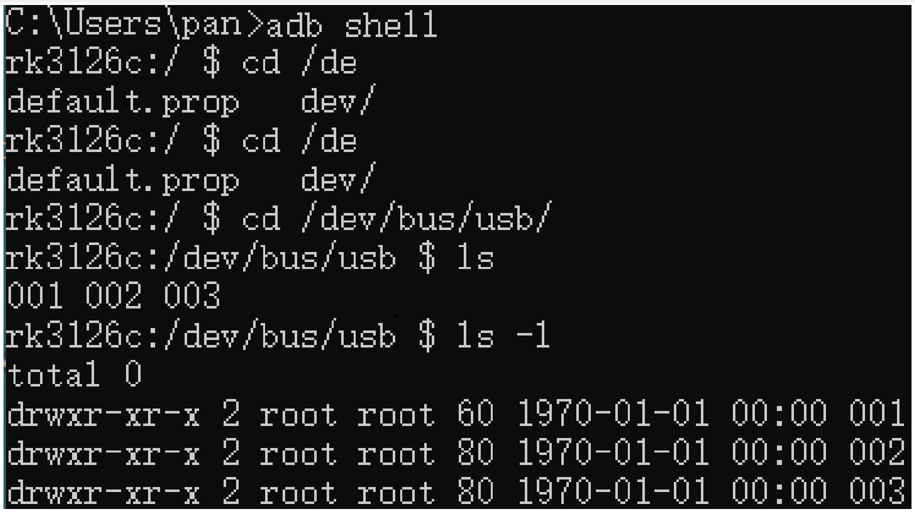
The USB port of the Android device does not have access permissions to connect the USB3Vision cameras.

Solution

Edit the permission of the USB port of the Android device via ADB (Android Debug Bridge) as below.

1. Get the current permissions of the USB port.

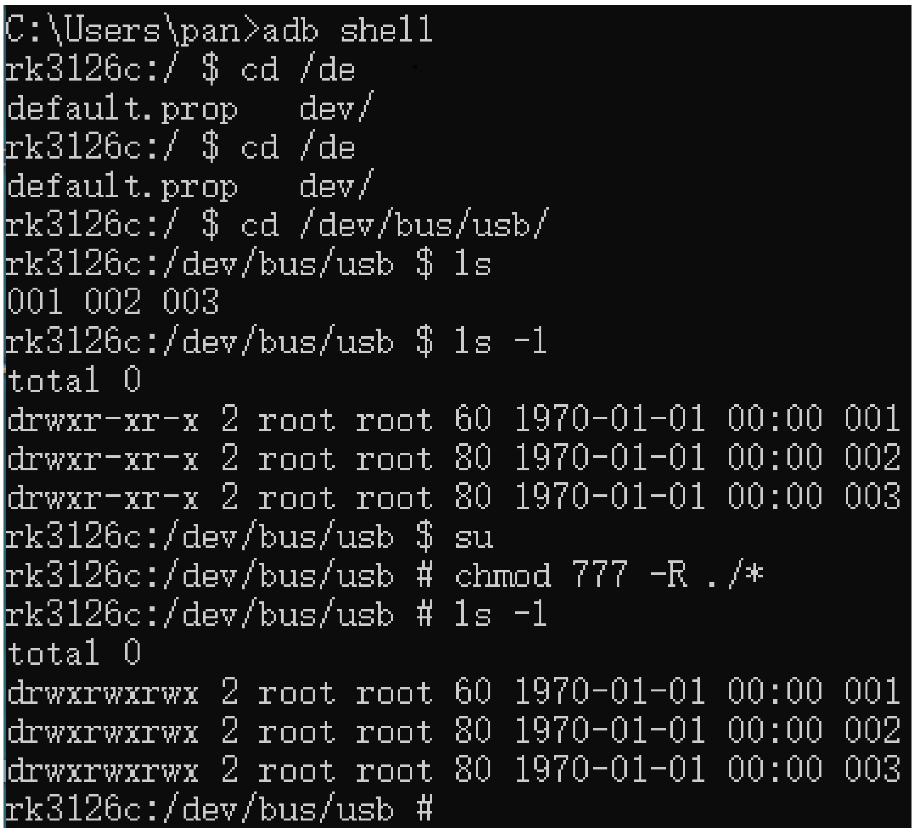
```
adb shell
cd /dev/bus/usb
ls –l
```



```
C:\Users\pan>adb shell
rk3126c:/ $ cd /de
default.prop   dev/
rk3126c:/ $ cd /de
default.prop   dev/
rk3126c:/ $ cd /dev/bus/usb/
rk3126c:/dev/bus/usb $ ls
001 002 003
rk3126c:/dev/bus/usb $ ls -l
total 0
drwxr-xr-x 2 root root 60 1970-01-01 00:00 001
drwxr-xr-x 2 root root 80 1970-01-01 00:00 002
drwxr-xr-x 2 root root 80 1970-01-01 00:00 003
```

2. Edit the permission of the USB port.

```
adb shell
cd /dev/bus/usb
su
chmod 777 –R ./*
ls –l
```



```
C:\Users\pan>adb shell
rk3126c:/ $ cd /de
default.prop   dev/
rk3126c:/ $ cd /de
default.prop   dev/
rk3126c:/ $ cd /dev/bus/usb/
rk3126c:/dev/bus/usb $ ls
001 002 003
rk3126c:/dev/bus/usb $ ls -l
total 0
drwxr-xr-x 2 root root 60 1970-01-01 00:00 001
drwxr-xr-x 2 root root 80 1970-01-01 00:00 002
drwxr-xr-x 2 root root 80 1970-01-01 00:00 003
rk3126c:/dev/bus/usb $ su
rk3126c:/dev/bus/usb # chmod 777 -R ./*
rk3126c:/dev/bus/usb # ls -l
total 0
drwxrwxrwx 2 root root 60 1970-01-01 00:00 001
drwxrwxrwx 2 root root 80 1970-01-01 00:00 002
drwxrwxrwx 2 root root 80 1970-01-01 00:00 003
rk3126c:/dev/bus/usb #
```

6.5 Why is it interrupted when the USB3Vision camera is getting stream？

Cause

The stream is too large and the frequency of DDR SDRAM is too small, causing the system to fail to read and write.

Solution

Set the frequency of DDR SDRAM to the maximum. Please consult the relevant chip manufacturer for details.

Appendix A. Appendixes

A.1 Error Code

The error may occurred during the MVC SDK integration are listed here for reference. You can search for the error description according to returned error codes or name.

Error Type	Error Code	Description
General Error Codes: From 0x80000000 to 0x800000FF		
MV_E_HANDLE	0x80000000	Error or invalid handle.
MV_E_SUPPORT	0x80000001	Not supported function.
MV_E_BUFOVER	0x80000002	Buffer is full.
MV_E_CALLORDER	0x80000003	Incorrect calling order
MV_E_PARAMETER	0x80000004	Incorrect parameter.
MV_E_RESOURCE	0x80000006	Applying resource failed.
MV_E_NODATA	0x80000007	No data.
MV_E_PRECONDITION	0x80000008	Precondition error, or the running environment changed.
MV_E_VERSION	0x80000009	Version mismatches.
MV_E_NOENOUGH_BUF	0x8000000A	Insufficient memory.
MV_E_ABNORMAL_IMAGE	0x8000000B	Abnormal image. Incomplete image caused by packet loss.
MV_E_LOAD_LIBRARY	0x8000000C	Importing DLL (Dynamic Link Library) failed.
MV_E_NOOUTBUF	0x8000000D	No buffer node can be outputted.
MV_E_UNKNOW	0x800000FF	Unknown error.
GenICam Series Error Codes: RFrom 0x80000100 to 0x800001FF		
MV_E_GC_GENERIC	0x80000100	Generic error.
MV_E_GC_ARGUMENT	0x80000101	Illegal parameters.
MV_E_GC_RANGE	0x80000102	The value is out of range.
MV_E_GC_PROPERTY	0x80000103	Attribute error
MV_E_GC_RUNTIME	0x80000104	Running environment error.
MV_E_GC_LOGICAL	0x80000105	Incorrect logic

Error Type	Error Code	Description
MV_E_GC_ACCESS	0x80000106	Node accessing condition error.
MV_E_GC_TIMEOUT	0x80000107	Timed out.
MV_E_GC_DYNAMICCAST	0x80000108	Conversion exception.
MV_E_GC_UNKNOW	0x800001FF	GenICam unknown error.
GigE Error Codes: From 0x80000200 to 0x800002FF, 0x80000221		
MV_E_NOT_IMPLEMENTED	0x80000200	The command is not supported by the device.
MV_E_INVALID_ADDRESS	0x80000201	The target address being accessed does not exist.
MV_E_WRITE_PROTECT	0x80000202	The target address is not writable.
MV_E_ACCESS_DENIED	0x80000203	The device has no access permission.
MV_E_BUSY	0x80000204	Device is busy, or the network disconnected.
MV_E_PACKET	0x80000205	Network packet error.
MV_E_NETER	0x80000206	Network error.
MV_E_IP_CONFLICT	0x80000221	Device IP address conflicted.
USB_STATUS Error Codes: From 0x80000300 to 0x800003FF		
MV_E_USB_READ	0x80000300	Reading USB error.
MV_E_USB_WRITE	0x80000301	Writing USB error.
MV_E_USB_DEVICE	0x80000302	Device exception.
MV_E_USB_GENICAM	0x80000303	GenICam error.
MV_E_USB_BANDWIDTH	0x80000304	Insufficient bandwidth.
MV_E_USB_UNKNOW	0x800003FF	USB unknown error.
Upgrade Error Codes: From 0x80000400 to 0x800004FF		
MV_E_UPG_FILE_MISMATCH	0x80000400	Firmware mismatches
MV_E_UPG_LANGUSGE_MISMATCH	0x80000401	Firmware language mismatches.
MV_E_UPG_CONFLICT	0x80000402	Upgrading conflicted (repeated upgrading requests during device upgrade).
MV_E_UPG_INNER_ERR	0x80000403	Camera internal error during upgrade.
MV_E_UPG_UNKNOW	0x800004FF	Unknown error during upgrade.
Exception Error Codes: From 0x00008001 to 0x00008002		

Error Type	Error Code	Description
MV_EXCEPTION_DEV_DISCONNECT	0x00008001	Device disconnected.
MV_EXCEPTION_VERSION_CHECK	0x00008002	SDK doesn't match the driver version.

A.2 MvCameraNode

Most open properties of camera, including name, definition, data type, value range, and access mode of each node are described in the sheet below. And the actual supported nodes depend on the camera type.

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
Device Control	DeviceType	Ienumeration	0: Transmitter 1: Receiver 2: Transceiver 3: Peripheral	R	Device Type
	DeviceScanType	Ienumeration	0: Areascan 1: Linescan	R/(W)	Device sensor scanning type, show that is the line scan camera or area scan camera
	DeviceVendorName	Istring	Any Null-terminated String	R	Device manufacturer name
	DeviceModelName	Istring	Any Null-terminated String	R	Device type
	DeviceManufacturerInfo	Istring	Any Null-terminated String	R	Device manufacturer information
	DeviceVersion	Istring	Any Null-terminated String	R	Device version
	DeviceFirmwareVersion	Istring	Any Null-terminated String	R	Firmware version

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	DeviceSerialNumber	Istring	Any Null-terminated String	R	Device serial number
	DeviceID	Istring	Any Null-terminated String	R	Device ID
	DeviceUserID	Istring	Any Null-terminated String	R/W	User-defined name
	DeviceUptime	Iinteger	≥0	R	Device running time
	BoardDeviceType	Iinteger	≥0	R	Device type
	DeviceConnectionSelector	Iinteger	≥0	R/(W)	DeviceConnectionSelector
	DeviceConnectionSpeed	Iinteger	≥0, UnitMbps	R	DeviceConnectionSpeed
	DeviceConnectionStatus	Ienumeration	0: Active 1: Inactive	R	DeviceConnectionStatus
	DeviceLinkSelector	Iinteger	≥0	R/(W)	DeviceLinkSelector
	DeviceLinkSpeed	Iinteger	≥0	R	DeviceLinkSpeed
	DeviceLinkConnectionCount	Iinteger	≥0	R	DeviceLinkConnectionCount
	DeviceLinkHeartbeatMode	Ienumeration	0: Off 1: On	R/W	Whether to need a heartbeat
	DeviceLinkHeartbeatTimeout	Iinteger	500-600000	R/W	Heartbeat timeout
	DeviceStreamChannelCount	Iinteger	≥0	R	StreamChannelCount
	DeviceStreamChannelSelector	Iinteger	≥0	R/W	StreamChannelSelector

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	DeviceStreamChannelType	Ienumeration	0: Transmitter 1: Receiver	R	StreamChannelType
	DeviceStreamChannelLink	Iinteger	≥0	R/(W)	StreamChannelLinkCount
	DeviceStreamChannelEndianness	Ienumeration	0: Little 1: Big	R/(W)	ImageDataEndianness
	DeviceStreamChannelPacketSize	Iinteger	Related to camera. Generally range in 220-220, step 8 ；	R/(W)	ReceiverStreanDataPacketSize
	DeviceEventChannelCount	Iinteger	≥0	R	Device supported
	DeviceCharacterSet	Ienumeration	1: UTF-8 2: ASCII	R	DeviceRegisterUseTheCharacterSet
	DeviceReset	Icommand	-	W	ResetDevice
	DeviceTemperatureSelector	Ienumeration	0: Sensor 1: Mainboard	R/W	TheSelectedDeviceTemperatureMeasurement
	DeviceTemperature	Ifloat	Unit, degree	R	TheSelectedDeviceTemperature
	FindMe	Icommand	-	W	Find current device
	DeviceMaxThroughput	Iinteger	≥0	R	DeviceMaxThroughput(bandwidth)
	WidthMax	Iinteger	＞0	R	ImageWidthMax, the data after the binning
	HeightMax	Iinteger	＞0	R	ImageHeightMax, the data after the binning

Function	**Name (key)** **Node A[Node B]: Indicates that node A depends on node B**	**Data Type**	**Numerical Value Definition**	**Access Mode** **R: Read** **W: Write** **(W): Write without Acquisitio n**	**Description**
	RegionSelector	Ienumer ation	0: Region0 1: Region1 2: Region2 8: All	R/(W)	ROISelector
	RegionDestination	Ienumer ation	0: Stream0 1: Stream1 2: Stream2	R/(W)	The ROI corresponds to the stream
	Width	Iinteger	＞0	R/(W)	the Width of ROI
	Height	Iinteger	＞0	R/(W)	the Height of ROI
	OffsetX	Iinteger	≥0	R/W	the OffsetX of ROI
	OffsetY	Iinteger	≥0	R/W	the OffsetY of ROI
	ReverseX	Iboolean	True False	R/W	WhetherToNeedReverseX
	ReverseY	Iboolean	True False	R/W	WhetherToNeedReverseY
	ReverseScanDirecti on	Iboolean	≥0	R/(W)	ReverseScanDirection
	PixelFormat	Ienumer ation	0x01080001: Mono8 0x01100003: Mono10 0x010C0004: Mono10Packed 0x01100005: Mono12 0x010C0006: Mono12Packed 0x01100007: Mono16	R/(W)	TheImagePixelFormat, Different types of camera support different pixel format, take the actual as the standard.

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
			0x02180014: RGB8Packed 0x02100032: YUV422_8 0x0210001F: YUV422_8_UYVY 0x02180020: YUV8_UYV 0x020C001E: YUV411_8_ UYYVYY 0x01080008: BayerGR8 0x01080009: BayerRG8 0x0108000A: BayerGB8 0x0108000B: BayerBG8 0x0110000c: BayerGR10 0x0110000d: BayerRG10 0x0110000e: BayerGB10 0x0110000f: BayerBG10 0x010C0029: BayerBG10Packed 0x010C0026: BayerGR10Packed		

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
			0x010C0027: BayerRG10Packed 0x010C0028: BayerGB10Packed 0x01100010: BayerGR12 0x01100011: BayerRG12 0x01100012: BayerGB12 0x01100013: BayerBG12 0x010C002D: BayerBG12Packed 0x010C002A: BayerGR12Packed 0x010C002B: BayerRG12Packed 0x010C002C: BayerGB12Packed 0x0110002E: BayerGR16 0x0110002F: BayerRG16 0x01100030: BayerGB16 0x01100031: BayerBG16		
	PixelSize	Ienumeration	8 : Bpp8 10: Bpp10	R/(W)	a pixel contains the number of bits

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
			12: Bpp12 16: Bpp16 24: Bpp24 32: Bpp32		
	ImageCompressionMode	Ienumeration	0: Off 1: JPEG	R/W	ImageCompressionMode
	ImageCompressionQuality	Iinteger	≥50	R/(W)	ImageCompressionQuality
	TestPatternGeneratorSelector	Ienumeration	8: Sensor 0: Region0 1: Region1 2: Region2	R/W	TestPatternGeneratorSelector
	TestPattern[TestPatternGeneratorSelector]	Ienumeration	0 : Off 1: Black 2: White 3: GreyHorizontalRamp 4: GreyVerticalRamp 5: GreyHorizontalRampMoving 6: GreyVerticalRampMoving 7: HorizontalLineMoving	R/W	TestImageSelector

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
			8: VerticalLineMoving 9: ColorBar 10: FrameCounter 11: MonoBar 12: TestImage12 13: TestImage13 14: ObliqueMonoBar 15: ObliqueColorBar 16: GradualMonoBar		
	BinningSelector	Ienumeration	0: Region0 1: Region1 2: Region2	R/W	PixelBinningSelector
	BinningHorizontal[BinningSelector]	Ienumeration	1: BinningHorizontal1 2: BinningHorizontal2 3: BinningHorizontal3 4: BinningHorizontal4	R/W	PixelBinningHorizontal
	BinningVertical[BinningSelector]	Ienumeration	1: BinningVertical1 2: BinningVertical2 3: BinningVertical3 4: BinningVertical4	R/W	PixelBinningVertical

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	DecimationHorizontal	Ienumeration	1: DecimationHorizontal1 2: DecimationHorizontal2 3: DecimationHorizontal3 4: DecimationHorizontal4	R/W	HorizontalPixelSampling
	DecimationVertical	Ienumeration	1: DecimationVertical1 2: DecimationVertical2 3: DecimationVertical3 4: DecimationVertical4	R/W	VerticalPixelSampling
	Deinterlacing	Ienumeration	0: Off 1: LineDuplication 2: Weave	R/W	Controls how the device performs de-interlacing
	FrameSpecInfoSelector	Ienumeration	0 : Timestamp 1: Gain 2: Exposure	R/(W)	WatermarkInformationSelector

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
			3: BrightnessInfo 4: WhiteBalance 5: Framecounter 6: ExtTriggerCount 7: LineInputOutput 8: ROIPosition		
	FrameSpecInfo	Iboolean	True False	R/W	Whether to use the watermark informaton
AcquisitionControl	AcquisitionMode	Ienumeration	0: SingleFrame 1: MultiFrame 2: Continuous	R/(W)	AcquisitionMode, SingleFrame /MultiFrame / Continuous
	AcquisitionStart	Icommand	-	(R)/W	AcquisitionStart
	AcquisitionStop	Icommand	-	(R)/W	AcquisitionStop
	AcquisitionBurstFrameCount	Iinteger	≥1	R/(W)	AcquisitionBurstFrameCount
	AcquisitionFrameRate	Ifloat	≥0.0, Unitfps	R/W	The value is in effect when Trigger Mode is off
	AcquisitionFrameRateEnable	Iboolean		R/W	Set the frame rate whether to effect
	AcquisitionLineRate	Iinteger	≥1	R/W	LineRateSet
	AcquisitionLineRateEnable	Iboolean	True False	R/W	LineRateControlEnable
	ResultingLineRate	Iinteger	≥0, Unithz	R	ResultingLineRate

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	ResultingFrameRate	Ifloat	≥0.0, Unitfps	R	The actual frame rate of the camera acquisition
	TriggerSelector	Ienumeration	0 : AcquisitionStart 1: AcquisitionEnd 2: AcquisitionActive 3: FrameStart 4: FrameEnd 5: FrameActive 6: FrameBurstStart 7: FrameBurstEnd 8: FrameBurstActive 9: LineStart 10: ExposureStart 11: ExposureEnd 12: ExposureActive	R/W	TriggerSelector
	TriggerMode[TriggerSelector]	Ienumeration	0: Off 1: On	R/W	TriggerMode
	TriggerSoftware[TriggerSelector]	Icommand	-	(R)/W	Perform a soft trigger
	TriggerSource[TriggerSelector]	Ienumeration	0: Line0 1: Line1 2: Line2 3.Line3 4: Counter0 7: Software	R/W	TriggerSource

Function	**Name (key)** **Node A[Node B]: Indicates that node A depends on node B**	**Data Type**	**Numerical Value Definition**	**Access Mode** **R: Read** **W: Write** **(W): Write without Acquisition**	**Description**
			8: FrequencyConverter		
	TriggerActivation[TriggerSelector]	Ienumeration	0: RisingEdge 1: FallingEdge 2.LevelHigh 3.LevelLow	R/W	TriggerActivation: RisingEdge FallingEdge LevelHigh LevelLow
	TriggerDelay[TriggerSelector]	Ifloat	≥0.0, Unitus	R/W	TriggerDelay
	TriggerCacheEnable	Iboolean	1 0	R/W	Whether to enable caching
	SensorShutterMode	Ienumeration	0: GlobalReset 1: TriggerRolling	R/W	Sensor exposure mode
	ExposureMode	Ienumeration	0: Timed 1: TriggerWidth	R/W	ExposureModeSelecton
	ExposureTime	Ifloat	≥0.0, Unitus	R/W	ExposureTime
	ExposureAuto	Ienumeration	0: Off 1: Once 2.Continuous	R/W	ExposureAuto
	AutoExposureTimeLowerLimit	Iinteger	≥2, Unitus	R/W	AutoExposureTimeLowerLimit
	AutoExposureTimeUpperLimit	Iinteger	≥2, Unitus	R/W	AutoExposureTimeUpperLimit
	GainShutPrior	Ienumeration	0: Shut 1: Gain	R/W	Sets the priority of the Exposure and Gain
	FrameTimeoutEnable	Iboolean	0: Off	R/W	FrameTimeoutEnable

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisitio n	Description
			1: On		
	FrameTimeoutTim e	Iinteger	≥87, Unitms	R/W	FrameTimeoutTime
	HDREnable	Iboolean	1 0	R/W	Whether to enable wide dynamic
	HDRSelector	Iinteger	≥0, ≤3	R/W	HDRSelector
	HDRShuter	Iinteger	≥32	R/W	HDRShuter
	HDRGain	Ifloat	≥0	R/W	HDRGain
DigitalIO Control	LineSelector	Ienumer ation	0: Line0 1: Line1 2: Line2 3: Line3 4: Line4	R/W	I/Oselection
	LineMode[LineSele ctor]	Ienumer ation	0: Input 1: Output 2: Trigger 8: Strobe	R/W	I/Omode
	LineInverter[LineS elector]	Iboolean	1 0	R/W	I/OLevelConversion
	LineTermination	Iboolean	1 0	R/W	I/Osingle ended differential selection
	LineStatus[LineSel ector]	Iboolean	-	R/(W)	I/Ostatus
	LineStatusAll	Iinteger	≥0	R	AllI/Ostatus
	LineSource[LineSel ector]	Ienumer ation	0: ExposureActive	R/W	Output event source

Function	**Name (key)** **Node A[Node B]: Indicates that node A depends on node B**	**Data Type**	**Numerical Value Definition**	**Access Mode** **R: Read** **W: Write** **(W): Write without Acquisition**	**Description**
	StrobeEnable	Iboolean	1 0	R/W	Enables the strobe signal to be output to the selected line
	LineDebouncerTime	Iinteger	-	R/W	I/O DebouncerTime
	StrobeLineDuration	Iinteger	≥0	R/W	OutputLevelDuration, Unit: us
	StrobeLineDelay	Iinteger	≥0	R/W	OuputDelay, Unit: us
	StrobeLinePreDelay	Iinteger	≥0	R/W	PreDelay, Unit: us
Counter And Timer Control	CounterSelector	Ienumeration	0: Counter0 1: Counter1 2: Counter2	R/W	CounterSelector
	CounterEventSource[CounterSelector]	Ienumeration	0: Off 11: Line0 12: Line1 13: Line2 1: AcquisitionTrigger 2: AcquisitionStart 3: AcquisitionEnd 4: FrameTrigger 5: FrameStart 6: FrameEnd 7: FrameBurstStart 8: FrameBurstEnd 9: FrameBurstEnd 10: LineEnd	R/W	CounterEventSource

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	CounterResetSource[CounterSelector]	Ienumeration	0: Off 1: CounterTrigger 3: Software 5: FrameTrigger 6: FrameStart	R/W	CounterResetSource
	CounterReset[CounterSelector]	Icommand	-	(R)/W	CounterReset
	CounterValue[CounterSelector]	Iinteger	≥1	R/W	CounterValue
	CounterCurrentValue	Iinteger	-	R	CounterCurrentValu
Analog Control	Gain[GainSelector]	Ifloat	≥0.0, UnitdB	R/W	Gain
	GainAuto[GainSelector]	Ienumeration	0: Off 1: Once 2.Continuous	R/W	GainAuto
	AutoGainLowerLimit	Ifloat	≥0.0, UnitdB	R/W	AutoGainLowerLimit
	AutoGainUpperLimit	Ifloat	≥0.1, UnitdB	R/W	AutoGainUpperLimit
	ADCGainEnable	Iboolean	0: Off 1: On	R/W	ADCGainEnable
	DigitalShift	Ifloat	≥0.0	R	DigitalShiftAdjustment
	DigitalShiftEnable	Iboolean	0: Off 1: On	R/W	DigitalShiftEnable
	Brightness	Iinteger	≥0	R/W	Brightness

Function	**Name (key)** **Node A[Node B]: Indicates that node A depends on node B**	**Data Type**	**Numerical Value Definition**	**Access Mode** **R: Read** **W: Write** **(W): Write without Acquisition**	**Description**
	BlackLevel[BlackLevelSelector]	Ifloat	≥0.0	R/W	BlackLevelAdjustment
	BlackLevelEnable	Iboolean	0: Off 1: On	R/W	BlackLeveAdjustmentlEnable
	BlackLevelAuto[BlackLevelSelector]	Ienumeration	0: Off 1: Continuous 2: Once	R/W	BlackLevelAdjustmentMode
	BalanceWhiteAuto	Ienumeration	0: Off 1: Continuous 2: Once	R/W	BalanceWhiteAuto
	BalanceRatioSelector	Ienumeration	0: Red 1: Green 2. Blue	R/W	WhiteBalanceRatioSelection
	BalanceRatio[BalanceRatioSelector]	Iinteger	≥0	R	WhiteBalanceRatio
	Gamma	Ifloat	＞0.0	R/W	GammaAdjustment
	GammaSelector	Ienumeration	1: User 2: sRGB	R/W	GammaSelection
	GammaEnable	Iboolean	0: Off 1: On	R/W	GammaEnable
	Sharpness	Iinteger	≥0	R/W	ImageSharpness
	SharpnessEnable	Iboolean		R/W	Enabel/DisableSharpness
	SharpnessAuto	Ienumeration	0: Off 1: Continuous 2: Once	R/W	SharpnessAutoType
	Hue	Iinteger	≥0	R	HueAdjustment

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	HueEnable	Iboolean	0: Off 1: On	R/W	HueEnable
	HueAuto	Ienumeration	0: Off 1: Continuous 2: Once	R/W	Gray automatic regulation type
	Saturation	Iinteger	≥0	R	SaturationAdjustment
	SaturationEnable	Iboolean	0: Off 1: On	R/W	SaturationEnable
	SaturationAuto	Ienumeration	0: Off 1: Continuous 2: Once	R/W	SaturationAuto
	DigitalNoiseReductionMode	Ienumeration	0: Off 1: Normal 2: Expert	R/W	DigitalNoiseReductionMode
	NoiseReduction	Iinteger	≥1	R/W	NoiseReductionValue
	AirspaceNoiseReduction	Iinteger	≥1	R/W	AirspaceNoiseReduction
	TemporalNoiseReduction	Iinteger	≥1	R/W	TemporalNoiseReduction
	AutoFunctionAOISelector	Ienumeration	0: AOI1 1: AOI2	R/W	AutoFunctionAOISelector
	AutoFunctionAOIWidth	Iinteger	≥0	R/W	AutoFunctionAOIWidth
	AutoFunctionAOIHeight	Iinteger	≥0	R/W	AutoFunctionAOIHeight

Function	**Name (key)** **Node A[Node B]: Indicates that node A depends on node B**	**Data Type**	**Numerical Value Definition**	**Access Mode** **R: Read** **W: Write** **(W): Write without Acquisition**	**Description**
	AutoFunctionAOIOffsetX	Iinteger	≥0	R	AutoFunctionAOIOffsetX
	AutoFunctionAOIOffsetY	Iinteger	≥0	R	AutoFunctionAOIOffsetY
	AutoFunctionAOIUsageIntensity	Iboolean	0: Off 1: On	R/W	Automatic exposure according to AOI area
	AutoFunctionAOIUsageWhiteBalance	Iboolean	0: Off 1: On	R	Automatic white balance according to AOI area
LUT Control	LUTSelector	Ienumeration	0: Luminance 1: Red 2: Green 3: Blue	R/W	Brightness、 R\G\B
	LUTEnable[LUTSelector]	Iboolean	True False	R/W	Enable
	LUTIndex[LUTSelector]	Iinteger	≥0	R/W	Index
	LUTValue[LUTSelector][LUTIndex]	Iinteger	Device-specific	R/W	Value
	LUTValueAll[LUTSelector]	Register	Device-specific	R/W	AllLUTValue
Encoder Control	EncoderSelector	Ienumeration	0 : Encoder0 1: Encoder1 2: Encoder2	R/W	EncoderSelector
	EncoderSourceA	Ienumeration	0: Line0 1: Line1 2: Line2 3: Line3	R/W	EncoderSourceASelector

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	EncoderSourceB	Ienumeration	0: Line0 1: Line1 2: Line2 3: Line3	R/W	EncoderSourceBSelector
	EncoderTriggerMode	Ienumeration	0: AnyDirection 1: ForwardOnly	R/W	EncoderTriggerMode
	EncoderCounterMode	Ienumeration	0: IgnoreDirection 1: FollowDirection	R/W	EncoderCounterMode
	EncoderCounter	Iinteger	≥0	R	Encoder Counter value adjustment
	EncoderCounterMax	Iinteger	≥1	R/W	EncoderCounterMax
	EncoderCounterReset	Icommand	-	R/W	EncoderCounterReset
	EncoderMaxReverseCounter	Iinteger	≥1	R/W	EncoderMaxReverseCounter
	EncoderReverseCounterReset	Icommand	-	R/W	EncoderReverseCounterReset
FrequencyConverterControl	InputSource	Ienumeration	0: Line0 1: Line1 2: Line2 3: Line3	R/W	Frequency divider input source
	SignalAlignment	Ienumeration	0 : RisingEdge 1: FallingEdge	R/W	Frequency divider signal direction
	PreDivider	Iinteger	≥1	R/W	PreDividerAdjustment
	Multiplier	Iinteger	≥1	R/W	MultiplierAdjustment

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	PostDivider	Iinteger	≥1	R/W	PostDividerAdjustment
Shading Correction	ShadingSelector	Ienumeration	0: FPNCCorrection 1: PRNUCCorrection	R/W	ShadingSelector
	ActivateShading	Icommand	-	R/(W)	ActivateShading
	NUCEnable	Iboolean	0: Off 1: On	R/W	NUCEnableSwitch
	FPNCEnable	Iboolean	1: On 0: Off	R	FPNCStateSwitch
	PRNUCEnable	Iboolean	0: Off 1: On	R	PRNUCtateSwitch
User Set Control	UserSetCurrent	Iinteger	>=0	R	CurrentUserParameters
	UserSetSelector	Ienumeration	0: Default 1: User set 1 2: User set 2 3: User set 3	R/W	Set load parameters
	UserSetLoad[UserSetSelector]	Icommand	-	R/W	Load
	UserSetSave[UserSetSelector]	Icommand	-	(R)/W	UserParametersSave
	UserSetDefault	Ienumeration	0: Default 1: User set 1 2: User set 2 3: User set 3	R/W	Default

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
Transport Layer Control	PayloadSize	Iinteger	≥0	R	The size of a frame of data
	GevVersionMajor	Iinteger	GEV Main Version Number	R	GEVVersionMajor
	GevVersionMinor	Iinteger	GEV Deputy Version Number	R	GEVVersionMinor
	GevDeviceModelIsBigEndian	Iboolean	0: not BigEndian 1: Is BigEndian	R	BigEndian
	GevDeviceModeCharacterSet	Ienumeration	1: UTF8	R	CharacterSet
	GevInterfaceSelector	Iinteger	>=0	R/W	GEVInterfaceSelector
	GevMACAddress	Iinteger	Mac Address	R	MACAddress
	GevSupportedOptionSelector	Ienumeration	31 : UserDefinedName 30: SerialNumber 29: HeartbeatDisable 28: LinkSpeed 27 : CCPApplicationSocket 26: ManifestTable 25: TestData 24: DiscoveryAckDelay 23 : DiscoveryAckDelayWritable	R/W	Selects the GEV option to interrogate for existing support

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
			22: ExtendedStatusCodes 21: PrimaryApplicationSwitchover 6 : Action 5: PendingAck 4: EventData 3: Event 2 : PacketResend 1: WriteMem 0: CommandsConcatenation 34: IPConfigurationLLA 33: IPConfigurationDHCP 32: IPConfigurationPersistentIP 63: PAUSEFrameReception 66: StreamChannelSourceSocket		

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	GevSupportedOption[GevSupportedOptionSelector]	Iboolean	0: Off 1: On	R	Indicates whether or not the selected GEV option is supported
	GevCurrentIPConfigurationLLA	Iboolean	0: Off 1: On	R	WhetherIPisLLA
	GevCurrentIPConfigurationDHCP[GevInterfaceSelector]	Iboolean	0: Off 1: On	R/W	WhetherIPisDHCP
	GevCurrentIPConfigurationPersistentIP[GevInterfaceSelector]	Iboolean	0: Off 1: On	R/W	WhetherIPisStaticIP
	GevPAUSEFrameReception[GevInterfaceSelector]	Iboolean	0: Off 1: On	R/W	Controls whether incoming PAUSE Frames are handled on the given logical link
	GevCurrentIPAddress[GevInterfaceSelector]	Iinteger	IP Address	R	IPAddress
	GevCurrentSubnetMask[GevInterfaceSelector]	Iinteger	Subnet Mask	R	SubnetMask
	GevCurrentDefaultGateway[GevInterfaceSelector]	Iinteger	Default Gateway	R	DefaultGateway
	GevFirstURL	Istring	-	R	XMLFirstURL
	GevSecondURL	Istring	-	R	XMLSecondURL
	GevNumberOfInterfaces	Iinteger	≥0	R	GEVNumberOfInterfaces

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	GevPersistentIPAddress[GevInterfaceSelector]	Iinteger	≥0	R/W	PersistentIPAddress
	GevPersistentSubnetMask[GevInterfaceSelector]	Iinteger	≥0	R/W	PersistentSubnetMask
	GevPersistentDefaultGateway[GevInterfaceSelector]	Iinteger	≥0	R/W	PersistentDefaultGateway
	GevLinkSpeed	Iinteger	≥0	R	LinkSpeed
	GevMessageChannelCount	Iinteger	≥0	R	MessageChannelCount
	GevStreamChannelCount	Iinteger	≥0	R	StreamChanne
	GevHeartbeatTimeout	Iinteger	≥0	R/W	HeartbeatTimeout
	GevGVCPHeartbeatDisable	Iboolean	0: Off 1: On	R/W	HeartbeatDisable
	GevTimestampTickFrequency	Iinteger	≥0, Unithz	R	TimestampTickFrequency
	GevTimestampControlLatch	Icommand	-	W	TimestampControlLatch
	GevTimestampControlReset	Icommand	-	W	TimestampControlReset
	GevTimestampControlLatchReset	Icommand	-	W	TimestampControlLatchReset
	GevTimestampValue	Iinteger	-	R	TimestampValue

Function	Name (key) Node A[Node B]: Indicates that node A depends on node B	Data Type	Numerical Value Definition	Access Mode R: Read W: Write (W): Write without Acquisition	Description
	GevCCP	Ienumeration	0: OpenAcess 1: ExclusiveAccess 2: ControlAccess	R/W	App terminal control authority
	GevStreamChannelSelector	Iinteger	>=0	R/W	StreamChannelSelector
	GevSCPInterfaceIndex[GevStreamChannelSelector]	Iinteger	>=0	R/(W)	GEVInterfaceIndex
	GevSCPHostPort[GevStreamChannelSelector]	Iinteger	>=0	R/W	HostPort
	GevSCPDirectionGevStreamChannelSelector]	Iinteger	>=0	R	StreamChannelDirection
	GevSCPSFireTestPacket[GevStreamChannelSelector]	Iboolean	0: Off 1: On	R/W	Fire Test PacketEnable
	GevSCPSDoNotFragment[GevStreamChannelSelector]	Iboolean	0: Off 1: On	R/W	Fire Test PacketEnable
	GevSCPSBigEndian[GevStreamChannelSelector]	Iboolean	0: Off 1: On	R/W	Stream data size
	PacketUnorderSupport	Iboolean	0: Off 1: On	R/W	Whether to support GVSP package to send out-of-order
	GevSCPSPacketSize	Iinteger	＞ 0, related to camera. Generally	R/(W)	NetworkPacketSize

Function	**Name (key)** **Node A[Node B]: Indicates that node A depends on node B**	**Data Type**	**Numerical Value Definition**	**Access Mode** **R: Read** **W: Write** **(W): Write without Acquisition**	**Description**
			range in 220-220, step 8 ;		
	GevSCPD[GevStreamChannelSelector]	Iinteger	≥0	R/W	ContractDelay
	GevSCDA[GevStreamChannelSelector]	Iinteger	IP Address	R/W	Destination address for streaming data
	GevSCSP[GevStreamChannelSelector]	Iinteger	Port Number	R	Source port for streaming data
	TLParamsLocked	Iinteger	≥0, ≤1	R/W	When the flow is 1

UD18501N