China Daheng Group, Inc. Beijing Image Vision Technology Branch

Python Interface Development User Manual

Version: V1.0.5

Date: 2021-03-08

DAHENG IMAGING 大恒图像

Web: www.daheng-imaging.com/en

Sales Email: isales@daheng-imaging.com

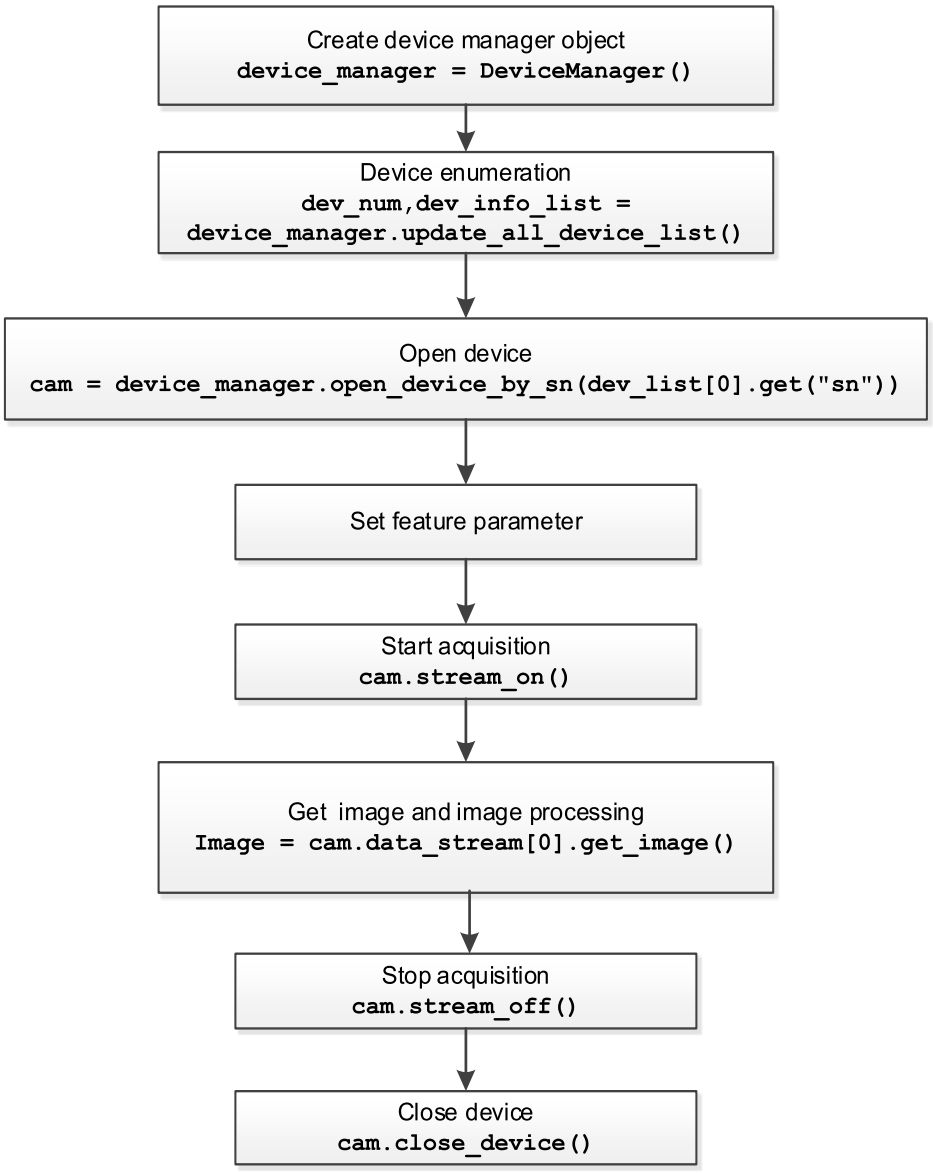
Sales Tel: +86 10 8282 8878

Support Email: isupport@daheng-imaging.com

Contents

1. Camera Workflow

1.1. Overall workflow



1.2. Function control flow

1.3. Overall code sample

```
import gxipy as gx
# Enumerate device. dev_info_list is a list of device information.
# The number of elements in the list is the number of devices
# enumerated. The list elements are dictionaries, which contain
# device information such as device index, ip information and so on
device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)

# Open device
# Get the list of basic device information
strSN = dev_info_list[0].get("sn")
# Open the device by serial number
cam = device_manager.open_device_by_sn(strSN)

# Start acquisition
cam.stream_on()

# Get the number of stream channels
# If int_channel_num == 1, the device has only one stream channel,
# and the number of data_stream elements in the list is 1
# If int_channel_num > 1, the device has multiple stream channels,
# and the number of data_stream elements in the list is greater than
# 1
# Currently, GigE, USB3.0, and USB2.0 cameras do not support
# multi-stream channels
# int_channel_num = cam.get_stream_channel_num()

# Get data
# num is the number of images acquired
num = 1
for i in range(num):
        # Get an image from the 0th stream channel
        raw_image = cam.data_stream[0].get_image()
        # Get RGB images from color raw images
        rgb_image = raw_image.convert("RGB")
        if rgb_image is None:
            continue
        # Create numpy array from RGB image data
        numpy_image = rgb_image.get_numpy_array()
        if numpy_image is None:
            continue
        # Display and save the acquired RGB image
        image = Image.fromarray(numpy_image, 'RGB')
        image.show()
        image.save("image.jpg")
# Stop acquisition, close device
cam.stream_off()
cam.close_device()
```

2. Programming Guide

2.1. Build programming environment

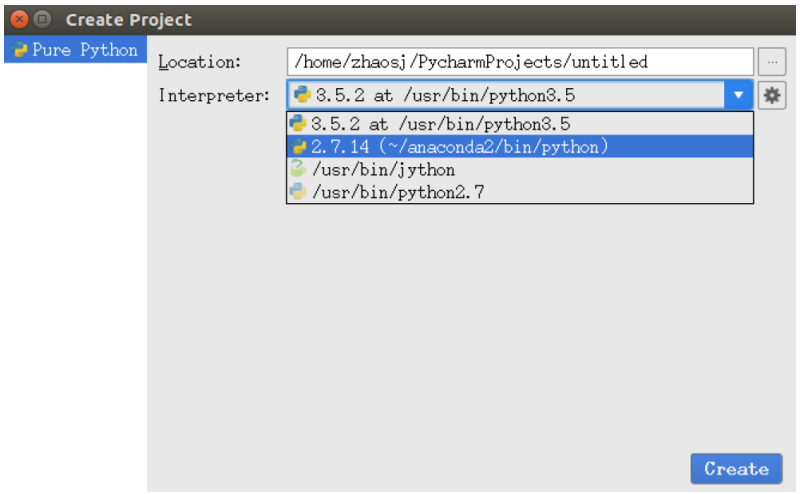
It is recommended that users use Python 2.7 and Python 3.5 first. This interface library has been tested in the above two versions.

2.1.1. Linux

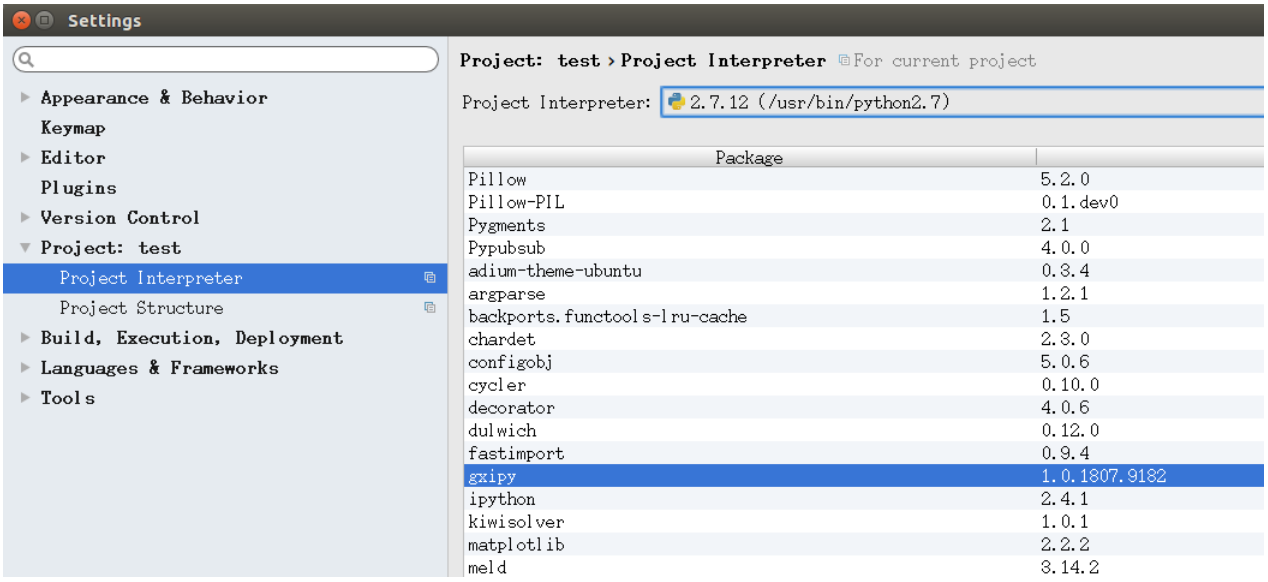
1) Install **pycharm-community** (community free version).

```
sudo apt-get install pycharm-community
```

2) Create a new **project**, select the **Python** interpreter that has a **gxipy** library. If there is no option, select **Settings** on the right to add the corresponding version of the Python interpreter.



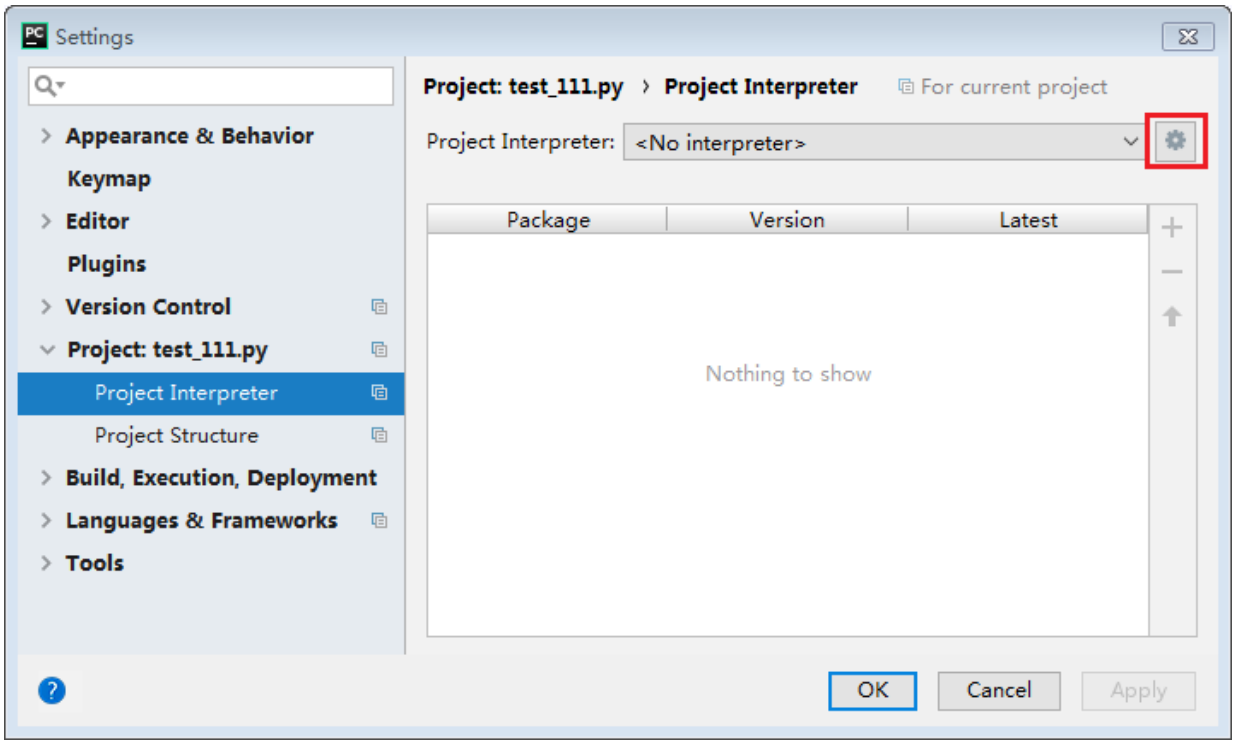
3) **File->Settings->Project->Project Interpreter** can check the **Package** installed by the Python interpreter. As shown in the figure, the **gxipy** library is installed normally and can be imported by user's program.



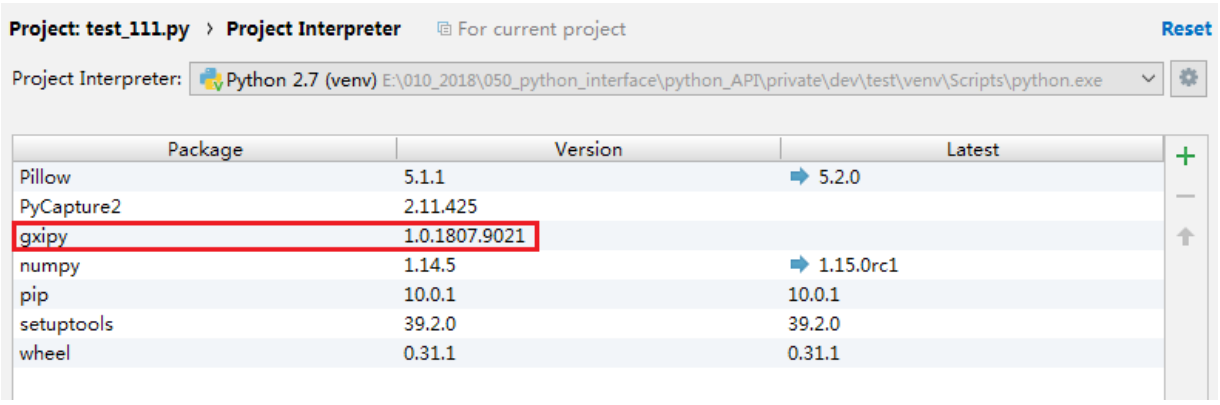
2.1.2. Windows

Take the Python2.7 and pycharm2018 platforms as examples to demonstrate the installation of the **gxipy** library in the Windows 7 64-bit operating system environment. When installing the **gxipy** library for the first time:

1) Open the pycharm2018 project to configure **File->Settings->Project->Project Interpreter**, select **Add** in the red box below.



2) Select **New environment**, select **Inherit global site-packages** and **Make available to all projects**, and select **OK**. At this point, the user will see that the **gxipy** library has been successfully loaded into the current project environment in the newly created **interpreter**.



2.2. QuickStart

2.2.1. Importing library

When using any classes, methods, and data types of the library in your code, the libraries should be imported at the beginning of the program in order to use the installed libraries.

Sample code:

```
import gxipy as gx
device_manager = gx.DeviceManager()
```

2.2.2. Enumeration device

The user enumerates all currently available devices by calling DeviceManager.update_device_list(). The return values of the function are the number of devices **dev_num** and the device information list **dev_info_list**. The number of elements in the device information list is the number of devices enumerated, the data type of the elements in the list is a dictionary, and the keys of the dictionary are as follows:

Key name	Significance	Type
index	Device index	int
vendor_name	Vendor name	string
model_name	Model name	string
sn	Device serial number	string
display_name	Device display name	string
device_id	Device identification	string
user_id	User-defined name	string
access_status	Access status	GxAccessStatus
device_class	Device class	GxDeviceClassList
mac	mac address（GEV camera only）	string
ip	ip address（GEV camera only）	string
subnet_mask	Subnet mask（GEV camera only）	string
gateway	Gateway（GEV camera only）	string
nic_mac	nic_mac address（GEV camera only）	string
nic_ip	nic_ip address（GEV camera only）	string
nic_subnet_mask	nic subnet mask（GEV camera only）	string
nic_gateway	nic gateway（GEV camera only）	string
nic_description	nic description（GEV camera only）	string

The code snippets for enumerating devices are as follows:

```
# Enumeration device
device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
   sys.exit(1)
```

Note:

In addition to the enumeration interfaces above, DeviceManager provides another enumeration interface DeviceManager.update_all_device_list():

1) For non-GigE Vision cameras, the two enumeration interfaces are functionally identical.

2) For GigE Vision cameras, the enumeration mechanisms used inside the library are different:

update_all_device_list：Using entire network enumeration, you can enumerate all GigE Vision cameras on the LAN.

update_device_list：Using subnet enumeration, you can only enumerate GigE Vision cameras that are on the same network segment within the LAN.

2.2.3. Open or close the device

The user can open the device by the following five different methods:

DeviceManager.open_device_by_sn(self, sn, access_mode=GxAccessMode.CONTROL)

DeviceManager.open_device_by_user_id(self,user_id,access_mode=GxAccessMode.CONTROL)

DeviceManager.open_device_by_index(self,index,access_mode=GxAccessMode.CONTROL)

DeviceManager.open_device_by_ip(ip, access_mode=GxAccessMode.CONTROL)

DeviceManager.open_device_by_mac(mac, access_mode=GxAccessMode.CONTROL)

Parameters:

sn	The device serial number
use-id	The user-defined name
index	The device index (1, 2, 3...)
mac	The device mac address (non-GigE Vision cameras are not support)
ip	The device ip address (non-GigE Vision cameras are not support)

Note:

The last two functions are only for GigE Vision cameras.

The user can call the Device. close_device() interface provided by Device to close the device and release all device's resources.

Sample code:

```
import gxipy as gx
```

```
# Enumerate devices. dev_info_list is a list of device information.
#The number of elements in the list is the number of devices
#enumerated. The list elements are dictionaries, which contain
#device information such as device index and ip information
device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)

# Open device
# Method 1
# Get the list of basic device information
str_sn = dev_info_list[0].get("sn")
# Open the device by serial number
cam = device_manager.open_device_by_sn(str_sn)

# Method 2
# Open the device by user ID
# str_user_id = dev_info_list[0].get("user_id")
# cam = device_manager.open_device_by_user_id(str_user_id)
# Method 3
# Open the device by index
# str_index = dev_info_list[0].get("index")
# cam = device_manager.open_device_by_index(str_index)

# The following methods only for GigE Vision cameras

# Method 4
# Open the device by ip address
# str_ip= dev_info_list[0].get("ip")
# cam = device_manager.open_device_by_ip(str_ip)

# Method 5
# Open the device by mac address
# str_mac = dev_info_list[0].get("mac")
# cam = device_manager.open_device_by_mac(str_mac)

# Close device
cam.close_device()
```

2.2.4. Acquisition control

After the device is opened successfully and the camera acquisition parameters are set, the user can call Device.stream_on() and Device.stream_off() to start and stop acquiring:

The interface and parameters related to the acquisition are provided by DataStream, and you can control the number of images acquired by setting the number of loops. You can get the number of stream channels of the device through the Device.get_stream_channel_num() interface. The acquired image uses RawImage.get_status() to determine whether it is an incomplete frame. The returned data structure codes are as follows:

◆ **Using get_image**

```
# Start acquisition
cam.stream_on()

# Get the number of stream channels
# If int_channel_num == 1, the device has only one stream channel,
#and the number of data_stream elements in the list is 1
# If int_channel_num > 1, the device has multiple stream channels,
#and the number of data_stream elements in the list is greater than
#1
# Currently, GigE, USB3.0, and USB2.0 cameras do not support multi-
#stream channels
# int_channel_num = cam.get_stream_channel_num()

# Get data
# Num is the number of images acquired
num = 1
for i in range(num):
    # Open the data stream of channel 0
    raw_image = cam.data_stream[0].get_image()
    if raw_image.get_status() == gx.GxFrameStatusList.INCOMPLETE:
        print("incomplete frame")

# Stop acquisition
cam.stream_off()
```

◆ **Using callback**

```
# Callback function define
def capture_callback(raw_image):
    if raw_image.get_status() == gx.GxFrameStatusList.INCOMPLETE:
        print("incomplete frame")

# Register callback
cam.data_stream[0].register_capture_callback(capture_callback)

# Start acquisition
cam.stream_on()

# Wait for 1 second, During this time, the callback function will be called
# automatically
time.sleep(1)

# Stop acquisition
cam.stream_off()

# Unregister callback
cam.data_stream[0].unregister_capture_callback()
```

2.2.5. Image processing

◆ **Image format conversion**

The image format conversion object is the **raw_image** got by acquiring the image with DataStream.get_image().

Functional description:

Convert Bayer format images to RGB format images. See the RawImage.convert() interface of RawImage for details.

Sample code:

1) Color camera

```
raw_image = cam.data_stream[0].get_image()
# Save raw images
raw_image.save_raw("raw_image.raw")
# Get RGB images from color raw images
rgb_image = raw_image.convert("RGB")
if rgb_image is None:
    continue
# Create numpy array from RGB image data
numpy_image = rgb_image.get_numpy_array()
if numpy_image is None:
    continue
# Then, the user can display and save the image according to the
#got numpy_array
```

2) Monochrome camera

```
raw_image = cam.data_stream[0].get_image()
# Get numpy array from mono raw images
numpy_image = raw_image.get_numpy()
if numpy_image is None:
    continue
# Then, the user can display and save the image according to the
#got numpy_array
```

◆ **Image quality improvement**

The interface library also provides an image quality improvement interface on the software side, and the user can selectively perform color correction, contrast, Gamma and other image quality improvement operations. The user can call the RGBImage.image_improvement() interface of RGBImage to implement the function. A simple example is as follows:

```
# Set image quality improvement parameters
if cam.GammaParam.is_readable():
    gamma_value = cam.GammaParam.get()
    gamma_lut = gx.Utility.get_gamma_lut(gamma_value)
else:
    gamma_lut = None
if cam.ContrastParam.is_readable():
    contrast_value = cam.ContrastParam.get()
    contrast_lut = gx.Utility.get_contrast_lut(contrast_value)
else:
    contrast_lut = None
color_correction_param = cam.ColorCorrectionParam.get()
# Acquire images, and convert the format
# .......
# Implement image quality improvement
rgb_image.image_improvement(color_correction_param,  contrast_lut,
gamma_lut)
```

1) Color correction:

Device feature parameter name: ColorCorrectionParam

Explanation: Improve the color reproduction of the camera to make the image closer to the visual perception of the human eye.

2) Contrast adjustment:

Contrast value: int, range [-50, 100], default value is 0

Device feature parameter name: ContrastParam

Explanation: The brightness ratio between the bright part and the dark part of the image is called contrast. For images with high contrast, the contour of the subject is clearer and the image is clearer. Conversely, the image with low contrast has unclear contour and unclear image.

3) Gamma adjustment:

Gamma value: int or float, range [0.1, 10.0], default value is 1

Device feature parameter name: GammaParam

Explanation: Gamma adjustment is to make the output of the display as close as possible to the input.

◆ **Image display and save**

Call the **Image.fromarray ()** interface of **PIL (Python Imaging Library)**, convert the numpy array to an Image, display and save it. The codes are as follows:

1） Monochrome camera

```
# Display and save the got mono image
image = Image.fromarray(numpy_image, 'L')
image.show()
image.save("acquisition_mono_image.jpg")
```

2） Color camera

```
# Display and save the got color image
image = Image.fromarray(numpy_image, 'RGB')
image.show()
image.save("acquisition_RGB_image.jpg")
```

2.2.6. Camera control

◆ **Feature parameter access type**

Here are three types of access to feature parameter: whether it is implemented, readable, or writable. The interfaces are designed as follows:

Feature.is_implement	Is this current feature controller support this feature
Feature.is_readable	Is this feature readable
Feature.is_writable	Is this feature writable

It is recommended that the user query the access type of the feature before operating the feature

parameter.

Sample code:

```
# Whether the acquisition function is implemented
is_implemented = cam.PixelFormat.is_implemented()
if is_implemented == True:
    # Is this writable
    is_writable = cam.PixelFormat.is_writable()
    if is_writable == True:
       # Set pixel format
        cam.PixelFormat.set(gx.GxPixelFormatEntry.MONO8)
    # Is this readable
    is_readable = cam.PixelFormat.is_readable()
    if is_readable == True:
       # Print pixel format
        print(cam.PixelFormat.get())
```

- **Feature control**

Feature control is divided into two categories: Device feature parameter and Stream feature parameter.

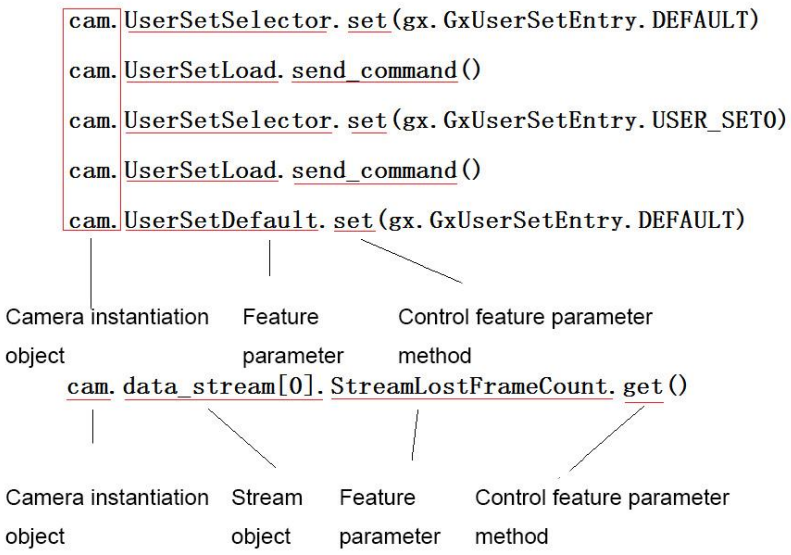
The difference is that the control functions of the feature parameters are different:

Device feature parameter: device information, such as width, height, exposure, gain, etc.

Stream feature parameter: feature access controllers for acquisition control and the acquisition of data statistics.

After acquiring the camera instantiation object, you can access the control interfaces of the feature parameter by accessing the feature parameter, which are in the Camera control.

For example:



Interface calls are classified according to the type of feature parameters:

Int:

Related interfaces:

IntFeature.set(int_value)	// Set
IntFeature.get()	// Get
IntFeature.get_range()	// Get the minimum, maximum and step size

Sample code:

```
# Get the settable range of the image width
int_range = cam.Width.get_range()
# Set the current image width to any value in the range
cam.Width.set(800)
# Get the width of current image
int_Width_value = cam.Width.get()
```

Float:

Related interfaces:

FloatFeature.set(float_value)	// Set
FloatFeature.get()	// Get
FloatFeature.get_range()	// Get the minimum, maximum, step size, unit, and whether the unit is valid

Sample code:

```
# Get the settable range and maximum value of the exposure time
float_range = cam.ExposureTime.get_range()
float_max = float_range["max"]
# Set the current exposure time to any value in the range
cam.ExposureTime.set(10.0)
# Get the current exposure time
float_exposure_value = cam.ExposureTime.get()
```

Enum:

Related interfaces:

EnumFeature.set(enum_value)	// Set
EnumFeature.get()	// Get
EnumFeature.get_range()	// Get the dictionary

Sample code:

```
# Get the settable range of enum value
enum_range = cam.PixelFormat.get_range()
# Set the current enum value
cam.PixelFormat.set(gx.GxPixelFormatEntry.MONO8)
# Print the current enum value
enum_PixelForamt_value,enum_PixelFormat_key = cam.PixelFormat.get()
```

Bool:

Related interfaces:

BoolFeature.set(bool_value)	// Set
BoolFeature.get()	// Get

Sample code:

```
# Set the current bool value
cam.LineInverter.set(True)
# Get bool value
bool_LineInverter_value = cam.LineInverter.get()
```

String:

Related interfaces:

StringFeature.set(string_value)	// Set
StringFeature.get()	// Get
StringFeature.get_string_max_length()	// Get the maximum length value of string feature

Sample code:

```
# Get the maximum length of the string feature can be set
string_max_length = cam.DeviceUserID.get_string_max_length()
# Get the current value of string
current_string = cam.DeviceUserID.get()
# Set string value
cam.DeviceUserID.set("MyUserID")
```

Buffer:

Related interfaces:

BufferFeature.set_buffer(buf)	// Set
BufferFeature.get_buffer()	// Get
BufferFeature.get_buffer_length()	// Get the length of the buffer type feature parameter

Sample code:

```
import gxipy as gx
# Get the length of buffer data
buffer_length = cam. UserData.get_buffer_length()
# Set the buffer data
cam.UserData.set_buffer(gx.Buffer.from_string(b'BufferFeature
Test!'))
# Get the buffer data
buffer_data = cam.LUTValueAll.get_buffer()
print("UserData: %s" % (buffer_data.get_data().decode())
```

Command:

Related interface:

```
CommandFeature.send_command      // Send command
```

Sample code:

```
# Send command: start or stop asquisition
cam.AcquisitionStart.send_command()
cam.AcquisitionStop.send_command()
```

Different types of devices have slightly different feature functions.

All the camera's feature parameters can be got in the Feature parameter of appendix.

2.2.7. Import and export camera configuration parameter

In the interface library, there is an interface for importing and exporting device configuration files for the user to call.

Sample code:

```
# Import camera configuration parameter file
cam.import_config_file("import_config_file.txt")
# Export camera configuration parameter file
cam.export_config_file("export_config_file.txt")
```

2.2.8. Error handling

When an exception occurs inside the calling interface function, the error handling mechanism detects and throws different types of exception, and the exception types inherit from **Exception**.

A typical error handling sample code:

```
try:
# When the interface function is called, the function throws an
# exception internally
dev_num, dev_info_list = device_manager.updata_device_list()
except Exception as exception:
   print("Print error message:%s" % exception)
    exit(1)
```

The user can also perform classification processing by testing the specific type of error captured:

```
if isinstance(exception, OutOfRange):
   print("OutOfRange: %s" % exception)
elif isinstance(exception, OffLine):
   print("OffLine: %s" % exception)
else:
   print("Other Error Type %s" % exception)
```

Exception type:

Exception type	Significance
UnexpectedError	Unexpected

NotFoundTL	Not found TL
NotFoundDevice	Not found device
OffLine	Offline
InvalidParameter	Invalid parameter
InvalidHandle	Invalid handle
InvalidCall	Invalid call
InvalidAccess	Invalid access
NeedMoreBuffer	Insufficient buffer
FeatureTypeError	Feature type error
OutOfRange	Out of range
NotInitApi	Not initialized
Timeout	Timeout
ParameterTypeError	Parameter type error

3. Appendix

3.1. Feature parameter

3.1.1. Device feature parameter

Feature parameter	Explanation	Feature class
DeviceInformation Section		
DeviceVendorName	Device vendor name	StringFeature
DeviceModelName	Device model name	StringFeature
DeviceFirmwareVersion	Device firmware version	StringFeature
DeviceVersion	Device version	StringFeature
DeviceSerialNumber	Device serial number	StringFeature
FactorySettingVersion	Factory setting version	StringFeature
DeviceUserID	User-defined name	StringFeature
DeviceLinkSelector	Device link selector, see C Software Development Manual for details	IntFeature
DeviceLinkThroughputLimit	Device link throughput limit	IntFeature
DeviceLinkThroughputLimitMode	Device link throughput limit mode, see GxSwitchEntry for details	EnumFeature
DeviceLinkCurrentThroughput	Current device link throughput	IntFeature
DeviceReset	Device reset	CommandFeature
TimestampTickFrequency	Timestamp tick frequency	IntFeature
TimestampLatch	Timestamp latch	CommandFeature
TimestampReset	Timestamp reset	CommandFeature
TimestampLatchReset	Timestamp latch reset	CommandFeature
TimestampLatchValue	Timestamp latch value	IntFeature
DevicePHYVersion	Device network chip version	StringFeature
DeviceTemperatureSelector	Device temperature selection, see GxDeviceTemperatureSelectorEntry for details	EnumFeature
DeviceTemperature	Device temperature	FloatFeature
ImageFormat Section		
SensorWidth	Sensor width	IntFeature

SensorHeight	Sensor height	IntFeature
WidthMax	Maximum width	IntFeature
HeightMax	Maximum height	IntFeature
OffsetX	Horizontal offset	IntFeature
OffsetY	Vertical offset	IntFeature
Width	Image width	IntFeature
Height	Image height	IntFeature
BinningHorizontal	Horizontal Binning	IntFeature
BinningVertical	Vertical Binning	IntFeature
DecimationHorizontal	Horizontal decimation	IntFeature
DecimationVertical	Vertical decimation	IntFeature
PixelSize	Pixel size, see GxPixelSizeEntry for details	EnumFeature
PixelColorFilter	Bayer format, see GxPixelColorFilterEntry for details	EnumFeature
PixelFormat	Pixel format, see GxPixelFormatEntry for details	EnumFeature
ReverseX	Horizontal reverse	BoolFeature
ReverseY	Vertical reverse	BoolFeature
TestPattern	Test pattern, see GxTestPatternEntry for details	EnumFeature
TestPatternGeneratorSelector	Test pattern generator selector, see C Software Development Manual and GxTestPatternGeneratorSelectorEntry for details	EnumFeature
RegionSendMode	ROI send mode, see GxRegionSendModeEntry for details	EnumFeature
RegionMode	Region mode, see GxSwitchEntry for details	EnumFeature
RegionSelector	Region selector, see C Software Development Manual and GxRegionSelectorEntry for details	EnumFeature
CenterWidth	Center width	IntFeature
CenterHeight	Center height	IntFeature

BinningHorizontalMode	Horizontal Binning mode, see GxBinningHorizontalModeEntry for details	EnumFeature
BinningVerticalMode	Vertical Binning mode, see GxBinningVerticalModeEntry for details	EnumFeature
SensorShutterMode	Sensor shutter mode, see GxSensorShutterModeEntry for details	EnumFeature
TransportLayer Section		
PayloadSize	Payload size	IntFeature
CenterWidth	Center width	IntFeature
CenterHeight	Center height	IntFeature
GevCurrentIPConfigurationLLA	Configuring IP in LLA mode	BoolFeature
GevCurrentIPConfigurationDHCP	Configuring IP in DHCP mode	BoolFeature
GevCurrentIPConfigurationPersistentIP	Configuring IP in permanent IP mode	BoolFeature
EstimatedBandwidth	Estimated bandwidth	IntFeature
GevHeartbeatTimeout	Heartbeat timeout time	IntFeature
GevSCPSPacketSize	Stream channel packet size	IntFeature
GevSCPD	Stream channel packet delay	IntFeature
GevLinkSpeed	Link speed	IntFeature
DigitalIO Section		
UserOutputSelector	User-defined output selector, see C Software Development Manual and GxUserOutputSelectorEntry for details	EnumFeature
UserOutputValue	User-defined output value	BoolFeature
UserOutputMode	User IO output mode, see GxUserOutputModeEntry for details	EnumFeature
StrobeSwitch	Strobe switch, see GxSwitchEntry for details	EnumFeature
LineSelector	Line selector, see C Software Development Manual and GxLineSelectorEntry for details	EnumFeature
LineMode	Line mode, see GxLineModeEntry for details	EnumFeature
LineSource	Line output source, see	EnumFeature

	GxLineSourceEntry for details	
LineInverter	Line level inversion	BoolFeature
LineStatus	Line status	BoolFeature
LineStatusAll	Status of all lines	IntFeature
AnalogControls Section		
GainAuto	Automatic gain, see GxAutoEntry for details	EnumFeature
GainSelector	Gain channel selector, see C Software Development Manual and GxGainSelectorEntry for details	EnumFeature
BlackLevelAuto	Automatic black level, see GxAutoEntry for details	EnumFeature
BlackLevelSelector	Black level channel selector, see C Software Development Manual and GxBlackLevelSelectEntry for details	EnumFeature
BalanceWhiteAuto	Automatic white balance, see GxAutoEntry for details	EnumFeature
BalanceRatioSelector	White balance channel selector, see C Software Development Manual and GxBalanceRatioSelectorEntry for details	EnumFeature
BalanceRatio	White balance ratio	FloatFeature
DeadPixelCorrect	Defective pixel correction, see GxSwitchEntry for details	EnumFeature
Gain	Gain	FloatFeature
BlackLevel	Black level	FloatFeature
GammaEnable	Gamma enable	BoolFeature
GammaMode	Gamma mode, see GxGammaModeEntry for details	EnumFeature
Gamma	Gamma	FloatFeature
DigitalShift	Digital shift	IntFeature
LightSourcePreset	Light source preset, see GxLightSourcePresetEntry for details	EnumFeature
CustomFeature Section		
ADCLevel	AD conversion level	IntFeature

HBlanking	Horizontal blanking	IntFeature
VBlanking	Vertical blanking	IntFeature
UserPassword	User encryption zone password	StringFeature
VerifyPassword	User encryption zone verify password	StringFeature
UserData	User encryption zone content	BufferFeature
ExpectedGrayValue	Expected gray value	IntFeature
AALightEnvironment	Auto exposure, auto gain, lighting environment type, see GxAALightEnvironmentEntry for details	EnumFeature
ImageGrayRaiseSwitch	Image gray raise switch, see GxSwitchEntry for details	EnumFeature
AAROIOffsetX	Auto adjust the X offset of ROI	IntFeature
AAROIOffsetY	Auto adjust the Y offset of ROI	IntFeature
AAROIWidth	Auto adjust the width of ROI	IntFeature
AAROIHeight	Auto adjust the height of ROI	IntFeature
AutoGainMin	Minimum automatic gain	FloatFeature
AutoGainMax	Maximum automatic gain	FloatFeature
AutoExposureTimeMin	Minimum automatic exposure	FloatFeature
AutoExposureTimeMax	Maximum automatic exposure	FloatFeature
ContrastParam	Contrast parameter	IntFeature
ColorCorrectionParam	Color correction parameter	IntFeature
AWBROIOffsetX	The X coordinate of automatic white balance ROI	IntFeature
AWBROIOffsetY	The Y coordinate of automatic white balance ROI	IntFeature
AWBROIWidth	The width of automatic white balance ROI	IntFeature
AWBROIHeight	The height of automatic white balance ROI	IntFeature
GammaParam	Gamma parameter	FloatFeature
AWBLampHouse	Automatic white balance lamp house, see GxAWBLampHouseEntry for details	EnumFeature

SharpnessMode	Sharpening model, see GxSwitchEntry for details	EnumFeature
Sharpness	Sharpness	FloatFeature
FrameInformation	Image frame information	BufferFeature
DataFieldSelector	User selects the flash data field, see GxUserDataFieldSelectorEntry for details	EnumFeature
DataFieldValue	User data field content	BufferFeature
FlatFieldCorrection	Flat field correction switch, see GxSwitchEntry for details	EnumFeature
NoiseReductionMode	Noise reduction switch, see GxSwitchEntry for details	EnumFeature
NoiseReduction	Noise reduction value	FloatFeature
FFCLoad	Get flat field correction parameters	BufferFeature
FFCSave	Set flat field correction parameters	BufferFeature
StaticDefectCorrection	Static dead pixel correction switch, see GxSwitchEntry for details	EnumFeature
UserSetControl Section		
UserSetLoad	Load the user set	CommandFeature
UserSetSave	Save the user set	CommandFeature
UserSetSelector	User set selector, see C Software Development Manual and GxUserSetEntry for details	EnumFeature
UserSetDefault	Default the user set, see GxUserSetEntry for details	EnumFeature
Event Section		
EventSelector	Event source select, see GxEventSelectorEntry for details	EnumFeature
EventNotification	Switch of the notification to the host application of the occurrence of the selected Event, see GxSwitchEntry for details	EnumFeature
EventExposureEnd	Exposure end event	IntFeature
EventExposureEndTimestamp	The timestamp of Exposure end event	IntFeature
EventExposureEndFrameID	The frame id of Exposure end event	IntFeature

EventBlockDiscard	Block discard event	IntFeature
EventBlockDiscardTimestamp	The timestamp of Block discard event	IntFeature
EventOverrun	Event queue overflow event	IntFeature
EventOverrunTimestamp	The timestamp of event queue overflow event	IntFeature
EventFrameStartOvertrigger	Trigger signal shield event	IntFeature
EventFrameStartOvertriggerTimestamp	The timestamp of trigger signal shield event	IntFeature
EventBlockNotEmpty	Frame memory not empty event	IntFeature
EventBlockNotEmptyTimestamp	The timestamp of frame memory not empty event	IntFeature
EventInternalError	Internal erroneous event	IntFeature
EventInternalErrorTimestamp	The timestamp of internal erroneous event	IntFeature
EventFrameBurstStartOvertrigger	Frame burst start overtrigger event ID	IntFeature
EventFrameBurstStartOvertriggerFrameID	Frame burst start overtrigger event frame ID	IntFeature
EventFrameBurstStartOvertriggerTimestamp	Frame burst start overtrigger event timestamp	IntFeature
EventFrameStartWait	Frame start wait event ID	IntFeature
EventFrameStartWaitTimestamp	Frame start wait event timestamp	IntFeature
EventFrameBurstStartWait	Frame burst start wait event ID	IntFeature
EventFrameBurstStartWaitTimestamp	Frame burst start wait event timestamp	IntFeature
EventBlockDiscardFrameID	Data block discard event frame ID	IntFeature
EventFrameStartOvertriggerFrameID	Frame start wait overtrigger event frame ID	IntFeature
EventBlockNotEmptyFrameID	Data block not empty event frame ID	IntFeature
EventFrameStartWaitFrameID	Frame start wait event frame ID	IntFeature
EventFrameBurstStartWaitFrameID	Frame burst start wait event frame ID	IntFeature
LUT Section		
LUTValueAll	LUT content	BufferFeature
LUTSelector	LUT selector, see C Software Development Manual and	EnumFeature

	GxLutSelectorEntry for details	
LUTEnable	LUT enable	BoolFeature
LUTIndex	LUT index	IntFeature
LUTValue	LUT value	IntFeature
Color Transformation Control		
ColorTransformationMode	Color transformation mode, see GxColorTransformationModeEntry for details	EnumFeature
ColorTransformationEnable	Color transformation enable	BoolFeature
ColorTransformationValueSelector	Color transformation matrix value selector, see GxColorTransformationValueSelectorEntry for details	EnumFeature
ColorTransformationValue	Color transformation matrix value	FloatFeature
SaturationMode	Saturation switch, see GxSwitchEntry for details	EnumFeature
Saturation	Saturation value	IntFeature
ChunkData Section		
ChunkModeActive	Chunk data enable	BoolFeature
ChunkEnable	Single chunk data enable	BoolFeature
ChunkSelector	Chunk data selector, see C Software Development Manual and GxChunkSelectorEntry for details	EnumFeature
Device Feature		
DeviceCommandTimeout	Device command timeout	IntFeature
DeviceCommandRetryCount	Device command retry count	IntFeature
AcquisitionTrigger Section		
FrameBufferOverwriteActive	Frame buffer overwrite enable	BoolFeature
AcquisitionStart	Start acquisition	CommandFeature
AcquisitionStop	Stop acquisition	CommandFeature
TriggerSoftware	Software trigger	CommandFeature
TransferStart	Start transfer	CommandFeature
AcquisitionMode	Acquisition mode, see	EnumFeature

	GxAcquisitionModeEntry for details	
TriggerMode	Trigger mode, see GxSwitchEntry for details	EnumFeature
TriggerActivation	Trigger activation, see GxTriggerActivationEntry for details	EnumFeature
ExposureAuto	Auto exposure, see GxAutoEntry for details	EnumFeature
TriggerSource	Trigger source, see GxTriggerSourceEntry for details	EnumFeature
ExposureMode	Exposure mode, see GxExposureModeEntry for details	EnumFeature
TriggerSelector	Trigger type selector, see C Software Development Manual and GxTriggerSelectorEntry for details.	EnumFeature
TransferControlMode	Transfer control mode, see GxTransferControlModeEntry for details	EnumFeature
TransferOperationMode	Transfer operation mode, see GxTransferOperationModeEntry for details	EnumFeature
AcquisitionFrameRateMode	Acquisition frame rate adjustment mode, see GxSwitchEntry for details	EnumFeature
FixedPatternNoiseCorrectMode	Fixed pattern noise correction mode, see GxSwitchEntry for details	EnumFeature
ExposureTime	Exposure time	FloatFeature
TriggerFilterRaisingEdge	Raising edge trigger filter	FloatFeature
TriggerFilterFallingEdge	Falling edge trigger filter	FloatFeature
TriggerDelay	Trigger delay	FloatFeature
AcquisitionFrameRate	Acquisition frame rate	FloatFeature
CurrentAcquisitionFrameRate	Current acquisition frame rate	FloatFeature
TransferBlockCount	Transfer block count	IntFeature
TriggerSwitch	External trigger switch, see GxSwitchEntry for details	EnumFeature
AcquisitionSpeedLevel	Acquisition speed level	IntFeature
AcquisitionFrameCount	Acquisition frame count	IntFeature

AcquisitionBurstFrameCount	Acquisition burst frame count	IntFeature
AcquisitionStatusSelector	Acquisition status selector, see C Software Development Manual and GxAcquisitionStatusSelectorEntry for details	EnumFeature
AcquisitionStatus	Acquisition status	BoolFeature
ExposureDelay	Exposure delay	FloatFeature
ExposureOverlapTimeMax	Maximum exposure overlap time	FloatFeature
ExposureTimeMode	Exposure time mode，see GxExposureTimeModeEntry for details	EnumFeature
CounterAndTimerControl Section		
TimerSelector	Selects which Counter to configure, see GxTimerSelectorEntry for details	EnumFeature
TimerDuration	Sets the duration (in microseconds) of the Timer pulse	FloatFeature
TimerDelay	Sets the duration (in microseconds) of the delay to apply at the reception of a trigger before starting the Timer	FloatFeature
TimerTriggerSource	Selects the source of the trigger to start the Timer, see GxTimerTriggerSourceEntry for details	EnumFeature
CounterSelector	Selects which Counter to configure, see GxCounterSelectorEntry for details	EnumFeature
CounterEventSource	Selects the events that will be the source to increment the Counter, see GxCounterEventSourceEntry for details	EnumFeature
CounterResetSource	Selects the signals that will be the source to reset the Counter, see GxCounterResetSourceEntry for details	EnumFeature
CounterResetActivation	Selects the Activation mode of the Counter Reset Source signal, see GxCounterResetActivationEntry for details	EnumFeature
CounterReset	Does a software reset of the selected Counter and starts it	CommandFeature
CounterTriggerSource	Counter trigger source, see	EnumFeature

	GxCounterTriggerSourceEntry for details	
CounterDuration	Counter duration value	IntFeature
TimerTriggerActivation	Timer Trigger Activation, see GxTimerTriggerActivationEntry for details	EnumFeature
RemoveParameterLimitControl Section		
RemoveParameterLimit	Remove paremeter range restriction switch, see GxSwitchEntry for details	EnumFeature

3.1.2. Stream feature parameter

Feature parameter	Explanation	Feature class
StreamAnnouncedBufferCount	Announced buffer count	IntFeature
StreamDeliveredFrameCount	Delivered frame count (including incomplete frames)	IntFeature
StreamLostFrameCount	The number of lost frames caused by insufficient buffer	IntFeature
StreamIncompleteFrameCount	Delivered incomplete frame count	IntFeature
StreamDeliveredPacketCount	Delivered packet count	IntFeature
StreamResendPacketCount	Resend packet count	IntFeature
StreamRescuedPacketCount	Rescued packet count	IntFeature
StreamResendCommandCount	Resend command count	IntFeature
StreamUnexpectedPacketCount	Unexpected packet count	IntFeature
MaxPacketCountInOneBlock	Maximum resend packet count in one block	IntFeature
MaxPacketCountInOneCommand	Maximum packet count in one resend command	IntFeature
ResendTimeout	Resend timeout	IntFeature
MaxWaitPacketCount	Maximum waiting packet count	IntFeature
ResendMode	Resend mode, see GxSwitchEntry for details	EnumFeature
StreamMissingBlockIDCount	Missing BlockID count	IntFeature
BlockTimeout	Data block timeout time	IntFeature
MaxNumQueueBuffer	Maximum number of Buffer in the acquisition queue	IntFeature

PacketTimeout	Packet timeout time	IntFeature
StreamTransferSize	Transfer data block size	IntFeature
StreamTransferNumberUrb	Transfer data block number	IntFeature
SocketBufferSize	Socket buffer size in kilobytes	IntFeature
StopAcquisitionMode	Stop acquisition mode, see GxStopAcquisitionModeEntry for details	EnumFeature
StreamBufferHandlingMode	Buffer handling mode see GxDSStreamBufferHandlingModeEntry for details	EnumFeature

3.2. Function class definition

3.2.1. Feature

It is responsible for checking the basic functions of various data type functions and testing if they are implemented, readable and writable.

The Feature class is the parent of the IntFeature/FloatFeature/ EnumFeature/BoolFeature/StringFeature /BufferFeature/CommandFeature.

Interface list:

is_implemented()	Test if the feature is implemented
is_readable()	Test if the feature is readable
is_writable()	Test if the feature is writable

◆ **Interface description**

➢ **is_implemented**

Statement:

```
Feature.is_implemented()
```

Significance:

Test if the feature has been implemented

Return value:

Ture: Implement

False: Unimplement

Exception handling:

1) If the feature is an invalid parameter, return False.

2) If it fails due to other reasons, an exception is thrown. For details, see Error handling.

➢ **is_readable**

Statement:

```
Feature.is_readable()
```

Significance:

Test if the feature is readable

Return value:

Ture: Readable

False: Unreadable

Exception handling:

1) If the function is not implemented, return False.

2) If it fails due to other reasons, an exception is thrown. For details, see Error handling.

➢ **is_writable**

Statement:

```
Feature.is_writable()
```

Significance:

Test if the feature is writable

Return value:

Ture: Writable

False: Unwritable

Exception handling:

1) If the function is not implemented, return False.

2) If it fails due to other reasons, an exception is thrown. For details, see Error handling.

3.2.2. IntFeature

It is responsible for checking and controlling the camera's int feature, inherited from the Feature class.

Interface list:

is_implemented()	Test if the int feature is implemented
is_readable()	Test if the int feature is readable
is_writable ()	Test if the int feature is writable
get_range()	Get the int feature range dictionary
get()	Get the int feature value
set(int_value)	Set the int feature value

◆ **Interface description**

➢ **is_implemented**

See Feature::is_implemented() for details.

➢ **is_readable**

See Feature::is_readable() for details.

➢ **is_writable**

See Feature::is_writable() for details.

➢ **get_range**

Statement:

```
IntFeature.get_range()
```

Significance:

Get the int feature range dictionary

Return value:

Record the dictionary of int feature ranges. The key contains: minimum, maximum and step size.

Exception handling:

1) If the int feature is not implemented, then prints the information that does not support the int feature to get range, and the function returns **None**.

2) If getting the parameter range of the int feature unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **get**

Statement:

```
IntFeature.get()
```

Significance:

Get the int feature value

Return value:

The int value got

Exception handling:

1) If the int feature is not implemented or is unreadable, then prints the unreadable information of the int feature, and the function returns **None**.

2) If getting the int feature value unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **set**

Statement:

```
IntFeature.set(self,int_value)
```

Significance:

Set the value of int feature

Formal parameter:

The set int value

Exception handling:

1) If the input parameter is not an int value, a **ParameterTypeError** exception is thrown.

2) If the int feature is not implemented or is not writable, then prints the information that the int feature is not writable, and the function returns **None**.

3) If the input parameter is not within the range of the int feature, then prints the information that exceeds the range of the int feature and prints the range, and the function returns **None**.

4) If setting the int feature unsuccessfully, an exception is thrown. For details, see Error handling.

3.2.3. FloatFeature

It is responsible for checking and controlling the camera's float feature, inherited from the Feature class.

Interface list:

is_implemented()	Test if the float feature is implemented
is_readable()	Test if the float feature is readable
is_writable ()	Test if the float feature is writable
get_range()	Get the float feature range dictionary
get()	Get the float feature value
set(float_value)	Set the float feature value

◆ **Interface description**

➢ **is_implemented**

See Feature::is_implemented() for details.

➢ **is_readable**

See Feature::is_readable() for details.

➢ **is_writable**

See Feature::is_writable() for details.

➢ **get_range**

Statement:

```
FloatFeature.get_range()
```

Significance:

Get the float feature range dictionary

Return value:

Record the dictionary of float feature ranges. The keys contain: minimum, maximum, inc step size, unit, whether the inc_is_valid unit is valid.

Exception handling:

1) If the float feature is not implemented, then prints the information that does not support the float feature to get range, and the function returns **None**.

2) If getting the parameter range of the float feature unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **get**

Statement:

```
FloatFeature.get()
```

Significance:

Get the value of float feature

Return value:

The float feature value got

Exception handling:

1) If the float feature is not implemented or is unreadable, then prints the unreadable information of the float feature, and the function returns **None**.

2) If getting the float feature value unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **set**

Statement:

```
FloatFeature.set(float_value)
```

Significance:

Set the value of float feature

Formal parameter:

```
[in]float_value  The set value of float
```

Exception handling:

1) If the input parameter is not a float value, throw a **ParameterTypeError** exception.

2) If the float feature is not implemented or is not writable, then prints the information that the float feature is not writable, and the function returns **None**.

3) If the input parameter is not within the range of the float feature, then prints the information that exceeds the range of the float feature and prints the range, and the function returns **None**.

4) If setting the float feature unsuccessfully, an exception is thrown. For details, see Error handling.

3.2.4. EnumFeature

It is responsible for checking and controlling the camera's enum feature, inherited from the Feature class.

Interface list:

is_implemented()	Test if the enum feature is implemented
is_readable()	Test if the enum feature is readable
is_writable ()	Test if the enum feature is writable
get_range()	Get the enum feature range dictionary
get()	Get the value and string of the enum feature
set(enum_value)	Set the enum feature value

◆ **Interface description**

➢ **is_implemented**

See Feature::is_implemented() for details.

➢ **is_readable**

See Feature::is_readable() for details.

➢ **is_writable**

See Feature::is_writable() for details.

➢ **get_range**

Statement:

```
EnumFeature.get_range()
```

Significance:

Get the enmu feature range dictionary

Return value:

Record enum feature range dictionary

Exception handling:

1) If the enum feature is not implemented, then prints the information that does not support the enum feature to get range, and the function returns **None**.

2) If getting the range of the enum feature unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **get**

Statement:

```
EnumFeature.get()
```

Significance:

Get the value and string of the enum feature

Return value:

1) The value of the enum feature

2) The string of enum feature

Exception handling:

1) If the enum feature is not implemented or is unreadable, then prints the unreadable information of the enmu feature, and the function returns **None**.

2) If getting enmu feature unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **set**

Statement:

```
EnumFeature.set( enum_value)
```

Significance:

Set the value of enmu feature

Formal parameter:

```
[in]enum_value    The set value of enmu
```

Exception handling:

1) If the input parameter is not an int value, throw a **ParameterTypeError** exception.

2) If the enmu feature is not implemented or is not writable, then prints the information that the enmu feature is not writable, and the function returns **None**.

3) If the input parameter is not within the range of the enmu feature, then prints the information that exceeds the range of the enmu feature and prints the range, and the function returns **None**.

4) If setting the enmu feature unsuccessfully, an exception is thrown. For details, see Error handling.

3.2.5. BoolFeature

It is responsible for checking and controlling the camera's bool feature, inherited from the Feature class.

Interface list:

is_implemented()	Test if the bool feature is implemented

is_readable()	Test if the bool feature is readable
is_writable ()	Test if the bool feature is writable
get()	Get the value of bool feature
set(bool_value)	Set the bool feature value

◆ **Interface description**

➢ **is_implemented**

See Feature::is_implemented() for details.

➢ **is_readable**

See Feature::is_readable() for details.

➢ **is_writable**

See Feature::is_writable() for details.

➢ **get**

Statement:

```
BoolFeature.get()
```

Significance:

Get the feature value of bool

Return value:

The bool feature value got

Exception handling:

1) If the bool feature is not implemented or is unreadable, then prints the unreadable information of the bool feature, and the function returns **None.**

2) If getting bool feature value unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **set**

Statement:

```
BoolFeature.set( bool_value)
```

Significance:

Set the value of bool feature

Formal parameter:

```
[in]bool_value    The set value of got bool
```

Exception handling:

1) If the input parameter is not a bool value, throw a **ParameterTypeError** exception.

2) If the bool feature is not implemented or is not writable, then prints the information that the bool feature is not writable, and the function returns **None**.

3) If setting the bool feature unsuccessfully, an exception is thrown. For details, see Error handling.

3.2.6. StringFeature

It is responsible for checking and controlling the camera's string feature, inherited from the Feature class.

Interface list:

is_implemented()	Test if the string feature is implemented
is_readable()	Test if the string feature is readable
is_writable ()	Test if the string feature is writable
get_string_max_length()	Get the maximum length that a string feature value can be set
get()	Get the value of string feature
set(input_string)	Set the string feature value

◆ **Interface description**

➢ **is_implemented**

See Feature::is_implemented() for details.

➢ **is_readable**

See Feature::is_readable() for details.

➢ **is_writable**

See Feature::is_writable() for details.

➢ **get_string_max_length**

Statement:

```
StringFeature.get_string_max_length()
```

Significance:

Get the maximum length of string feature can be set

Return value:

The maximum length that string feature can be set

Exception handling:

1) If the string feature is not implemented, then prints the information that is not implemented, and the function returns **None**.

2) If getting the maximum length of the string feature unsuccessfully, an exception is thrown. For details, see Error handling.

- **get**

Statement:

```
StringFeature.get()
```

Significance:

Get the value of string feature

Return value:

The string feature value got

Exception handling:

1) If the string feature is not implemented or is unreadable, then prints the unreadable information of the string feature, and the function returns **None**.

2) If getting the string feature value unsuccessfully, an exception is thrown. For details, see Error handling.

- **set**

Statement:

```
StringFeature.set( input_string)
```

Significance:

Set the value of string feature

Formal parameter:

```
[in]input_string    The set value of string
```

Exception handling:

1) If the input parameter is not a string value, throw a **ParameterTypeError** exception.

2) If the string feature is not implemented or is not writable, then prints the information that the string feature is not writable, and the function returns **None**.

3) If the input parameter length is greater than the settable maximum length, then prints the information that exceeds the maximum value of the string type feature length and prints the maximum value, and the function returns **None**.

4) If setting the string feature unsuccessfully, an exception is thrown. For details, see Error handling.

3.2.7. BufferFeature

It is responsible for checking and controlling the camera's buffer feature, inherited from the Feature class.

Interface list:

is_implemented()	Test if the buffer feature is implemented
is_readable()	Test if the buffer feature is readable
is_writable ()	Test if the buffer feature is writable
get_buffer_length()	Get the length of the buffer feature
get_buffer()	Get the data of the buffer feature
set_buffer(buf)	Set the buffer feature data

◆ **Interface description**

➢ **is_implemented**

See Feature::is_implemented() for details.

➢ **is_readable**

See Feature::is_readable() for details.

➢ **is_writable**

See Feature::is_writable() for details.

➢ **get_buffer_length**

Statement:

```
BufferFeature.get_buffer_length()
```

Significance:

Get the length of the buffer feature

Return value:

The length of the buffer feature

Exception handling:

1) If the buffer feature is not implemented, then prints the information that does not support the buffer feature to get range, and the function returns **None**.

2) If getting the range of the buffer feature unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **get_buffer**

Statement:

```
BufferFeature.get_buffer()
```

Significance:

Get the data of buffer feature

Return value:

Buffer object

Exception handling:

1) If the buffer feature is not implemented or is unreadable, then prints the unreadable information of the buffer feature, and the function returns **None**.

2) If getting the buffer feature value unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **set_buffer**

Statement:

```
BufferFeature.set_buffer(buf)
```

Significance:

Set the data of buffer feature

Formal parameter:

```
[in]buffer        Set the data buffer [buffer type]
```

Exception handling:

1) If the input parameter is not a buffer value, throw a **ParameterTypeError** exception.

2) If the buffer feature is not implemented or is not writable, then prints the information that the buffer feature is not writable, and the function returns **None**.

3) If the input buffer feature data length is greater than the maximum length, then prints the information that exceeding the maximum length of the buffer feature and prints the maximum value, and the function returns **None**.

4) If setting the buffer feature unsuccessfully, an exception is thrown. For details, see Error handling.

3.2.8. CommandFeature

It is responsible for the command type function of the camera, inherited from the Feature class.

Interface list:

is_implemented()	Test if the command feature is implemented
is_readable()	Test if the command feature is readable
is_writable ()	Test if the command feature is writable
send_command()	Send command

◆ **Interface description**

➢ **is_implemented**

See Feature::is_implemented() for details.

➢ **is_readable**

See Feature::is_readable() for details.

➢ **is_writable**

See Feature::is_writable() for details.

➢ **send_command**

Statement:

```
CommandFeature.send_command()
```

Significance:

Send command

Exception handling:

1) If the command feature is not implemented, then prints the information that does not support the command feature, and the function returns **None**.

2) If sending the command feature unsuccessfully, an exception is thrown. For details, see Error handling.

3.3. Data type definition

3.3.1. GxDeviceClassList

Definition	Value	Explanation
UNKNOWN	0	Unknown device type
USB2	1	USB2.0 camera
GEV	2	GigE Vision camera (GigE Vision)
U3V	3	USB3.0 camera (USB3 Vision)

3.3.2. GxAccessStatus

Definition	Value	Explanation
UNKNOWN	0	The current status of the device is unknown
READWRITE	1	The device is currently readable and writable
READONLY	2	The device currently only supports reading
NOACCESS	3	The device currently doesn't support reading and writing

3.3.3. GxAccessMode

Definition	Value	Explanation
READONLY	2	Open the device in read-only mode
CONTROL	3	Open the device in control mode
EXCLUSIVE	4	Open the device in exclusive mode

3.3.4. GxPixelFormatEntry

Definition	Value Value	Explanation
UNDEFINED	0x00000000	Undefined
MONO8	0x01080001	Monochrome 8-bit
MONO8_SIGNED	0x01080002	Monochrome 8-bit signed
MONO10	0x01100003	Monochrome 10-bit unpacked
MONO12	0x01100005	Monochrome 12-bit unpacked
MONO14	0x01100025	Monochrome 14-bit unpacked
MONO16	0x01100007	Monochrome 16-bit
BAYER_GR8	0x01080008	Bayer Green-Red 8-bit

BAYER_RG8	0x01080009	Bayer Red-Green 8-bit
BAYER_GB8	0x0108000A	Bayer Green-Blue 8-bit
BAYER_BG8	0x0108000B	Bayer Blue-Green 8-bit
BAYER_GR10	0x0110000C	Bayer Green-Red 10-bit
BAYER_RG10	0x0110000D	Bayer Red-Green 10-bit
BAYER_GB10	0x0110000E	Bayer Green-Blue 10-bit
BAYER_BG10	0x0110000F	Bayer Blue-Green 10-bit
BAYER_GR12	0x01100010	Bayer Green-Red 12-bit
BAYER_RG12	0x01100011	Bayer Red-Green 12-bit
BAYER_GB12	0x01100012	Bayer Green-Blue 12-bit
BAYER_BG12	0x01100013	Bayer Blue-Green 12-bit
BAYER_GR16	0x0110002E	Bayer Green-Red 16-bit
BAYER_RG16	0x0110002F	Bayer Red-Green 16-bit
BAYER_GB16	0x01100030	Bayer Green-Blue 16-bit
BAYER_BG16	0x01100031	Bayer Blue-Green 16-bit
RGB8_PLANAR	0x02180021	Red-Green-Blue 8-bit planar
RGB10_PLANAR	0x02300022	Red-Green-Blue 10-bit planar
RGB12_PLANAR	0x02300023	Red-Green-Blue 12-bit planar
RGB16_PLANAR	0x02300024	Red-Green-Blue 16-bit planar

3.3.5. GxFrameStatusList

Definition	Value	Explanation
SUCCESS	0	Normal frame
IMCOMPLETE	-1	Incomplete frame

3.3.6. GxDeviceTemperatureSelectorEntry

Definition	Value	Explanation
SENSOR	1	Sensor
MAINBOARD	2	Mainboard

3.3.7. GxPixelSizeEntry

Definition	Value	Explanation
BPP8	8	Pixel size of BPP8
BPP10	10	Pixel size of BPP10

BPP12	12	Pixel size of BPP12
BPP16	16	Pixel size of BPP16
BPP24	24	Pixel size of BPP24
BPP30	30	Pixel size of BPP30
BPP32	32	Pixel size of BPP32
BPP36	36	Pixel size of BPP36
BPP48	48	Pixel size of BPP48
BPP64	64	Pixel size of BPP64

3.3.8. GxPixelColorFilterEntry

Definition	Value	Explanation
NONE	0	None
BAYER_RG	1	RG format
BAYER_GB	2	GB format
BAYER_GR	3	GR format
BAYER_BG	4	BG format

3.3.9. GxAcquisitionModeEntry

Definition	Value	Explanation
SINGLE_FRAME	0	Single frame mode
MULITI_FRAME	1	Multi-frame mode
CONTINUOUS	2	Continuous mode

3.3.10. GxTriggerSourceEntry

Definition	Value	Explanation
SOFTWARE	0	Software trigger
LINE0	1	Trigger source 0
LINE1	2	Trigger source 1
LINE2	3	Trigger source 2
LINE3	4	Trigger source 3
COUNTER2END	5	COUNTER2END trigger

3.3.11. GxTriggerActivationEntry

Definition	Value	Explanation
FALLING_EDGE	0	Falling edge trigger
RISING_EDGE	1	Rising edge trigger

3.3.12. GxExposureModeEntry

Definition	Value	Explanation
TIMED	1	Exposure time register controls exposure time
TRIGGER_WIDTH	2	Trigger signal width controls exposure time

3.3.13. GxUserOutputSelectorEntry

Definition	Value	Explanation
OUTPUT0	1	Output 0
OUTPUT1	2	Output 1
OUTPUT2	4	Output 2

3.3.14. GxUserOutputModeEntry

Definition	Value	Explanation
STROBE	0	Strobe
USER_DEFINED	1	User defined

3.3.15. GxGainSelectorEntry

Definition	Value	Explanation
ALL	0	All gain channels
RED	1	Red channel gain
GREEN	2	Green channel gain
BLUE	3	Blue channel gain

3.3.16. GxBlackLevelSelectEntry

Definition	Value	Explanation
ALL	0	All black level channels
RED	1	Red channel black level
GREEN	2	Green channel black level
BLUE	3	Blue channel black level

3.3.17. GxBalanceRatioSelectorEntry

Definition	Value	Explanation
RED	0	Red channel
GREEN	1	Green channel
BLUE	2	Blue channel

3.3.18. GxAALightEnvironmentEntry

Definition	Value	Explanation
NATURE_LIGHT	0	Natural light
AC50HZ	1	50 Hz fluorescent lamp
AC60HZ	2	60 Hz fluorescent lamp

3.3.19. GxUserSetEntry

Definition	Value	Explanation
DEFAULT	0	Default parameter set
USER_SET0	1	User parameter set 0

3.3.20. GxAWBLampHouseEntry

Definition	Value	Explanation
ADAPTIVE	0	Adaptive light source
D65	1	The designated color temperature is 6500k
FLUORESCENCE	2	Designated fluorescent lamp
INCANDESCENT	3	Designated incandescent lamp
D75	4	The designated color temperature is 7500k
D50	5	The designated color temperature is 5000k
U30	6	The designated color temperature is 3000k

3.3.21. GxUserDataFieldSelectorEntry

Definition	Value	Explanation
FIELD_0	0	Flash data field 0
FIELD_1	1	Flash data field 1
FIELD_2	2	Flash data field 2
FIELD_3	3	Flash data field 3

3.3.22. GxTestPatternEntry

Definition	Value	Explanation
OFF	0	Off
GRAY_FRAME_RAMP_MOVING	1	Still grayscale increment
SLANT_LINE_MOVING	2	Rolling diagonal stripes
VERTICAL_LINE_MOVING	3	Rolling vertical stripes
HORIZONTAL_LINE_MOVING	4	Rolling horizontal stripes
GREY_VERTICAL_RAMP	5	Vertical grey increment
SLANT_LINE	6	Static slant line

3.3.23. GxTriggerSelectorEntry

Definition	Value	Explanation
FRAME_START	1	Get one frame
FRAME_BURST_START	2	Start the frame burst acquisition

3.3.24. GxLineSelectorEntry

Definition	Value	Explanation
LINE0	0	Pin 0
LINE1	1	Pin 1
LINE2	2	Pin 2
LINE3	3	Pin 3
LINE4	4	Pin 4
LINE5	5	Pin 5
LINE6	6	Pin 6
LINE7	7	Pin 7
LINE8	8	Pin 8
LINE9	9	Pin 9
LINE10	10	Pin 10
LINE_STROBE	11	Dedicated strobe pin

3.3.25. GxLineModeEntry

Definition	Value	Explanation
INPUT	0	Input

OUTPUT	1	Output

3.3.26. GxLineSourceEntry

Definition	Value	Explanation
OFF	0	Off
STROBE	1	Strobe
USER_OUTPUT0	2	User-defined output 0
USER_OUTPUT1	3	User-defined output 1
USER_OUTPUT2	4	User-defined output 2
EXPOSURE_ACTIVE	5	Active exposure
FRAME_TRIGGER_WAIT	6	Single frame trigger waiting
ACQUISITION_TRIGGER_WAIT	7	Multi-frame trigger waiting
TIMER1_ACTIVE	8	Active timer1
USER_OUTPUT3	9	User-defined output 3
USER_OUTPUT4	10	User-defined output 4
USER_OUTPUT5	11	User-defined output 5
USER_OUTPUT6	12	User-defined output 6

3.3.27. GxEventSelectorEntry

Definition	Value	Explanation
EXPOSURE_END	0x0004	End of exposure
BLOCK_DISCARD	0x9000	Image frame discarding
EVENT_OVERRUN	0x9001	Event queue overflow
FRAME_START_OVER_TRIGGER	0x9002	Trigger signal overflow
BLOCK_NOT_EMPTY	0x9003	Image frame memory is not empty
INTERNAL_ERROR	0x9004	Internal error events
FRAME_BURST_START_OVERTRIGGER	0x9005	Multi-frame event triggering shield
FRAME_START_WAIT	0x9006	Frame wait event
FRAME_BURST_START_WAIT	0x9007	Multi-frame wait event

3.3.28. GxLutSelectorEntry

Definition	Value	Explanation
LUMINANCE	0	Luminance

3.3.29. GxTransferControlModeEntry

Definition	Value	Explanation
BASIC	0	Basic mode
USER_CONTROLED	1	User control mode

3.3.30. GxTransferOperationModeEntry

Definition	Value	Explanation
MULTI_BLOCK	0	Designated the number of frames to send

3.3.31. GxTestPatternGeneratorSelectorEntry

Definition	Value	Explanation
SENSOR	0	The test image of sensor
REGION0	1	The test image of FPGA

3.3.32. GxChunkSelectorEntry

Definition	Value	Explanation
FRAME_ID	1	Frame ID
TIME_STAMP	2	Timestamp
COUNTER_VALUE	3	Counter value

3.3.33. GxBinningHorizontalModeEntry

Definition	Value	Explanation
SUM	0	The response from combine horizontal photo-sensitive cells will be added
AVERAGE	1	The response from combine horizontal photo-sensitive cells will be averaged

3.3.34. GxBinningVerticalModeEntry

Definition	Value	Explanation
SUM	0	The response from combine vertical photo-sensitive cells will be added
AVERAGE	1	The response from combine vertical photo-

		sensitive cells will be averaged

3.3.35. GxSensorShutterModeEntry

Definition	Value	Explanation
GLOBAL	0	All pixels are exposed simultaneously with same exposure time
ROLLING	1	All pixels have the same exposure time, but exposure start at different time
GLOBALRESET	2	All pixels start exposure at same time, but exposure time are different

3.3.36. GxAcquisitionStatusSelectorEntry

Definition	Value	Explanation
ACQUISITION_TRIGGER_WAIT	0	Acquisition trigger waiting
FRAME_TRIGGER_WAIT	1	Frame trigger waiting

3.3.37. GxExposureTimeModeEntry

Definition	Value	Explanation
ULTRASHORT	0	Ultra short mode
STANDARD	1	Standard mode

3.3.38. GxGammaModeEntry

Definition	Value	Explanation
SRGB	0	Default Gamma correction
USER	1	User-defined Gamma correction

3.3.39. GxLightSourcePresetEntry

Definition	Value	Explanation
OFF	0	Light source preset OFF
CUSTOM	1	Light source preset CUSTOM
DAYLIGHT_6500K	2	Light source preset DAYLIGHT(6500K)
DAYLIGHT_5000K	3	Light source preset DAYLIGHT(5000K)
COOL_WHITE_FLUORESCENCE	4	Light source preset COOL_WHITE_FLUORESCENCE(4150K)
INCA	5	Light source preset INCANDESCENT_A(2856K)

3.3.40. GxColorTransformationModeEntry

Definition	Value	Explanation
RGB_TO_RGB	0	Default color correction
USER	1	User-defined color correction

3.3.41. GxColorTransformationValueSelectorEntry

Definition	Value	Explanation
GAIN00	0	The gain value of color transformation component GAIN00
GAIN01	1	The gain value of color transformation component GAIN01
GAIN02	2	The gain value of color transformation component GAIN02
GAIN10	3	The gain value of color transformation component GAIN10
GAIN11	4	The gain value of color transformation component GAIN11
GAIN12	5	The gain value of color transformation component GAIN12
GAIN20	6	The gain value of color transformation component GAIN20
GAIN21	7	The gain value of color transformation component GAIN21
GAIN22	8	The gain value of color transformation component GAIN22

3.3.42. GxAutoEntry

Definition	Value	Explanation
OFF	0	Off
CONTINUOUS	1	Continuous
ONCE	2	Once

3.3.43. GxSwitchEntry

Definition	Value	Explanation
OFF	0	Off
ON	1	On

3.3.44. GxRegionSendModeEntry

Definition	Value	Explanation
SINGLE_ROI	0	Single ROI
MULTI_ROI	1	Multiple ROI

3.3.45. GxRegionSelectorEntry

Definition	Value	Explanation
REGION0	0	Region 0
REGION1	1	Region 1
REGION2	2	Region 2
REGION3	3	Region 3
REGION4	4	Region 4
REGION5	5	Region 5
REGION6	6	Region 6
REGION7	7·	Region 7

3.3.46. GxTimerSelectorEntry

Definition	Value	Explanation
TIMER1	1	Timer 1

3.3.47. GxTimerTriggerSourceEntry

Definition	Value	Explanation
EXPOSURE_START	1	Exposure start
LINE10	10	Start when receive pin 10 signal
STROBE	16	Start when receive strobe signal

3.3.48. GxCounterSelectorEntry

Definition	Value	Explanation
COUNTER1	1	Counter 1
COUNTER2	2	Counter 2

3.3.49. GxCounterEventSourceEntry

Definition	Value	Explanation
FRAME_START	1	Count the number of "FRAME_START" events
FRAME_TRIGGER	2	Count the number of "FRAME_TRIGGER" events
ACQUISITION_TRIGGER	3	Count the number of "ACQUISITION_TRIGGER" events
OFF	4	Off

SOFTWARE	5	Count the number of "SOFTWARE_TRIGGER" events
LINE0	6	Count the number of "LINE0_TRIGGER" events
LINE1	7	Count the number of "LINE1_TRIGGER" events
LINE2	8	Count the number of "LINE2_TRIGGER" events
LINE3	9	Count the number of "LINE3_TRIGGER" events

3.3.50. GxCounterResetSourceEntry

Definition	Value	Explanation
OFF	0	Counter reset off
SOFTWARE	1	Software trigger
LINE0	2	Pin 0
LINE1	3	Pin 1
LINE2	4	Pin 2
LINE3	5	Pin 3
COUNTER2END	6	COUNTER2END trigger

3.3.51. GxCounterResetActivationEntry

Definition	Value	Explanation
RISINGEDGE	1	Rising edge counter reset

3.3.52. GxCounterTriggerSourceEntry

Definition	Value	Explanation
OFF	0	Counter trigger off
SOFTWARE	1	Software trigger
LINE0	2	Pin 0
LINE1	3	Pin 1
LINE2	4	Pin 2
LINE3	5	Pin 3

3.3.53. GxTimerTriggerActivationEntry

Definition	Value	Explanation
RISINGEDGE	1	Rising edge counter reset

3.3.54. GxStopAcquisitionModeEntry

Definition	Value	Explanation
GENERAL	0	General
LIGHT	1	Light

3.3.55. GxDSStreamBufferHandlingModeEntry

Definition	Value	Explanation
OLDEST_FIRST	1	OldestFirst mode
OLDEST_FIRST_OVERWRITE	2	OldestFirstOverwrite mode
NEWEST_ONLY	3	NewestOnly mode

3.3.56. DxBayerConvertType

Definition	Value	Explanation
NEIGHBOUR	0	Neighborhood average interpolation algorithm
ADAPTIVE	1	Edge adaptive interpolation algorithm
NEIGHBOUR3	2	Neighborhood average interpolation algorithm for larger regions

3.3.57. DxValidBit

Definition	Value	Explanation
BIT0_7	0	0-7 bits
BIT1_8	1	1-8 bits
BIT2_9	2	2-9 bits
BIT3_10	3	3-10bits

3.3.58. DxImageMirrorMode

Definition	Value	Explanation
HORIZONTAL_MIRROR	0	Horizontal mirroring
VERTICAL_MIRROR	1	Vertical mirroring

3.4. Module interface definition

3.4.1. DeviceManager

It is responsible for the management of the camera, including enumerating device, opening device, getting the number of devices, etc.

Interface list:

update_device_list (timeout=200)	Enumerate devices on the same network segment
update_all_device_list (timeout=200)	Enumerate devices on different network segments
get_device_number ()	Get the number of devices
get_device_info ()	Get the information of devices
open_device_by_sn (sn, access_mode=GxAccessMode.CONTROL)	Open the device by serial number
open_device_by_user_id (user_id,access_mode=GxAccessMode.CONTROL)	Open the device by user ID
open_device_by_index (index, access_mode=GxAccessMode.CONTROL)	Open the device by device index
open_device_by_ip (ip, access_mode=GxAccessMode.CONTROL)	Open the device by IP address
open_device_by_mac (mac, access_mode=GxAccessMode.CONTROL)	Open the device by mac address

- **Interface description**

  - **update_device_list**

Statement:

```
DeviceManager.update_device_list (timeout=200)
```

Significance:

For non-GigE Vision cameras, enumerate all devices. For GigE Vision cameras, enumerate devices on the same network segment

Formal parameter:

```
[in]timeout        Enumeration timeout [0, 0xffffffff], the default
                   value is 200 (ms)
```

Return value:

The return value is the number of the devices which are enumerated and the list that records the

enumeration information. The number of elements in the device information list is the number of devices enumerated. The data type of the elements in the list is a dictionary, and the key names in the dictionary are detailed in the Enumeration device.

Exception handling:

1) If the input parameter is not an int value, a **ParameterTypeError** exception is thrown.

2) If the input parameter is less than 0 or greater than the maximum value of the unsigned int, then prints **DeviceManager.update_device_list: Out of bounds, timeout:minimum=0, maximum= 0xffffffff**, and the function returns **None**.

3) If the enumeration of device on the same segment fails, an exception is thrown. For details, see Error handling.

4) If getting basic information of all devices unsuccessfully, an exception is thrown. For details, see Error handling.

- **update_all_device_list**

Statement:

```
DeviceManager.update_all_device_list (timeout=200)
```

Significance:

For non-GigE Vision cameras, enumerate all devices, for GigE Vision cameras, enumerate network-wide devices

Formal parameter:

```
[in]timeout    Enumeration timeout [0, 0xffffffff], the default
               is 200 (ms)
```

Return value:

The return value is the number of the devices which is enumerated and the list that records the enumeration information. The number of elements in the device information list is the number of devices enumerated, the data type of the elements in the list is a dictionary, and the key names in the dictionary are detailed in the Enumeration device.

Exception handling:

1) If the input parameter is not an int value, a **ParameterTypeError** exception is thrown.

2) If the input parameter is less than 0 or greater than the maximum value of the unsigned int, then prints **DeviceManager.update_all_device_list: Out of bounds, timeout:minimum=0, maximum= 0xffffffff,** and the function returns **None**.

3) If the enumeration of device on the different segment fails, an exception is thrown. For details, see Error handling.

4) If getting basic information of all devices unsuccessfully, an exception is thrown. For details, see Error

handling.

- **get_device_number**

Statement:

```
DeviceManager.get_device_number ()
```

Significance:

Get the number of devices

Return value:

The number of devices

- **get_device_info**

Statement:

```
DeviceManager.get_device_info()
```

Significance:

Get the information of devices

Return value:

A list of device information. The number of elements in the device information list is the number of devices enumerated, the data type of the elements in the list is a dictionary, and the keys of the dictionary are detailed in the Enumeration device.

- **open_device_by_sn**

Statement:

```
DeviceManager.open_device_by_sn( sn,access_mode=GxAccessMode.CONTR
OL)
```

Significance:

Open the device by serial number

Formal parameter:

```
[in]sn              Serial number [string type]
[in]access_mode     Open device mode, the default is
                    GxAccessMode.CONTROL, check GxAccessMode
```

Return value:

Device object

Exception handling:

1) If the input parameter 1 is not a string, a **ParameterTypeError** exception is thrown.

2) If the input parameter 2 is not an int, a **ParameterTypeError** exception is thrown.

3) If input parameter 2 is not in device mode GxAccessMode, then prints interface name, open device mode is not in the range and the enumeration value information supported by the current parameter, and the function returns **None**.

4) If repeatedly getting the device class unsuccessfully, a **NotFoundDevice** exception is thrown.

5) If opening the device unsuccessfully, an exception is thrown. For details, see Error handling.

6) If the got device is not one of the U3V/USB2/GEV classes, a **NotFoundDevice** exception is thrown.

- **open_device_by_user_id**

Statement:

```
DeviceManager.open_device_by_user_id(user_id,access_mode=GxAccessMode.CONTROL)
```

Significance:

Open the device by user ID

Formal parameter:

[in]user_id	User ID [string type]
[in]access_mode	Open device mode, the default is GxAccessMode.CONTROL, check GxAccessMode

Return value:

Device object

Exception handling:

1) If the input parameter 1 is not a string, a **ParameterTypeError** exception is thrown.

2) If the input parameter 2 is not an int, a **ParameterTypeError** exception is thrown.

3) If the input parameter 2 is not in the open device mode GxAccessMode, then prints interface name, open device mode is not in the range and the enumeration value information supported by the current parameter, and the function returns **None**.

4) If repeatedly getting the device class unsuccessfully, a **NotFoundDevice** exception is thrown.

5) If opening the device unsuccessfully, an exception is thrown. For details, see Error handling.

6) If the got device is not one of the U3V/USB2/GEV classes, a **NotFoundDevice** exception is thrown.

➢ **open_device_by_index**

Statement:

```
DeviceManager.open_device_by_index(index,access_mode=GxAccessMode.
CONTROL)
```

Significance:

Open device by device index

Formal parameter:

[in]index	Device index [1,2,3...0xffffffff]
[in]access_mode	Open device mode, the default is GxAccessMode.CONTROL, check GxAccessMode

Return value:

Device object

Exception handling:

1) If the input parameter 1 or 2 is not an int value, a **ParameterTypeError** exception is thrown.

2) If the input parameter 1 is less than 0 or greater than the maximum value of the unsigned int, then prints **DeviceManager.open_device_by_index: index out of bounds, index: minimum=1, maximum= 0xffffffff**, and the function returns **None**.

3) If the input parameter 2 is not in device mode GxAccessMode, then prints interface name, open device mode is not in the range and information about the enumeration values supported by the current parameter, the function returns **None**.

4) If the number of devices is less than the index of the input parameter 1, a **NotFoundDevice** exception is thrown.

5) If opening the device unsuccessfully, an exception is thrown. For details, see Error handling.

6) If the got device is not one of the U3V/USB2/GEV classes, a **NotFoundDevice** exception is thrown.

➢ **open_device_by_ip**

Statement:

```
DeviceManager.open_device_by_ip(ip,access_mode=GxAccessMode.CONTRO
L)
```

Significance:

Open the GigE Vision camera by device ip address

Formal parameter:

[in]ip	Device ip address [string type]
[in]access_mode	Open device mode, the default is GxAccessMode.CONTROL, check GxAccessMode

Return value:

Device object

Exception handling:

1) If the input parameter 1 is not a string, a **ParameterTypeError** exception is thrown.

2) If the input parameter 2 is not an int, a **ParameterTypeError** exception is thrown.

3) If the input parameter 2 is not in device mode GxAccessMode, then prints interface name, open device mode is not in the range and information of the enumeration values supported by the current parameter, the function returns **None**.

4) If opening the device unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **open_device_by_mac**

Statement:

```
DeviceManager.open_device_by_mac(mac,access_mode=GxAccessMode.CONTROL)
```

Significance:

Open the GigE Vision camera by device mac address

Formal parameter:

[in]mac	Device mac address [string type]
[in]access_mode	Open device mode, the default is GxAccessMode.CONTROL, check GxAccessMode

Return value:

Device object

Exception handling:

1) If the input parameter 1 is not a string, a **ParameterTypeError** exception is thrown.

2) If the input parameter 2 is not an int, a **ParameterTypeError** exception is thrown.

3) If the input parameter 2 is not in device mode GxAccessMode, then prints interface name, open device mode is not in the range and the enumeration value information supported by the current parameter, the function returns **None**.

4) If opening the device unsuccessfully, an exception is thrown. For details, see Error handling.

3.4.2. Device

It is responsible for camera acquisition control, device close, configuration file import and export, and get the device handles, etc.

Interface list:

get_stream_channel_num()	Get the number of stream channels supported by the current device
stream_on ()	Send a start command, the camera starts transmitting image data
stream_off ()	Send an end command, the camera stops transmitting image data
export_config_file (file_path)	Export current configuration file
import_config_file (file_path, verify=False)	Import configuration file
close_device ()	Close device, destroy the device handle, and set the handle to none

◆ **Interface description**

➢ **get_stream_channel_num**

Statement:

```
Device.get_stream_channel_num()
```

Significance:

Get the number of stream channels supported by the current device

Return value:

The number of stream channels

Note: Currently, GigE Vision cameras, USB3.0, and USB2.0 cameras do not support multi-stream channels.

➢ **stream_on**

Statement:

```
Device.stream_on()
```

Significance:

Send a start command, the camera starts transmitting image data

Return value:

None

Exception handling:

1) If sending start command unsuccessfully, an exception is thrown. For details, see Error handling.

- **stream_off**

Statement:

```
Device.stream_off()
```

Significance:

Send a stop command, the camera stops transmitting image data

Return value:

None

Exception handling:

1) If sending stop command unsuccessfully, an exception is thrown. For details, see Error handling.

- **export_config_file**

Statement:

```
Device.export_config_file( file_path)
```

Significance:

Export the current configuration file

Formal parameter:

```
[in]file_path    File path
```

Return value:

None

Exception handling:

1) If the input parameter is not a string type, a **ParameterTypeError** exception is thrown.

2) If exporting the current configuration file unsuccessfully, an exception is thrown. For details, see Error handling.

- **import_config_file**

Statement:

```
Device.import_config_file(file_path, verify=False)
```

Significance:

Import the configuration file

Formal parameter:

```
[in]file_path     File path

[in]verify        Whether all imported values will be verified for
                  consistency, the default is False
```

Return value:

None

Exception handling:

1) If the input parameter 1 is not a string, a **ParameterTypeError** exception is thrown.

2) If the input parameter 2 is not a Bool, a **ParameterTypeError** exception is thrown.

3) If importing configuration file unsuccessfully, an exception is thrown. For details, see Error handling.

- **close_device**

Statement:

```
Device.close_device()
```

Significance:

Close the device, destroy the device handle, and set the handle to none

Return value:

None

Exception handling:

1) If closing the device unsuccessfully, an exception is thrown. For details, see Error handling.

Note:

if you still want to use this camera after you close the device, please reopen it.

➢ **register_device_offline_callback**

Statement:

```
Device.register_device_offline_callback(callback_func)
```

Significance:

Register the device offline callback function.

Formal parameter:

```
[in]callback_func    callback function
```

Return value:

None

Exception handling:

1) If the input parameter 1 is not in FunctionType, a **ParameterTypeError** exception is thrown.

2) If register offline callback unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **unregister_device_offline_callback**

Statement:

```
Device.unregister_device_offline_callback()
```

Significance:

Unregister the device offline callback.

Return value:

None

Exception handling:

1) If unregister offline callback unsuccessfully, an exception is thrown. For details, see Error handling.

3.4.3. DataStream

It is responsible for camera data stream setting, control, image acquisition, etc.

Interface list:

set_acquisition_buffer_number(buf_num)	Set the size of the acquisition buffer
get_image(timeout=1000)	Get the image and successfully create the image class object

flush_queue()	Clear camera acquire buffer queue

◆ **Interface description**

➢ **set_acquisition_buffer_number**

Statement:

```
DataStream.set_acquisition_buffer_number(buf_num)
```

Significance:

Set the size of the acquisition buffer

Formal parameter:

```
[in]buf_num   The length of the buffer address [1,0xffffffff]
```

Return value:

None

Exception handling:

1) If the input parameter is not an int, a **ParameterTypeError** exception is thrown.

2) If the input parameter is less than 1 or greater than the maximum value of the unsigned int, then prints **DataStream.set_acquisition_buffer_number: buf_num out of bounds, minimum=1, maximum=0xffffffff**, and the function returns **None.**

3) If setting the size of the acquisition buffer unsuccessfully, an exception is thrown. For details, see Error handling.

➢ **get_image**

Statement:

```
DataStream.get_image(timeout=1000)
```

Significance:

Get the image and successfully create the image class object.

Formal parameter:

```
[in]timeout    Get timeout [0,0xffffffff], the default is 1000 (ms)
```

Return value:

Image object: Get image successful

None: Timeout

Throw an exception: Other errors

Exception handling:

1) If the input parameter is not an int, a **ParameterTypeError** exception is thrown.

2) If the input parameter is less than 0 or greater than 0xffffffff, then prints **DataStream.get_image: timeout out of bounds, minimum=0, maximum=0xffffffff**, and the function returns **None**.

3) If getting the data size unsuccessfully, an exception is thrown, and the type of the exception is described in Error handling.

4) If getting the image unsuccessfully caused by timeout, the function returns **None**.

5) If not timed out but failed to get the image, then prints **status, DataStream, get_image**, and the function returns **None**.

- **flush_queue**

Statement:

```
DataStream.flush_queue()
```

Significance:

Clear the camera acquisition buffer queue

Exception handling:

1) If clearing the camera acquisition buffer queue unsuccessfully, an exception is thrown. For details, see Error handling.

- **register_capture_callback**

Statement:

```
DataStream.register_capture_callback(callback_func)
```

Significance:

Register the capture callback function

Formal parameter:

```
[in]callback_func    capture callback function
```

Return value:

None

Exception handling:

1) If the input parameter 1 is not in FunctionType, a **ParameterTypeError** exception is thrown.

2) If register capture callback unsuccessfully, an exception is thrown. For details, see Error handling.

Note:

Must register capture callback before start acquisition.

➢ **unregister_capture_callback**

Statement:

```
Device.unregister_capture_callback()
```

Significance:

Unregister the capture callback. Using callback

Return value:

None

Exception handling:

1) If unregister capture callback unsuccessfully, an exception is thrown. For details, see Error handling.

3.4.4. RGBImage

It is responsible for the operation of RGB image.

Interface list:

image_improvement(color_correction_param=0, contrast_lut=None, gamma_lut=None)	Improve image quality
saturation(factor)	Image saturation adjustment
sharpen(factor)	Sharpen the image
get_numpy_array()	Convert RGB data to numpy object
get_image_size()	Get RGB data size

◆ **Interface description**

➢ **image_improvement**

Statement:

```
RGBImage.image_improvement(color_correction_param=0,contrast_lut=None, gamma_lut=None)
```

Significance:

Improve image quality

Formal parameter:

```
[in]contrast_lut    Contrast LUT

[in]gamma_lut       Gamma LUT

[in]color_collect   Color correction
```

Exception handling:

1) If parameter 1 and 2 is not a buffer type or None, throw the exception **ParameterTypeError**.

2) If parameter 3 is not an int or None, throw the exception **ParameterTypeError**.

3) If the improvement of image quality is unsuccessful, throw the exception **UnexpectedError**.

4) If parameters 1, 2, and 3 are default values, the image quality improvement processing is not performed and the function exits.

- **saturation**

Statement:

```
RGBImage.saturation(factor)
```

Significance:

Saturation adjustment on RGB images

Formal parameter:

```
[in]factor     Saturation adjustment parameter, range: 0 ~ 128.
               Note: 64: no change in saturation
               >64: increase saturation
               <64: decrease saturation
               128: saturation is twice the current value
               0: monochrome image
```

Return value:

None

Exception handling:

1) If the image saturation adjustment fails, throw the exception **UnexpectedError**.

- **sharpen**

Statement:

```
RGBImage.sharpen()
```

Significance:

Sharpen the RGB image

Formal parameter:

```
[in]factor   Sharpen adjustment parameter, range: 0.1 ~ 5.0
```

Return value:

None

Exception handling:

1) If the image sharpening adjustment fails, throw the exception **UnexpectedError**.

- **get_numpy_array**

Statement:

```
RGBImage.get_numpy_array()
```

Significance:

Convert RGB data to numpy object

Return value:

Numpy object

- **get_image_size**

Statement:

```
RGBImage.get_image_size()
```

Significance:

Get the data size of RGB

Return value:

The size of RGB image

3.4.5. RawImage

It is responsible for the operation of Raw image.

Interface list:

convert(mode, flip=False,	Image format conversion

valid_bits=DxValidBit.BIT4_11, convert_type=DxBayerConvertType.NEIGHBOUR)	
defective_pixel_correct()	Image defective pixel correction
get_numpy_array()	Convert raw data to numpy object
get_data()	Get raw data
save_raw(file_path)	Save the data of raw image
get_status()	Get the status of raw image
get_width()	Get the width of raw image
get_height()	Get the height of raw image
get_pixel_format()	Get the pixel format of image
get_image_size()	Get the data size of raw image
get_frame_id()	Get frame ID
get_timestamp()	Get timestamp

◆ **Interface description**

➢ **convert**

Statement:

```
RawImage.convert(mode, flip=False, valid_bits=DxValidBit.BIT4_11,
convert_type=DxBayerConvertType.NEIGHBOUR)
```

Significance:

Image format conversion

1) When mode = 'RAW8', convert the 16-bit raw image to an 8-bit raw image. The valid bit of the interception defaults is the upper 8 bits of the current pixel format. The user can also manually set the valid bit with the parameter **valid_bits**. Only 10/12-bit raw images are supported.

2) When mode = 'RGB', convert the raw image to RGB image. If the input is a 10/12-bit raw image, it is first converted to an 8-bit raw image and then converted to an RGB image.

Formal parameter:

```
[in]mode 'RAW8': Convert 16-bit raw image to 8-bit raw image

         'RGB': Convert raw image to RGB24 image

[in]flip Whether the output RGB image is flipped, the default
         value is False, only supported in mode= 'RGB'

[in]valid_bits  Valid bits, the default is the upper 8 bits of the
```

```
                current pixel format, refer to DxValidBit
[in]convert_type    Convert type, the default value is
                    DxBayerConvertType.NEIGHBOUR, which refers to
                    DxBayerConvertType and is valid only for mode =
                    'RGB'
```

Return value:

RGB image object

Exception handling:

1) If the status of frame information is unsuccessful, then prints the error message **RawImage. convert: This is an incomplete image,** and the function returns **None**.

2) If parameter 1 is not a string type, throw the exception **ParameterTypeError**.

3) If parameter 2 is not a bool, throw the exception **ParameterTypeError**.

4) If parameters 3 and 4 are not int, throw the exception **ParameterTypeError**.

5) If parameter 4 is not in DxBayerConvertType, then prints the prompt parameter is out of bounds, the enumeration value supported by the current parameter, and the function returns **None**.

6) If parameter 4 is not in DxValidBit, then prints the prompt parameter is out of bounds, the enumeration value supported by the current parameter, and the function returns **None**.

7) If the pixel is not 8/10/12bit, the error message **RawImage.convert: This pixel format is not support** is printed, and the function returns **None**.

8) If parameter 1 is 'RAW8' and parameter 2 is **True**, the error message **RawImage.convert:mode = 'RAW8' don't support flip = True** is printed , and the function returns **None**.

9) Mode = 'RAW8', the bit depth is not 10/12bit, the error message **RawImage.convert: mode=RAW8 only support 10bit and 12bit** is printed, and the function returns **None**.

10) If parameter 1 is not 'RAW8' or 'RGB', then prints interface name and the input mode are not in the range, and the function returns **None**.

- **defective_pixel_correct**

Statement:

```
RawImage.defective_pixel_correct()
```

Significance:

Defective pixel correction on raw data

Return value:

None

Exception handling:

1) If the defective pixel correction is unsuccessful, throw the exception **UnexpetedError**.

- **get_numpy_array**

Statement:

```
RawImage.get_numpy_array()
```

Significance:

Convert the raw data to numpy object

Return value:

numpy object	Successful
None	Unsuccessful

Exception handling:

1) If the status of the frame information is unsuccessful, the error message **RawImage.get_numpy_array: This is an incomplete image** is printed, and the function returns **None**.

2) If the pixel format is not 8 or 16 bits, and the function returns **None**.

- **get_data**

Statement:

```
RawImage.get_data()
```

Significance:

Get raw data

Return value:

Raw data [string type]

- **save_raw**

Statement:

```
RawImage.save_raw( file_path)
```

Significance:

Save the data of raw image

Formal parameter:

```
[in]file_path   File path

For example: file_path = 'raw_image.raw', save the raw image to
the current project path. file_path = 'E:#
python_gxiapi/raw_image.raw', save the raw image to the absolute
path 'E:# python_gxiapi/'.
```

Return value:

None

Exception handling:

1) If the parameter is not a string type, an exception **ParameterTypeError** is thrown.

2) If saving the raw image data unsuccessfully, throw the exception **UnexpectedError**.

- **get_status**

Statement:

```
RawImage.get_status()
```

Significance:

Get the status of raw image

Return value:

The status of raw image, data type reference GxFrameStatusList

- **get_width**

Statement:

```
RawImage.get_width()
```

Significance:

Get the width of raw image

Return value:

The width of raw image

- **get_height**

Statement:

```
RawImage.get_height()
```

Significance:

Get the height of raw image

Return value:

The height of raw image

- **get_pixel_format**

Statement:

```
RawImage.get_pixel_format()
```

Significance:

Get the pixel format of image

Return value:

The pixel format

- **get_image_size**

Statement:

```
RawImage.get_image_size()
```

Significance:

Get the data size of raw image

Return value:

The size of raw image

- **get_frame_id**

Statement:

```
RawImage.get_frame_id()
```

Significance:

Get frame ID

Return value:

Frame ID

➢ **get_timestamp**

Statement:

```
RawImage.get_timestamp()
```

Significance:

Get timestamp

Return value:

Timestamp

3.4.6. Buffer

It is responsible for the operation of the buffer class. The buffer class will be used in the image quality improvement section. The buffer type object returned by the Utility.get_gamma_lut(gamma) and Utility.get_contrast_lut(contrast) interfaces will be passed as an parameter to RGBImage.image_improvement(color_correction_param=0, contrast_lut=None, gamma_lut=None) interface.

Interface list:

from_file(file_name)	Get buffer object from a file
from_string(string_data)	Get buffer object from a string
get_data()	Return the string data of the buffer object
get_ctype_array()	Return the data array of the buffer object
get_numpy_array()	Return the numpy array of buffer objects
get_length()	Return the length of the data array of the buffer object

◆ **Interface description**

➢ **from_file (Static function)**

Statement:

```
Buffer.from_file(file_name)
```

Significance:

Get buffer object from a file

Formal parameter:

```
[in]file_name     File path
```

Return value:

Buffer object

- **from_string (Static function)**

Statement:

```
Buffer.from_string(string_data)
```

Significance:

Get buffer object from a string

Formal parameter:

```
[in]string_data      String
```

Return value:

Buffer object

- **get_data**

Statement:

```
Buffer.get_data()
```

Significance:

Returns the buffer object

Return value:

string_data String data

Note: Python2.7: returns a string type. Python3.5: returns a **bytes** type.

- **get_ctype_array**

Statement:

```
Buffer.get_ctype_array()
```

Significance:

Return the data array of the buffer object

Return value:

The data array of the buffer object [ctype type]

➢ **get_numpy_array**

Statement:

```
Buffer.get_numpy_array()
```

Significance:

Returns the numpy array of buffer object

Return value:

The data array of buffer object [numpy style]

➢ **get_length**

Statement:

```
Buffer.get_length()
```

Significance:

Return the length of the data array of the buffer object

Return value:

The length of the data array

3.4.7. Utility

It is responsible for the operation of gamma and contrast parameters.

Interface list:

get_gamma_lut(gamma=1)	Get gamma LUT by gamma value
get_contrast_lut(contrast=0)	Get contrast LUT by the contrast value

◆ **Interface description**

➢ **get_gamma_lut (Static function)**

Statement:

```
Utility.get_gamma_lut(gamma=1) (Static function)
```

Significance:

Get gamma LUT by gamma value

Formal parameter:

```
[in]gamma   Int or float, range [0.1, 10.0], the default is 1
```

Return value:

Gamma LUT

Exception handling:

1) If the input parameter is not an int or a float, a **ParameterTypeError** exception is thrown.

2) If the input parameter is not in the range of 0.1 to 10.0, the error message **Utility.get_gamma_lut:gamma out of bounds, range:[0.1, 10.0]** is printed, and the function returns **None**.

3) If getting the gamma LUT unsuccessfully, the interface name, the **gamma lut** fail to get, and the error code are printed. The function returns **None**.

- **get_contrast_lut (Static function)**

Statement:

```
Utility.get_contrast_lut(contrast=0)(Static function)
```

Significance:

Get contrast LUT by the contrast value

Formal parameter:

```
[in]contrast    Int, range [-50, 100], the default is 0
```

Return value:

Contrast LUT

Exception handling:

1) If the input parameter is not an int, a **ParameterTypeError** exception is thrown.

2) If the input parameter is not in the range of -50 to 100, the error message **Utility.get_contrast_lut:contrast out of bounds, range:[-50, 100]** is printed, and the function returns **None**.

3) If getting contrast LUT unsuccessfully, the interface name, the **gamma lut** fail to get, and the error code are printed, and the function returns **None.**

4. FAQ

No.	General Question	Answer
1	The following error occurred during program execution: **NotInitApi: DeviceManager.update_device_list: {-13}{Not init API}**	1) Please check and delete the statement of the **__del__ ()** function that calls the **DeviceManager** class object in the program. Because Python's garbage collection mechanism automatically calls the **__del__()** function to destroy the object, it does not need or allow the user to display the call to the **__del__()** function, such as: **device_manager.__del__()**.

5. Revision History

No.	Version	Changes	Date
1	V1.0.0	Initial release	2018-08-10
2	V1.0.1	Add the new function description for the Mercury2 cameras	2018-10-29
3	V1.0.2	Modify some subtitles and inaccurate descriptions	2019-04-16
4	V1.0.3	Add some descriptions	2019-05-07
5	V1.0.4	Add offline callback and capture callback register, unregister function, Synchronize new feature ID and corresponding Entrys	2021-03-04
6	V1.0.5	Modify some description issues	2021-03-08