

# Python Interface Development User Manual

Version: V1.0.6

Date: 2021-06-03

All rights reserved. No parts of this manual may be used or reproduced, in any forms or by any means, without prior written permission of China Daheng Group, Inc. Beijing Image Vision Technology Branch.

The right is also reserved to modify or change any parts of this manual in the future without prior notification.

All other trademarks are the properties of their respective owners.

© 2021 China Daheng Group, Inc. Beijing Image Vision Technology Branch

Web: [www.daheng-imaging.com/en](http://www.daheng-imaging.com/en)

Sales Email: [isales@daheng-imaging.com](mailto:isales@daheng-imaging.com)

Sales Tel: +86 10 8282 8878

Support Email: [isupport@daheng-imaging.com](mailto:isupport@daheng-imaging.com)

# Contents

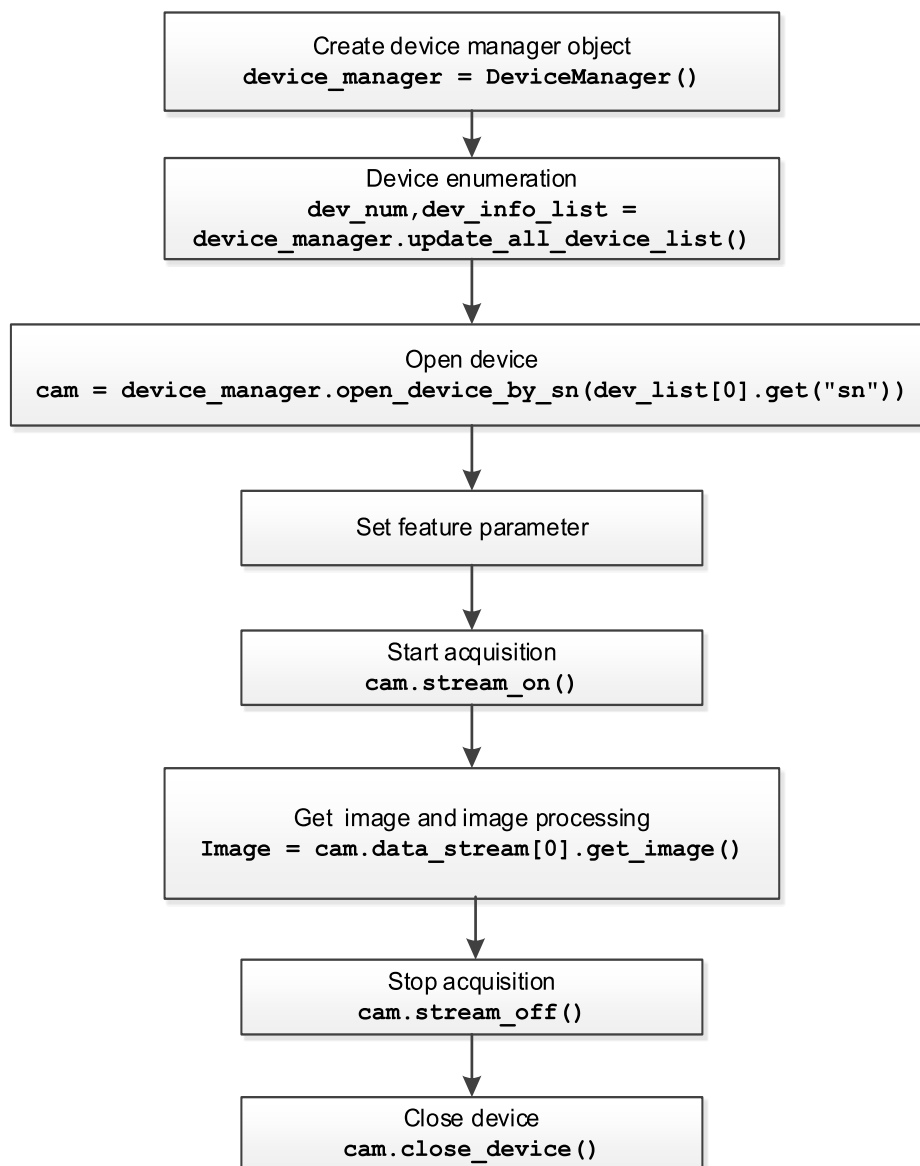
<b>1. Camera Workflow .....</b>	<b>1</b>
1.1. Overall workflow .....	1
1.2. Function control flow .....	2
1.3. Overall code sample.....	3
<b>2. Programming Guide .....</b>	<b>4</b>
2.1. Build programming environment.....	4
2.1.1. Linux .....	4
2.1.2. Windows .....	5
2.2. QuickStart.....	6
2.2.1. Importing library .....	6
2.2.2. Enumeration device.....	6
2.2.3. Open or close the device.....	7
2.2.4. Acquisition control.....	8
2.2.5. Image processing .....	9
2.2.6. Camera control .....	11
2.2.7. Import and export camera configuration parameter.....	15
2.2.8. Error handling .....	15
<b>3. Appendix .....</b>	<b>17</b>
3.1. Feature parameter.....	17
3.1.1. Device feature parameter .....	17
3.1.2. Stream feature parameter .....	27
3.2. Function class definition .....	28
3.2.1. Feature .....	28
3.2.2. IntFeature .....	29
3.2.3. FloatFeature .....	31
3.2.4. EnumFeature .....	33
3.2.5. BoolFeature .....	35
3.2.6. StringFeature .....	36
3.2.7. BufferFeature.....	38
3.2.8. CommandFeature.....	40
3.3. Data type definition.....	41
3.3.1. GxDeviceClassList .....	41
3.3.2. GxAccessStatus .....	41
3.3.3. GxAccessMode .....	42
3.3.4. GxPixelFormatEntry .....	42
3.3.5. GxFrameStatusList.....	43
3.3.6. GxDeviceTemperatureSelectorEntry .....	43
3.3.7. GxPixelSizeEntry.....	43
3.3.8. GxPixelColorFilterEntry .....	43
3.3.9. GxAcquisitionModeEntry .....	44
3.3.10. GxTriggerSourceEntry.....	44
3.3.11. GxTriggerActivationEntry.....	44
3.3.12. GxExposureModeEntry .....	44

3.3.13. GxUserOutputSelectorEntry .....	44
3.3.14. GxUserOutputModeEntry .....	44
3.3.15. GxGainSelectorEntry .....	45
3.3.16. GxBlackLevelSelectEntry .....	45
3.3.17. GxBalanceRatioSelectorEntry .....	45
3.3.18. GxAALightEnvironmentEntry .....	45
3.3.19. GxUserSetEntry .....	45
3.3.20. GxAWBLampHouseEntry .....	45
3.3.21. GxUserDataFieldSelectorEntry .....	46
3.3.22. GxTestPatternEntry .....	46
3.3.23. GxTriggerSelectorEntry .....	46
3.3.24. GxLineSelectorEntry .....	46
3.3.25. GxLineModeEntry .....	47
3.3.26. GxLineSourceEntry .....	47
3.3.27. GxEventSelectorEntry .....	48
3.3.28. GxLutSelectorEntry .....	48
3.3.29. GxTransferControlModeEntry .....	48
3.3.30. GxTransferOperationModeEntry .....	48
3.3.31. GxTestPatternGeneratorSelectorEntry .....	48
3.3.32. GxChunkSelectorEntry .....	48
3.3.33. GxBinningHorizontalModeEntry .....	49
3.3.34. GxBinningVerticalModeEntry .....	49
3.3.35. GxSensorShutterModeEntry .....	49
3.3.36. GxAcquisitionStatusSelectorEntry .....	49
3.3.37. GxExposureTimeModeEntry .....	49
3.3.38. GxGammaModeEntry .....	50
3.3.39. GxLightSourcePresetEntry .....	50
3.3.40. GxColorTransformationModeEntry .....	50
3.3.41. GxColorTransformationValueSelectorEntry .....	50
3.3.42. GxAutoEntry .....	51
3.3.43. GxSwitchEntry .....	51
3.3.44. GxRegionSendModeEntry .....	51
3.3.45. GxRegionSelectorEntry .....	51
3.3.46. GxTimerSelectorEntry .....	51
3.3.47. GxTimerTriggerSourceEntry .....	51
3.3.48. GxCounterSelectorEntry .....	52
3.3.49. GxCounterEventSourceEntry .....	52
3.3.50. GxCounterResetSourceEntry .....	52
3.3.51. GxCounterResetActivationEntry .....	52
3.3.52. GxCounterTriggerSourceEntry .....	53
3.3.53. GxTimerTriggerActivationEntry .....	53
3.3.54. GxStopAcquisitionModeEntry .....	53
3.3.55. GxDSSStreamBufferHandlingModeEntry .....	53
3.3.56. GxResetDeviceModeEntry .....	53
3.3.57. Dx Bayer Convert Type .....	53
3.3.58. DxValidBit .....	54

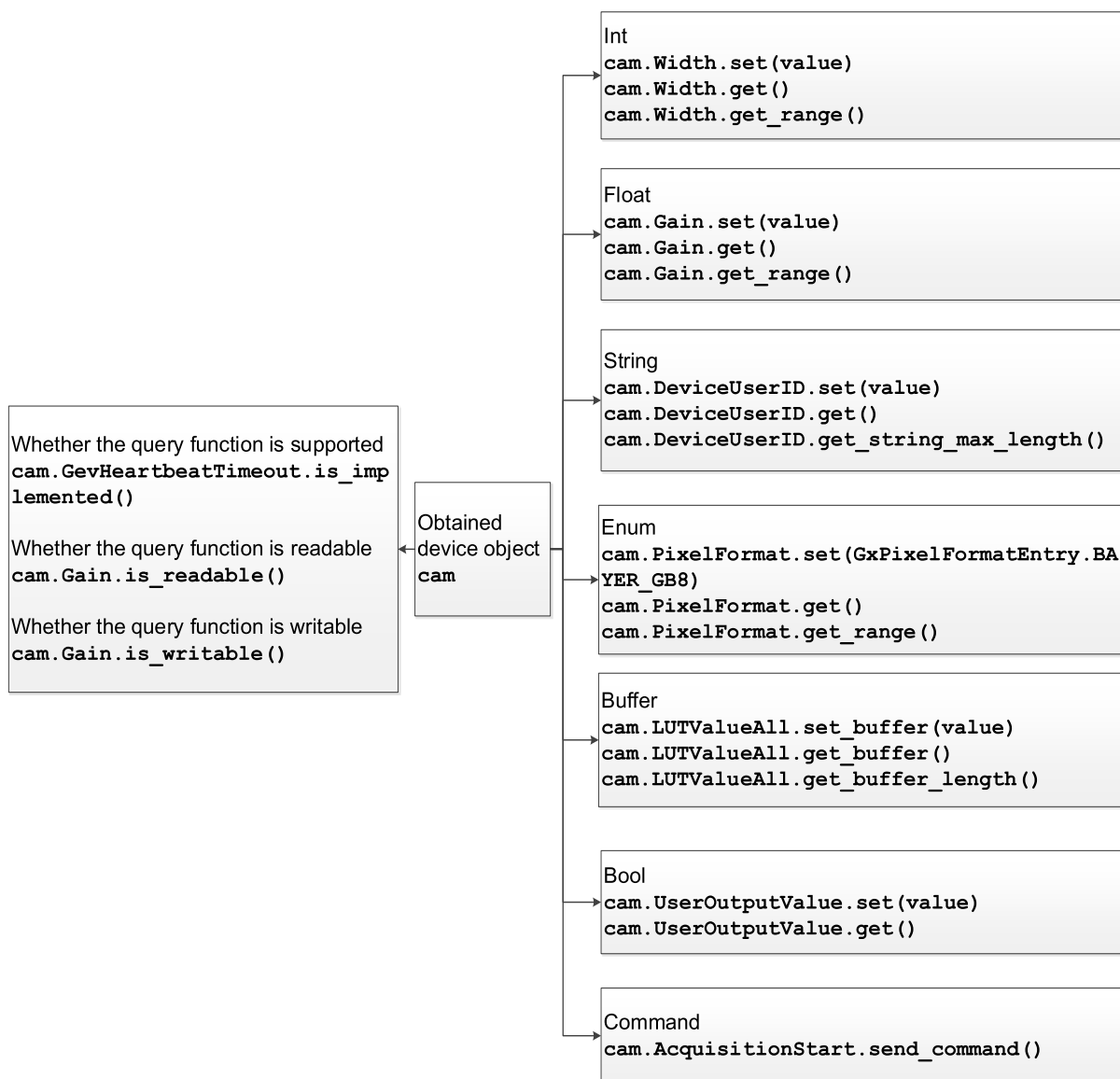
3.3.59. DxImageMirrorMode .....	54
3.3.60. DxRGBChannelOrder .....	54
3.4. Module interface definition .....	54
3.4.1. DeviceManager .....	54
3.4.2. Device .....	61
3.4.3. DataStream .....	64
3.4.4. RGBImage .....	67
3.4.5. RawImage .....	71
3.4.6. Buffer .....	80
3.4.7. Utility .....	82
<b>4. FAQ .....</b>	<b>86</b>
<b>5. Revision History .....</b>	<b>87</b>

## 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

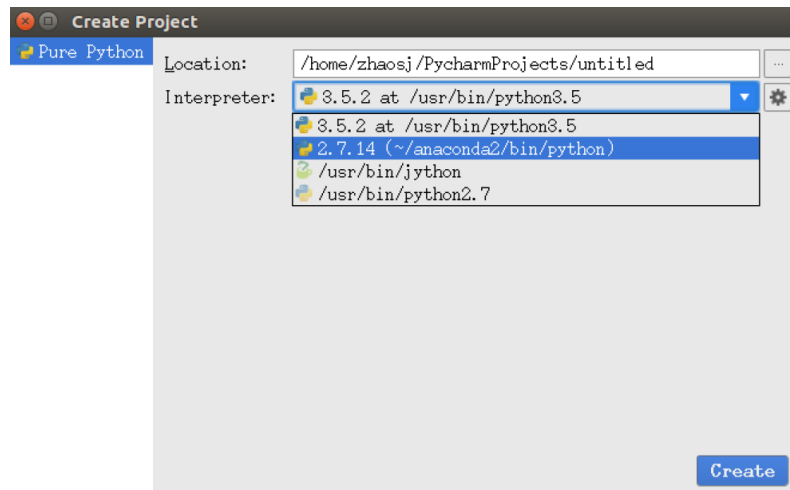
It is recommended that users use Python 2.7 and Python 3.5 first. This interface library has been tested in the above two versions.

#### 2.1.1. Linux

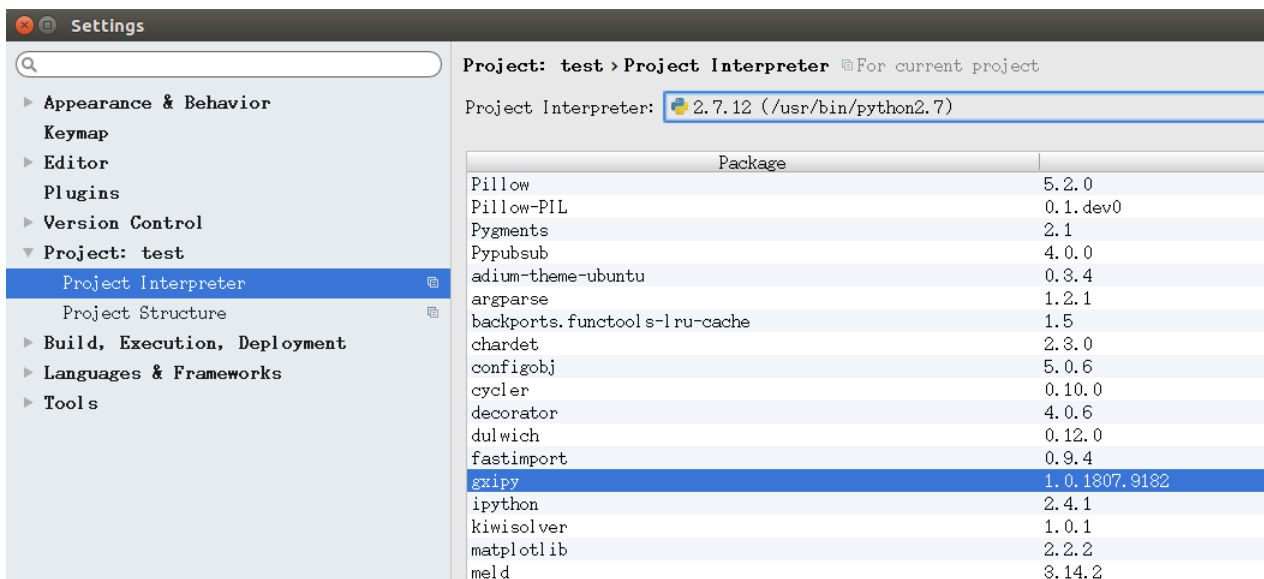
- 1) Install **pycharm-community** (community free version).

```
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.



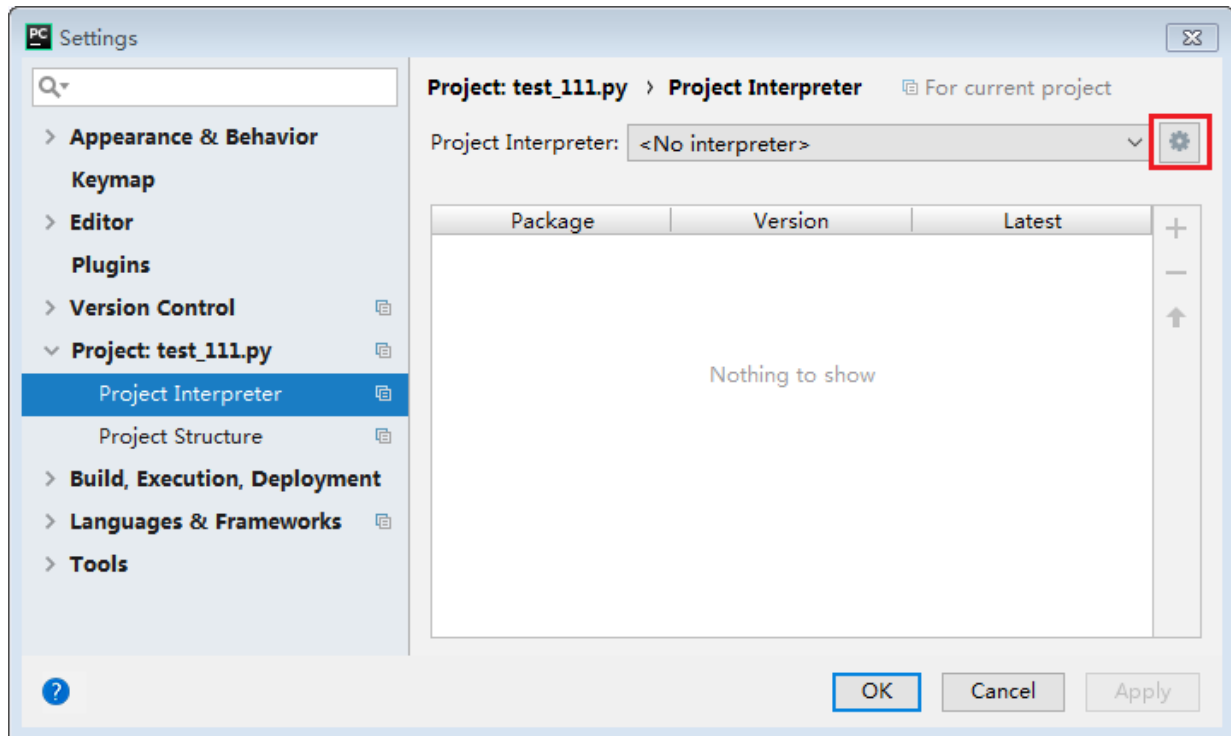
- 3) **File->Settings->Project->Project Interpreter** can check the **Package** installed by the Python interpreter. As shown in the figure, the **gxipy** library is installed normally and can be imported by user's program.



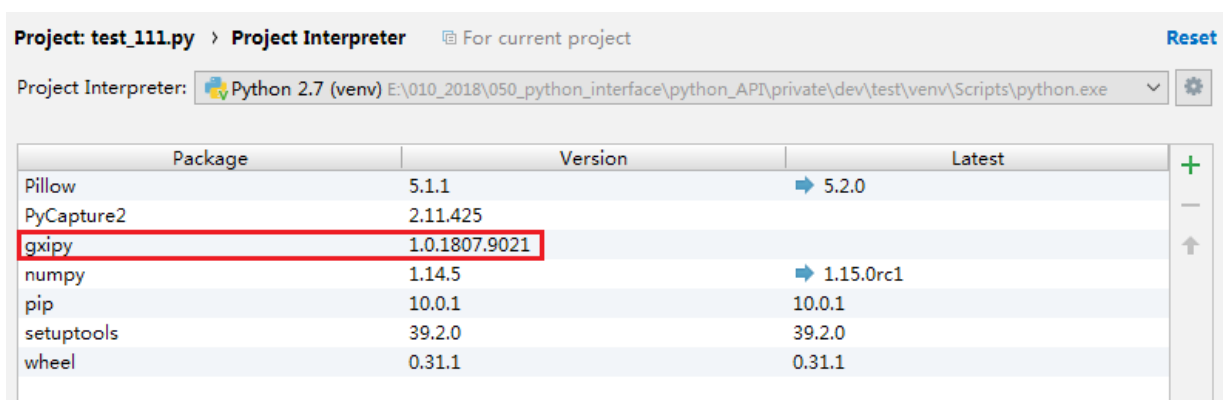
### 2.1.2. Windows

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	<a href="#">GxAccessStatus</a>
device_class	Device class	<a href="#">GxDeviceClassList</a>
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 gxi 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:

<a href="#">Feature.is_implement</a>	Is this current feature controller support this feature
<a href="#">Feature.is_readable</a>	Is this feature readable



<a href="#">Feature.is_writable</a>	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](#).

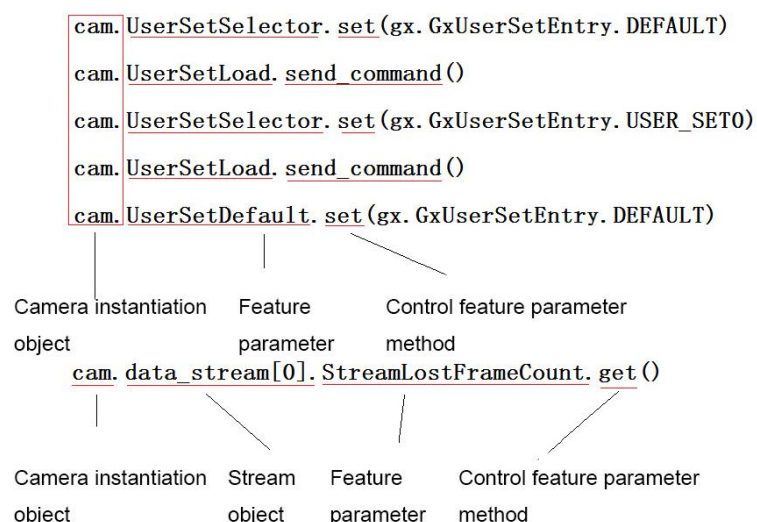
The difference is that the control functions of the feature parameters are different:

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

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

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



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

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

AWBLampHouse	Automatic white balance lamp house, see <a href="#">GxAWBLampHouseEntry</a> for details	<a href="#">EnumFeature</a>
SharpnessMode	Sharpening model, see <a href="#">GxSwitchEntry</a> for details	<a href="#">EnumFeature</a>
Sharpness	Sharpness	<a href="#">FloatFeature</a>
FrameInformation	Image frame information	<a href="#">BufferFeature</a>
DataFieldSelector	User selects the flash data field, see <a href="#">GxUserDataFieldSelectorEntry</a> for details	<a href="#">EnumFeature</a>
DataFieldValue	User data field content	<a href="#">BufferFeature</a>
FlatFieldCorrection	Flat field correction switch, see <a href="#">GxSwitchEntry</a> for details	<a href="#">EnumFeature</a>
NoiseReductionMode	Noise reduction switch, see <a href="#">GxSwitchEntry</a> for details	<a href="#">EnumFeature</a>
NoiseReduction	Noise reduction value	<a href="#">FloatFeature</a>
FFCLoad	Get flat field correction parameters	<a href="#">BufferFeature</a>
FFCSave	Set flat field correction parameters	<a href="#">BufferFeature</a>
StaticDefectCorrection	Static dead pixel correction switch, see <a href="#">GxSwitchEntry</a> for details	<a href="#">EnumFeature</a>
<b>UserSetControl Section</b>		
UserSetLoad	Load the user set	<a href="#">CommandFeature</a>
UserSetSave	Save the user set	<a href="#">CommandFeature</a>
UserSetSelector	User set selector, see C Software Development Manual and <a href="#">GxUserSetEntry</a> for details	<a href="#">EnumFeature</a>
UserSetDefault	Default the user set, see <a href="#">GxUserSetEntry</a> for details	<a href="#">EnumFeature</a>
<b>Event Section</b>		
EventSelector	Event source select, see <a href="#">GxEventSelectorEntry</a> for details	<a href="#">EnumFeature</a>
EventNotification	Switch of the notification to the host application of the occurrence of the selected Event, see <a href="#">GxSwitchEntry</a> for details	<a href="#">EnumFeature</a>
EventExposureEnd	Exposure end event	<a href="#">IntFeature</a>

EventExposureEndTimestamp	The timestamp of Exposure end event	<a href="#">IntFeature</a>
EventExposureEndFrameID	The frame id of Exposure end event	<a href="#">IntFeature</a>
EventBlockDiscard	Block discard event	<a href="#">IntFeature</a>
EventBlockDiscardTimestamp	The timestamp of Block discard event	<a href="#">IntFeature</a>
EventOverrun	Event queue overflow event	<a href="#">IntFeature</a>
EventOverrunTimestamp	The timestamp of event queue overflow event	<a href="#">IntFeature</a>
EventFrameStartOvertrigger	Trigger signal shield event	<a href="#">IntFeature</a>
EventFrameStartOvertriggerTimestamp	The timestamp of trigger signal shield event	<a href="#">IntFeature</a>
EventBlockNotEmpty	Frame memory not empty event	<a href="#">IntFeature</a>
EventBlockNotEmptyTimestamp	The timestamp of frame memory not empty event	<a href="#">IntFeature</a>
EventInternalError	Internal erroneous event	<a href="#">IntFeature</a>
EventInternalErrorTimestamp	The timestamp of internal erroneous event	<a href="#">IntFeature</a>
EventFrameBurstStartOvertrigger	Frame burst start overtrigger event ID	<a href="#">IntFeature</a>
EventFrameBurstStartOvertriggerFrame ID	Frame burst start overtrigger event frame ID	<a href="#">IntFeature</a>
EventFrameBurstStartOvertriggerTimestamp	Frame burst start overtrigger event timestamp	<a href="#">IntFeature</a>
EventFrameStartWait	Frame start wait event ID	<a href="#">IntFeature</a>
EventFrameStartWaitTimestamp	Frame start wait event timestamp	<a href="#">IntFeature</a>
EventFrameBurstStartWait	Frame burst start wait event ID	<a href="#">IntFeature</a>
EventFrameBurstStartWaitTimestamp	Frame burst start wait event timestamp	<a href="#">IntFeature</a>
EventBlockDiscardFrameID	Data block discard event frame ID	<a href="#">IntFeature</a>
EventFrameStartOvertriggerFrameID	Frame start wait overtrigger event frame ID	<a href="#">IntFeature</a>
EventBlockNotEmptyFrameID	Data block not empty event frame ID	<a href="#">IntFeature</a>
EventFrameStartWaitFrameID	Frame start wait event frame ID	<a href="#">IntFeature</a>
EventFrameBurstStartWaitFrameID	Frame burst start wait event frame ID	<a href="#">IntFeature</a>
<b>LUT Section</b>		
LUTValueAll	LUT content	<a href="#">BufferFeature</a>

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

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

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

CounterTriggerSource	Counter trigger source, see <a href="#">GxCounterTriggerSourceEntry</a> for details	<a href="#">EnumFeature</a>
CounterDuration	Counter duration value	<a href="#">IntFeature</a>
TimerTriggerActivation	Timer Trigger Activation, see <a href="#">GxTimerTriggerActivationEntry</a> for details	<a href="#">EnumFeature</a>
<b>RemoveParameterLimitControl Section</b>		
RemoveParameterLimit	Remove parameter range restriction switch, see <a href="#">GxSwitchEntry</a> for details	<a href="#">EnumFeature</a>

### 3.1.2. Stream feature parameter

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



MaxNumQueueBuffer	Maximum number of Buffer in the acquisition queue	<a href="#">IntFeature</a>
PacketTimeout	Packet timeout time	<a href="#">IntFeature</a>
StreamTransferSize	Transfer data block size	<a href="#">IntFeature</a>
StreamTransferNumberUrb	Transfer data block number	<a href="#">IntFeature</a>
SocketBufferSize	Socket buffer size in kilobytes	<a href="#">IntFeature</a>
StopAcquisitionMode	Stop acquisition mode, see <a href="#">GxStopAcquisitionModeEntry</a> for details	<a href="#">EnumFeature</a>
StreamBufferHandlingMode	Buffer handling mode see <a href="#">GxDSSStreamBufferHandlingModeEntry</a> for details	<a href="#">EnumFeature</a>

## 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:**

True: 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:**

True: 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:**

True: 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 enum 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 enum 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_OVERT RIGGER	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_CONTROLLED	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. GxDSSStreamBufferHandlingModeEntry

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

## 3.3.56. GxResetDeviceModeEntry

Definition	Value	Explanation
RECONNECT	1	Reconnect device
RESET	2	Reset device

## 3.3.57. Dx Bayer Convert Type

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

## 3.3.58. 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

## 3.3.59. DxImageMirrorMode

Definition	Value	Explanation
HORIZONTAL_MIRROR	0	Horizontal mirroring
VERTICAL_MIRROR	1	Vertical mirroring

## 3.3.60. DxRGBChannelOrder

Definition	Value	Explanation
ORDER_RGB	0	RGB channel
ORDER_BGR	1	BGR channel

## 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

<code>open_device_by_ip(ip, access_mode=GxAccessMode.CONTROL)</code>	Open the device by IP address
<code>open_device_by_mac(mac, access_mode=GxAccessMode.CONTROL)</code>	Open the device by mac address
<code>gige_reset_device(mac_address, reset_device_mode)</code>	Reconnect device or reset device

#### ◆ 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:

<code>[in] timeout</code>	Enumeration timeout <code>[0, 0xffffffff]</code> , 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.CONTROL)
```

**Significance:**

Open the device by serial number

**Formal parameter:**

[in] sn	Serial number [string type]
[in] access_mode	Open device mode, the default is <a href="#">GxAccessMode.CONTROL</a> , check <a href="#">GxAccessMode</a>

**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.CONTROL)
```

**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](#).

#### ➤ gige\_reset\_device

#### Statement:

```
DeviceManager.gige_reset_device(mac_address, reset_device_mode)
```

#### Significance:

Reconnect or reset device

Reconnect device is usually used when debugging a GigE camera. The program is abnormal when the device has been opened, then reopen the device and an error is reported (because the debugging heartbeat is 5 minutes, the device is still open). Then you can through the reconnect device function to make the device in the unopened state and then reopen the device again to succeed.

Reset device is usually used when the camera status is abnormal. At this time, the reconnect device function is not work, and there is no condition to power on the device again. You can try to use reset device function to power off the device and power on again. After reset device, you need to enumerate device and turn on the device again.

#### Note:

- 1) The reset time is about 1s, ensure that the enumeration interface is called after 1s.
- 2) If the device is in acquisition, forbidden to use reset device and reconnect device functions which will offline the device.

#### Formal parameter:

```
[in]mac address          Device mac address [string type]
[in]reset_device_mode Reset device mode, refer to GxResetDeviceModeEntry
```

**Return value:**

None

**Exception handling:**

- 1) If execute the command is failed, 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, channel_order=DxRGBChannelOrder.ORDER_RGB)	Improve image quality
brightness(factor)	Image brightness adjustment
contrast(factor)	Image contrast adjustment
saturation(factor)	Image saturation adjustment
sharpen(factor)	Sharpen the image
get_white_balance_ratio()	Get white balance ratio
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, channel_order=DxRGBChannelOrder.ORDER_RGB)
```

**Significance:**

Improve image quality

**Formal parameter:**

```
[in] contrast_lut    Contrast LUT  
[in] gamma_lut      Gamma LUT  
[in] color_collect   Color correction  
[in] channel_order   Image channel order, the default is  
                     DxRGBChannelOrder.ORDER_RGB, refer to  
                     DxRGBChannelOrder
```

**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 parameter 4 is not an int, throw the exception **ParameterTypeError**.
- 4) If the improvement of image quality is unsuccessful, throw the exception **UnexpectedError**.
- 5) If parameters 1, 2, and 3 are default values, the image quality improvement processing is not performed and the function exits.

**➤ brightness****Statement:**

```
RGBImage.brightness(factor)
```

**Significance:**

Brightness adjustment on RGB24 images

**Formal parameter:**

```
[in] factor    Brightness adjustment parameter, range: -150 ~ 150.  
               Note: 0: no change in brightness  
                   >0: increase brightness  
                   <0: decrease brightness
```

**Return value:**

None

**Exception handling:**

- 1) If the input parameter is not an int, throw the exception **ParameterTypeError**.
- 2) If the image brightness adjustment fails, throw the exception **UnexpectedError**.

**➤ contrast****Statement:**

```
RGBImage.contrast(factor)
```

**Significance:**

Contrast adjustment on RGB images

**Formal parameter:**

```
[in] factor      Contrast adjustment parameter, range: -50 ~ 150.  
                Note: 0: no change in contrast  
                    >0: increase contrast  
                    <0: decrease contrast
```

**Return value:**

None

**Exception handling:**

- 1) If the input parameter is not an int, throw the exception **ParameterTypeError**.
- 2) If the image contrast adjustment fails, throw the exception **UnexpectedError**.

**➤ 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\_white\_balance\_ratio**

**Statement:**

```
RGBImage.get_white_balance_ratio()
```

**Significance:**

Get the white balance ratio

**Return value:**

RGB component coefficient tuple

➤ **get\_numpy\_array**

**Statement:**

```
RGBImage.get_numpy_array()
```

**Significance:**

Get 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, valid_bits=DxValidBit.BIT4_11, convert_type=DxBayerConvertType.NEIGHBOUR, channel_order=DxRGBChannelOrder.ORDER_RGB)	Image format conversion
defective_pixel_correct()	Image defective pixel correction
raw8_rotate_90_cw()	8-bit image 90 degree clockwise rotate
raw8_rotate_90_ccw()	8-bit image 90 degree counterclockwise rotate
brightness(factor)	8-bit gray scale image brightness adjustment
contrast(factor)	8-bit gray scale image contrast adjustment
mirror(mirror_mode)	Mirroring 8-bit image
get_ffc_coefficients(dark_img=None, target_value=None)	Calculate Flat Field Correction (FFC) coefficients
flat_field_correction(ffc_coefficients)	Execute FFC
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,
channel_order=DxRGBChannelOrder.ORDER_RGB)
```

##### 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'

[in]channel_order Image channel order, the default value is
DxRGBChannelOrder.ORDER_RGB, which refers to
DxRGBChannelOrder
```

##### 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**.

➤ **raw8\_rotate\_90\_cw**

**Statement:**

```
RawImage.raw8_rotate_90_cw()
```

**Significance:**

8-bit image 90 degree clockwise rotate

**Return value:**

Rotated RawImage image object

**Exception handling:**

1) If the rotate is unsuccessful, throw the exception **UnexpetedError**.

➤ **raw8\_rotate\_90\_ccw**

**Statement:**

```
RawImage.raw8_rotate_90_ccw()
```

**Significance:**

8-bit image 90 degree counterclockwise rotate

**Return value:**

Rotated RawImage image object

**Exception handling:**

1) If the rotate is unsuccessful, throw the exception **UnexpetedError**.

➤ **brightness**

**Statement:**

```
RawImage.brightness(factor)
```

**Significance:**

Brightness adjustment on 8-bit gray scale images

**Formal parameter:**

```
[in] factor      Brightness adjustment parameter, range: -150 ~ 150.  
                  Note: 0: no change in brightness  
                       >0: increase brightness  
                       <0: decrease brightness
```

**Return value:**

None

**Exception handling:**

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

- 2) If the pixel format is not 'Mono8', the error message **RawImage.brightness only support mono8 image** is printed, and throw the exception **InvalidParameter**.
- 3) If the image brightness adjustment fails, throw the exception **UnexpectedError**.

➤ **contrast**

**Statement:**

```
RawImage.contrast(factor)
```

**Significance:**

Contrast adjustment on 8-bit gray scale images

**Formal parameter:**

```
[in] factor    Contrast adjustment parameter, range: -50 ~ 150.  
              Note: 0: no change in contrast  
                   >0: increase contrast  
                   <0: decrease contrast
```

**Return value:**

None

**Exception handling:**

- 1) If the input parameter is not an int, throw the exception **ParameterTypeError**.
- 2) If the pixel format is not 'Mono8', the error message **RawImage.brightness only support mono8 image** is printed, and throw the exception **InvalidParameter**.
- 3) If the image contrast adjustment fails, throw the exception **UnexpectedError**.

➤ **mirror**

**Statement:**

```
RawImage.mirror(mirror_mode)
```

**Significance:**

Generate a mirror image that is symmetrical to the original image in the horizontal or vertical direction for 8-bit images

**Formal parameter:**

```
[in] mirror_mode    Image mirror mode, refer to DxImageMirrorMode
```

**Return value:**

RawImage image object after mirroring



**Exception handling:**

- 1) If the input parameter is not an int, throw the exception **ParameterTypeError**.
- 2) If the pixel format is not Mono8, throw the exception **InvalidParameter**.
- 3) If the mirror fails, throw the exception **UnexpectedError**.

**➤ get\_ffc\_coefficients****Statement:**

```
RawImage.get_ffc_coefficients(dark=None, target_value=None)
```

**Significance:**

Calculate FFC coefficients, only 8~12bit RAW images are supported

**Formal parameter:**

```
[in]dark           Dark field image  
[in]target_value   Target gray value
```

**Return value:**

FFC coefficients

**Exception handling:**

- 1) If the dark type is not 'RawImage', throw the exception **ParameterTypeError**.
- 2) If the target value is not int, throw the exception **ParameterTypeError**.
- 3) If the pixel format is not 8/10/12bit, throw the exception **InvalidParameter**.
- 4) If the calling get\_ffc\_coefficients fails, throw the exception **UnexpectedError**.

**➤ flat\_field\_correction****Statement:**

```
RawImage.flat_field_correction(ffc_coefficients)
```

**Significance:**

Execute FFC for 8~12bit RAW images

**Formal parameter:**

```
[in]ffc_coefficients   Flat Field Correction coefficients
```

**Return value:**

None

**Exception handling:**

- 1) If the ffc\_coefficients type is not Buffer, throw the exception **ParameterTypeError**.
- 2) If the pixel format is not 8/10/12bit, throw the exception **InvalidParameter**.
- 3) If the calling flat\_field\_correction fails, throw the exception **UnexpectedError**.

**➤ 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
get_lut(contrast=0, gamma=1, lightness=0)	Calculate 8-bit LUT for image processing
calc_cc_param(color_correction_param, saturation=64)	Calculate color correction parameter array for image processing
calc_user_set_cc_param(color_transform_factor, saturation=64)	Calculate color correction parameter array by user set for image processing

◆ **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**.

➤ **get\_lut (Static function)**

**Statement:**

```
Utility.get_lut(contrast=0, gamma=1, lightness=0) (Static function)
```



### Significance:

Caculate 8-bit LUT for image processing

### Formal parameter:

[in]contrast	Contrast adjust parameter, int, range [-50, 100], the default is 0
[in]gamma	Adjust parameter, float, range [0.1, 10], the default is 1
[in]lightness	Lightness adjust parameter, int, range [-150, 150], the default is 0

### Return value:

LUT buffer object

### Exception handling:

- 1) If the contrast/gamma/lightness parameter is not an int/float/int, a **ParameterTypeError** exception is thrown.
- 2) If getting the LUT unsuccessfully, the interface name, the lut fail to get, and the error code are printed, and the function returns **None**.

### ➤ calc\_cc\_param (Static function)

### Statement:

```
Utility.calc_cc_param(color_correction_param, saturation=64) (Static function)
```

### Significance:

Caculate color correction parameter array for image processing

### Formal parameter:

[in]color_correction_param	Color correction parameter, Int, Can be obtained from the camera or set to 0
[in]saturation	Saturation adjust parameter, Int, range [0, 128],the default is 64

### Return value:

Color correction parameter buffer object

### Exception handling:

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

- 2) If getting the color correction parameter unsuccessfully, the interface name, the color correction parameter fail to get, and the error code are printed, and the function returns **None**.

➤ **calc\_user\_set\_cc\_param (Static function)**

**Statement:**

```
Utility.calc_user_set_cc_param(color_transform_factor, saturation=64)  
(Static function)
```

**Significance:**

Calculate color correction parameter array by user set for image processing

**Formal parameter:**

[in]color_transform_factor	Color correction/color transformation array, list or tuple type
[in]saturation	Saturation adjust parameter, Int, range [0, 128],the default is 64

**Return value:**

Color correction parameter buffer object

**Exception handling:**

- 1) If the input parameter is not a list (tuple) or int, a **ParameterTypeError** exception is thrown.
- 2) If getting the color correction parameter unsuccessfully, the interface name, the color correction parameter 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: <b>NotInitApi: DeviceManager.update_device_list: {-13}{Not init API}</b>	1) Please check and delete the statement of the <b>__del__()</b> function that calls the <b>DeviceManager</b> class object in the program. Because Python's garbage collection mechanism automatically calls the <b>__del__()</b> function to destroy the object, it does not need or allow the user to display the call to the <b>__del__()</b> function, such as: <b>device_manager.__del__()</b> .

## 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
7	V1.0.6	Add reconnect device and brightness adjust, mirror and other image processing library interfaces	2021-06-03