

Python Interface Development User Manual

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.

© 2025 China Daheng Group, Inc. Beijing Image Vision Technology Branch

Web: www.daheng-imaging.com/en

Sales Email: isales@daheng-imaging.com

Sales Tel: +86 10 8282 8878-8081

Support Email: isupport@daheng-imaging.com

Contents

1. Camera Workflow	1
1.1. Overall workflow	1
1.2. Function control flow	2
1.3. Overall code sample.....	3
2. Programming Guide	5
2.1. Build programming environment.....	5
2.1.1. Linux	5
2.1.2. Windows	6
2.2. QuickStart.....	7
2.2.1. Importing library	7
2.2.2. Enumeration device.....	7
2.2.3. Open or close the device.....	8
2.2.4. Acquisition control.....	10
2.2.5. Image processing	11
2.2.6. Camera control	14
2.2.7. Import and export camera configuration parameter.....	20
2.2.8. Error handling	20
3. Appendix	22
3.1. Function class definition	22
3.1.1. Feature_s (Recommended)	22
3.1.2. Feature (No longer maintained)	27
3.2. Data type definition.....	38
3.2.1. GxLogTypeList.....	38
3.2.2. GxDeviceClassList	38
3.2.3. GxAccessStatus	38
3.2.4. GxAccessMode	39
3.2.5. GxPixelFormatEntry	39
3.2.6. GxFrameStatusList.....	42
3.2.7. GxDeviceTemperatureSelectorEntry	42
3.2.8. GxPixelSizeEntry	42
3.2.9. GxPixelColorFilterEntry	43
3.2.10. GxAcquisitionModeEntry	43
3.2.11. GxTriggerSourceEntry	43
3.2.12. GxTriggerActivationEntry.....	44
3.2.13. GxExposureModeEntry	44
3.2.14. GxUserOutputSelectorEntry	44
3.2.15. GxUserOutputModeEntry	45
3.2.16. GxGainSelectorEntry.....	45
3.2.17. GxBlackLevelSelectEntry	45
3.2.18. GxBalanceRatioSelectorEntry.....	45
3.2.19. GxAALightEnvironmentEntry.....	45
3.2.20. GxUserSetEntry.....	46
3.2.21. GxAWBLampHouseEntry	46

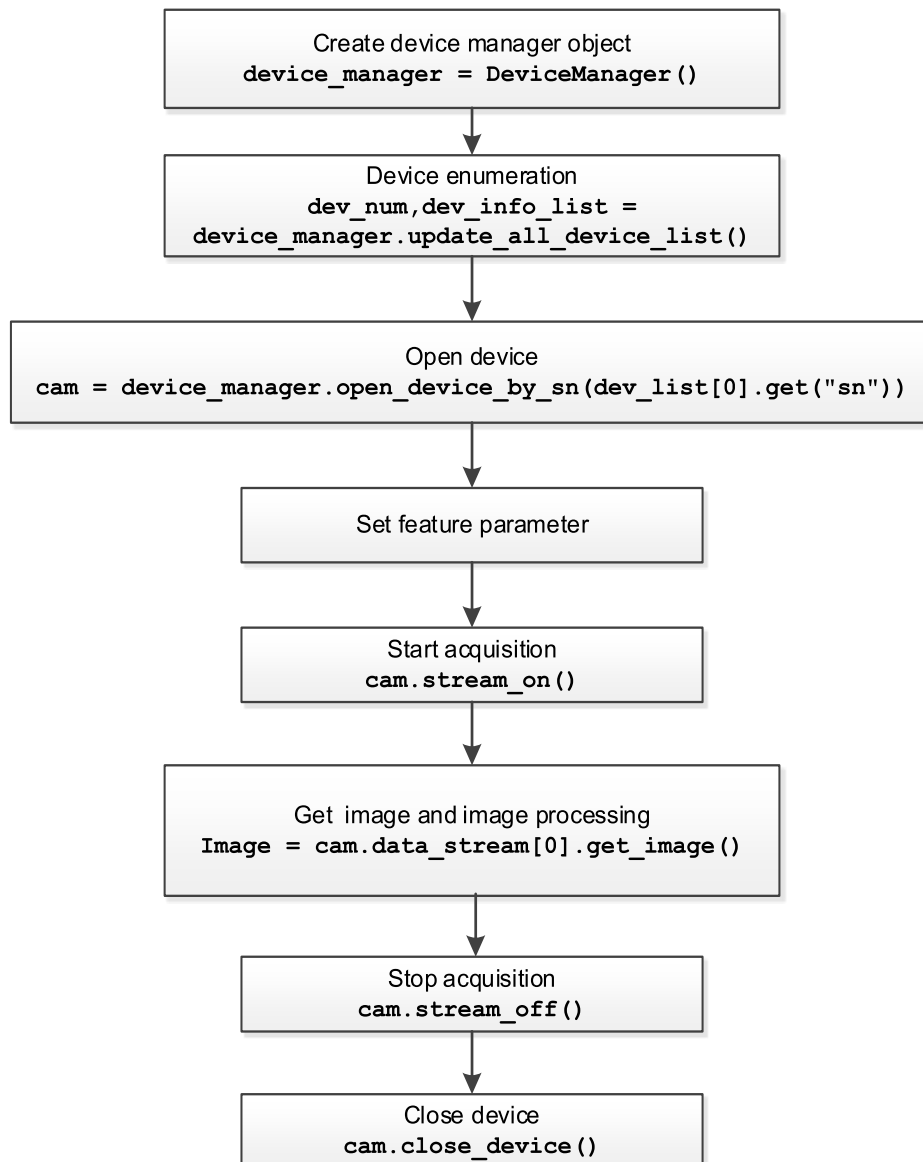
3.2.22. GxUserDataFieldSelectorEntry	46
3.2.23. GxTestPatternEntry	47
3.2.24. GxTriggerSelectorEntry	47
3.2.25. GxLineSelectorEntry	47
3.2.26. GxDeviceSerialPortBaudRateEntry	48
3.2.27. GxSerialPortStopBitsEntry	48
3.2.28. GxLineModeEntry	48
3.2.29. GxLineSourceEntry	49
3.2.30. GxEventSelectorEntry	50
3.2.31. GxLutSelectorEntry	50
3.2.32. GxTransferControlModeEntry	50
3.2.33. GxTransferOperationModeEntry	50
3.2.34. GxTestPatternGeneratorSelectorEntry	51
3.2.35. GxChunkSelectorEntry	51
3.2.36. GxBinningHorizontalModeEntry	51
3.2.37. GxBinningVerticalModeEntry	51
3.2.38. GxSensorShutterModeEntry	51
3.2.39. GxAcquisitionStatusSelectorEntry	52
3.2.40. GxExposureTimeModeEntry	52
3.2.41. GxGammaModeEntry	52
3.2.42. GxLightSourcePresetEntry	52
3.2.43. GxColorTransformationModeEntry	52
3.2.44. GxColorTransformationValueSelectorEntry	53
3.2.45. GxAutoEntry	53
3.2.46. GxSwitchEntry	53
3.2.47. GxSensorBitDepthEntry	53
3.2.48. GxMultisourceSelectorEntry	54
3.2.49. GxDeviceTapGeometryEntry	54
3.2.50. GxEncoderSelectorEntry	54
3.2.51. GxEncoderSourceAEntry	54
3.2.52. GxEncoderSourceBEntry	54
3.2.53. GxEncoderModeEntry	55
3.2.54. GxEncoderDirectionEntry	55
3.2.55. GxRegionSendModeEntry	55
3.2.56. GxShadingCorrectionModeEntry	55
3.2.57. GxFFCCGenerateStatusEntry	55
3.2.58. GxFFCCCoefficientEntry	56
3.2.59. GxDSNUSelectorEntry	56
3.2.60. GxDSNUGenerateStatusEntry	57
3.2.61. GxPRNUSelectorEntry	57
3.2.62. GxPRNUGenerateStatusEntry	58
3.2.63. GxCXPLinkConfigurationEntry	58
3.2.64. GxCXPLinkConfigurationPreferredEntry	58
3.2.65. GxCXPLinkConfigurationStatusEntry	58
3.2.66. GxCXPConectionSelectorEntry	59
3.2.67. GxCXPConectionTestModeEntry	59

3.2.68. GxSequencerFratureSelectorEntry	59
3.2.69. GxSequencerTriggerSourceEntry	59
3.2.70. GxRegionSelectorEntry	59
3.2.71. GxTimerSelectorEntry	60
3.2.72. GxTimerTriggerSourceEntry	60
3.2.73. GxNoiseReductionModeEntry	60
3.2.74. GxHDMREntry	60
3.2.75. GxMGCMODEntry	60
3.2.76. GxAcquisitionBurstModeEntry	61
3.2.77. GxSensorSelectorEntry	61
3.2.78. GxIMUConfigAccRangeEntry	61
3.2.79. GxIMUConfigAccOdrEntry	61
3.2.80. GxIMUConfigAccOdrLowPassFilterFrequencyEntry	61
3.2.81. GxIMUConfigGyroRangeEntry	62
3.2.82. GxIMUConfigGyroOdrEntry	62
3.2.83. GxIMUConfigGyroOdrLowPassFilterFrequencyEntry	62
3.2.84. GxIMUTemperatureOdrEntry	64
3.2.85. GxSerialportSelectorEntry	64
3.2.86. GxSerialportSourceEntry	64
3.2.87. GxSerialportBaundrateEntry	64
3.2.88. GxSerialporeStopBitsEntry	64
3.2.89. GxSerialportParityEntry	65
3.2.90. GxCounterSelectorEntry	65
3.2.91. GxCounterEventSourceEntry	65
3.2.92. GxCounterResetSourceEntry	66
3.2.93. GxCounterResetActivationEntry	66
3.2.94. GxCounterTriggerSourceEntry	66
3.2.95. GxTimerTriggerActivationEntry	66
3.2.96. GxStopAcquisitionModeEntry	67
3.2.97. GxDSSStreamBufferHandlingModeEntry	67
3.2.98. GxResetDeviceModeEntry	67
3.2.99. Dx BayerConvertType	67
3.2.100. DxValidBit	67
3.2.101. DxImageMirrorMode	68
3.2.102. DxRGBChannelOrder	68
3.2.103. GxTLCClassList	68
3.2.104. GxImageInfo	68
3.2.105. GxIPCConfigureModeList	69
3.2.106. GxActionCommandResult	69
3.2.107. GxRegisterStackEntry	69
3.2.108. ColorTransformFactor	69
3.3. Module interface definition	70
3.3.1. DeviceManager	70
3.3.2. Device	80
3.3.3. Interface	87
3.3.4. DataStream	88

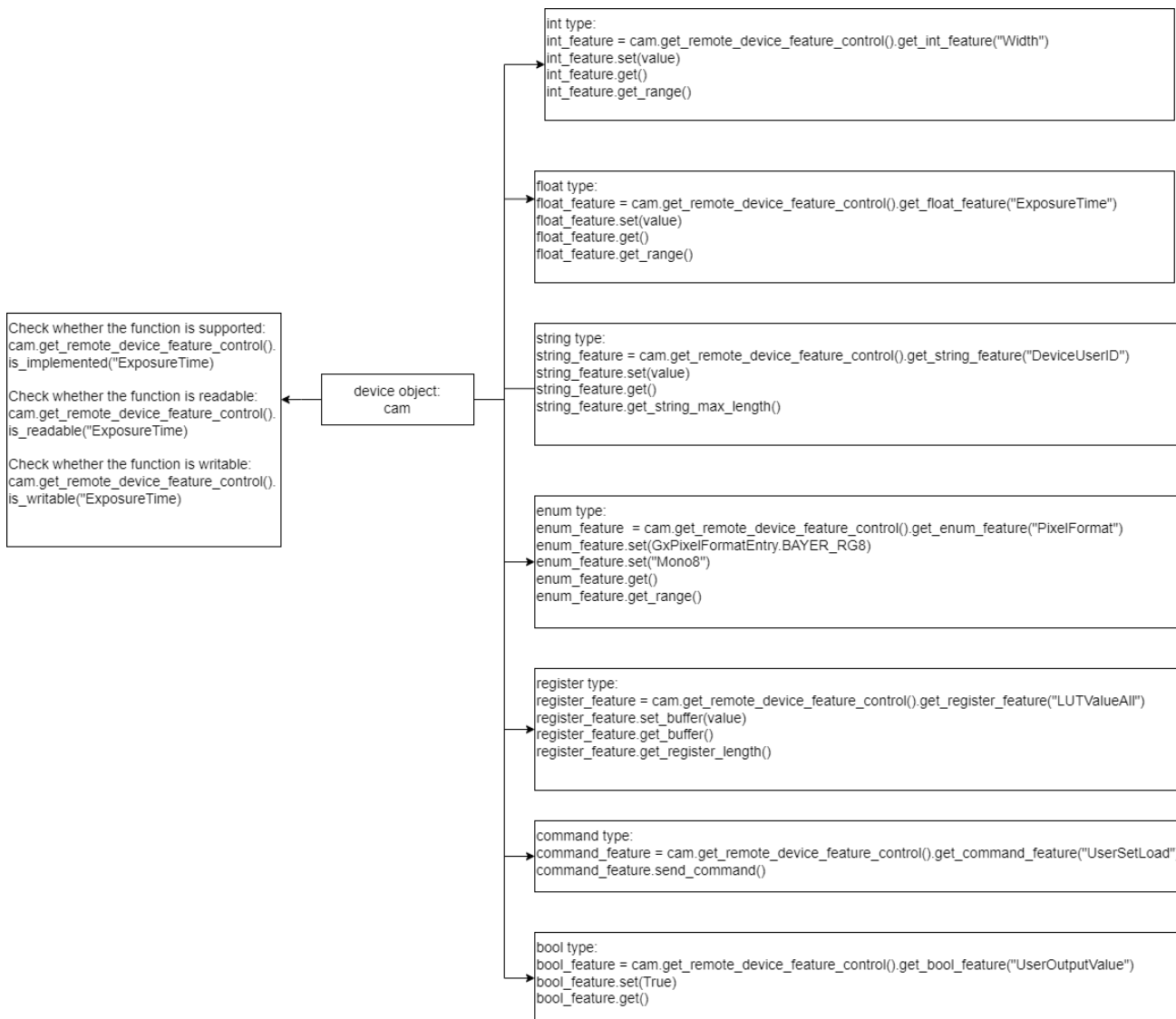
3.3.5. FeatureControl.....	93
3.3.6. RGBImage (No longer mantained).....	99
3.3.7. RawImage.....	102
3.3.8. Buffer	109
3.3.9. ImageProcessConfig	111
3.3.10. Utility	117
3.3.11. ImageProcess	120
3.3.12. ImageFormatConvert.....	121
3.4. Feature Parameter (No longer maintaining)	126
3.4.1. Device feature parameter	126
3.4.2. Stream feature parameter	137
4. FAQ	139
5. Revision History	140

1. Camera Workflow

1.1. Overall workflow



1.2. Function control flow



1.3. Overall code sample

```
# Users can customize the prefix for calling, and gx was used in the
# example
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_all_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
    # Create a format converter
    image_convert = device_manager.create_image_format_convert()

    # Set the target pixel format to be converted
    image_convert.set_dest_format(GxPixelFormatEntry.RGB8)

    # Set the valid bit to be converted. Assuming there are 12
    # significant bits, select the top 8 bits
    image_convert.set_valid_bits(DxValidBit.BIT4_11)

    # Create a converted image cache
    rgb_image_buffer_length =
        image_convert.image_format_convert_get_buffer_
        size_for_conversion(raw_image)
    rgb_image_array = (c_ubyte * buffer_out_size)()
```

```
rgb_image = addressof(output_image_array)

# Convert pixel format to RGB8
image_convert.convert(raw_image, rgb_image,
                      buffer_out_size, False)
if output_image is None:
    continue

# Create a numpy array
numpy_image = numpy.frombuffer(rgb_image_array, dtype=numpy.ubyte,
                               count=rgb_image_buffer_length).\
    reshape(raw_image.frame_data.height,
            raw_image.frame_data.width, 3)

if numpy_image is None:
    continue

# Display and save the obtained RGB image
image = Image.fromarray(numpy_image, 'RGB')
image.show()
image.save("image.jpg")

# Stop collection, turn off device
cam.stream_off()
cam.close_device()
```

2. Programming Guide

2.1. Build programming environment

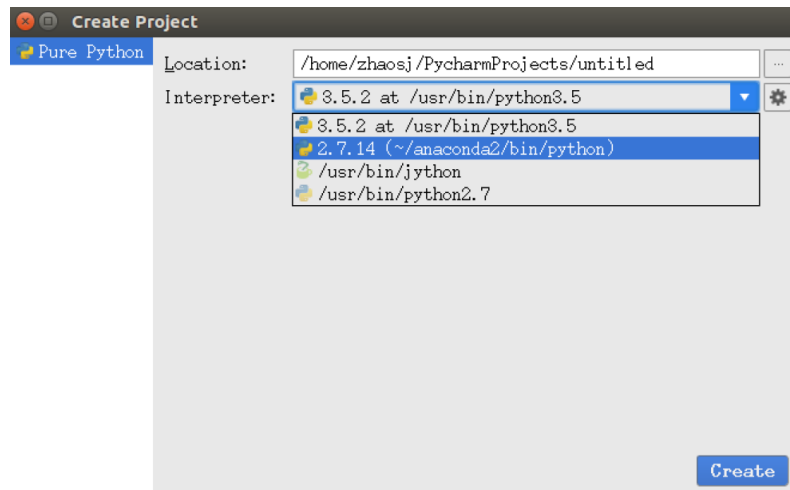
It is recommended that users use Python 2.7 and Python 3.5 first. This interface library has been tested in the above two versions.

2.1.1. Linux

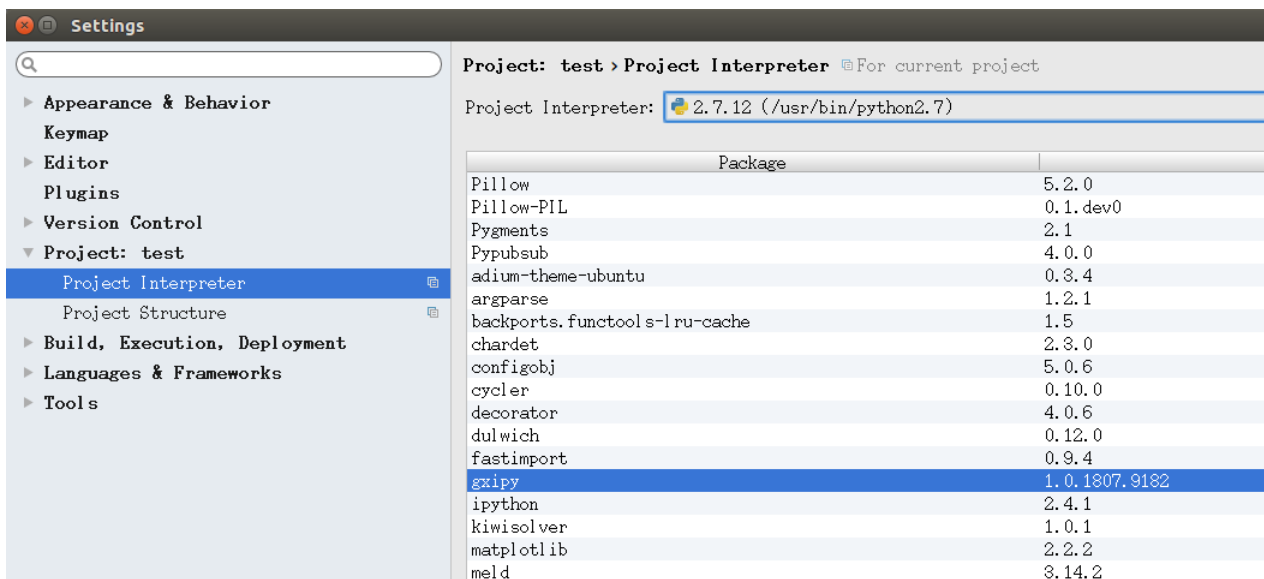
- 1) Install **pycharm-community** (community free version).

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

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



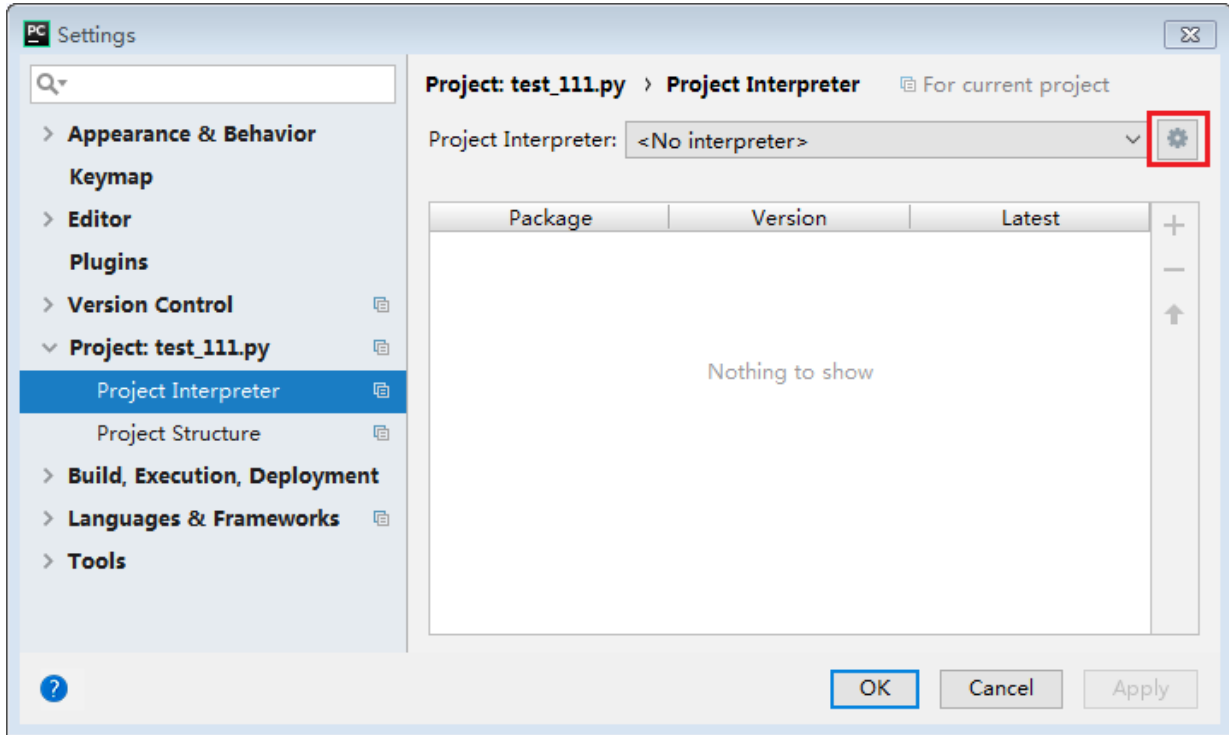
- 3) **File->Settings->Project->Project Interpreter** can check the **Package** installed by the Python interpreter. As shown in the figure, the **gxipy** library is installed normally and can be imported by user's program.



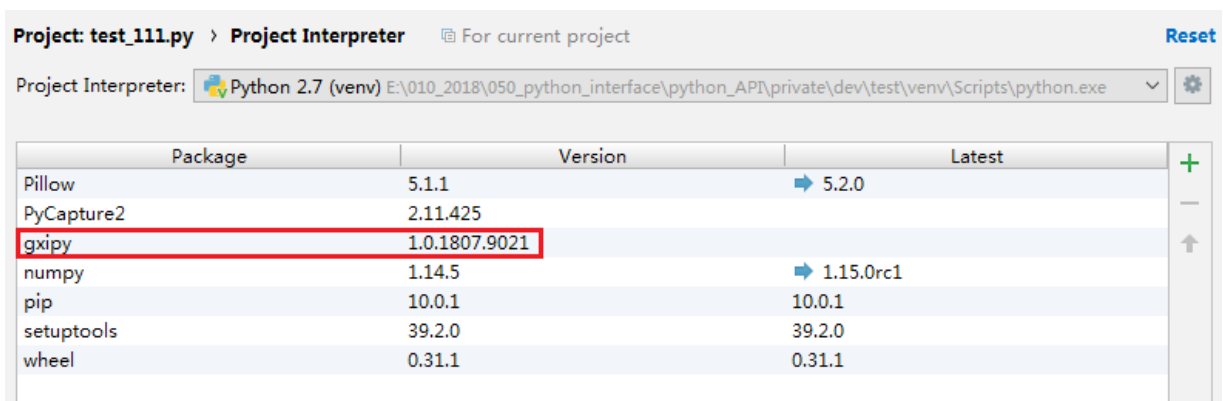
2.1.2. Windows

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

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



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



2.2. QuickStart

2.2.1. Importing library

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

Sample code:

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

2.2.2. Enumeration device

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

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

nic_gateway	nic gateway (GEV camera only)	string
nic_description	nic description (GEV camera only)	string

The code snippets for enumerating devices are as follows:

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

Note:

In addition to the enumeration interfaces above, [DeviceManager](#) provides another two enumeration interface [DeviceManager.update_all_device_list\(\)](#) and [DeviceManager.update_device_list_ex\(\)](#).

- 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.

update_device_list_ex: Use entire network enumeration to enumerate specific types devices. The code snippet for enumerating devices s as follows:

```
# Enumerate devices
device_manager = gx.DeviceManager()
dev_num, dev_info_list =
    device_manager.update_device_list_ex(
        GxTLClassList.TL_TYPE_CXP)
if dev_num == 0:
    sys.exit(1)
```

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:

```
# Users can customize the prefix for calling, and gx was used in the
# example
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()
device_manager = gx.DeviceManager()
tl_type = GxTLClassList.TL_TYPE_U3V | GxTLClassList.TL_TYPE_USB |
          GxTLClassList.TL_TYPE_GEV | GxTLClassList.TL_TYPE_CXP
dev_num, dev_info_list = device_manager.update_device_list_ex(tl_type)
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 dq_buf**

```
# 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].dq_buf()
    if raw_image.get_status() == gx.GxFrameStatusList.INCOMPLETE:
        print("incomplete frame")
    # dq_buf must be used in pairs with q_buf, otherwise it may
    # cause continuous capture to fail
    cam.data_stream[0].q_buf(raw_image)

# 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

2.2.5.1. Image format conversion

The image format conversion object is the **raw_image** got by acquiring the image with [DataStream.get_image\(\)](#), [DataStream.dq_buf](#).

Functional description:

Supports conversion of any pixel format. Please refer to the [ImageFormatConvert.convert](#) interface for details

Sample code:

```
# Create device manager
device_manager = gx.DeviceManager()

# Enumerate devices
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)

# Open the first device
cam = self.device_manager.open_device_by_index(1)

# Start acquisition
cam.stream_on()

# Obtain the raw image, assuming its pixel format is BayerRG12
raw_image = cam.data_stream[0].get_image()

# Stop acquisition
cam.stream_off()

# Create a format converter
image_convert = device_manager.create_image_format_convert()

# Set the target pixel format to be converted
image_convert.set_dest_format(GxPixelFormatEntry.RGB8)

# Set the significant bits to be converted
image_convert.set_valid_bits(DxValidBit.BIT4_11)

# Create a converted image cache
buffer_out_size =
    image_convert.get_buffer_size_for_conversion(raw_image)
output_image_array = (c_ubyte * buffer_out_size)()
output_image = addressof(output_image_array)

# Convert pixel format to RGB8
image_convert.convert(raw_image, output_image,
                     buffer_out_size, False)
if output_image is None:
    continue

# Create a numpy array
numpy_image = numpy.frombuffer(rgb_image_array, dtype=numpy.ubyte,
                              count=rgb_image_buffer_length). \
    reshape(raw_image.frame_data.height,
            raw_image.frame_data.width, 3)

if numpy_image is None:
    continue
# Afterwards, users can display and save images based on the obtained
numpy.array
```

2.2.5.2. 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 [ImageProcess.image_improvement\(\)](#) interface of [ImageProcess](#) to implement the function. A simple example is as follows:

```
# Assuming image quality improvement for images with pixel format
# RGB8

# Create Device Manager
device_manager = gx.DeviceManager()

# Open the first device
cam = device_manager.open_device_by_index(1)

# Create image quality improvement processing objects
image_process = device_manager.create_image_process()

# Creating Image Quality Improvement Parameter Configuration Objects
image_config = cam.create_image_process_config()

# Create Attribute Controller
remote_device_feature = cam.get_remote_device_feature_control()

# Obtain contrast and gamma parameters
if remote_device_feature.is_readable("GammaParam"):
    gamma_param =
        remote_device_feature.get_float_feature("GammaParam").get()
else:
    gamma_param = 0.1

if remote_device_feature.is_readable("ContrastParam"):
    contrast_param =
        remote_device_feature.get_int_feature("ContrastParam").get()
else:
    contrast_param = 0

# Set contrast and gamma parameters
image_config.set_contrast_param(gamma_param)
image_config.set_gamma_param(contrast_param)

# Create a buffer cache stored after image quality improvement
output_image_array = (c_ubyte * raw_image.frame_data.image_size)()
output_image = addressof(output_image_array)

# Acquire and obtain raw_image, call image processing interface to
# improve quality
image_process.image_improvement(raw_image, output_image,
                                image_config)
```

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.

2.2.5.3. 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 control types

There are several types of attribute control as follows:

1) Device.get_local_device_feature_control(): Obtain the local backup attribute controller.

2) Device.get_remote_device_feature_control(): Get the remote device property controller.

- 3) DeviceManagetr.get_interface().get_feature_control(): Get the interface property controller.
- 4) Device.get_stream().get_feature_control(): Get the flow property controller.

Open the device through the GalaxyView, see the property control tree on the right. There are three parts on the device property page. The upper part corresponds to the property set returned by Device.get_remote_device_feature_control() (Figure 2-1), the middle part corresponds to the property set returned by Device.get_local_device_feature_control() (Figure 2-1), and the lower part corresponds to the property set returned by Device.get_stream().get_feature_control() (Figure 2-1).

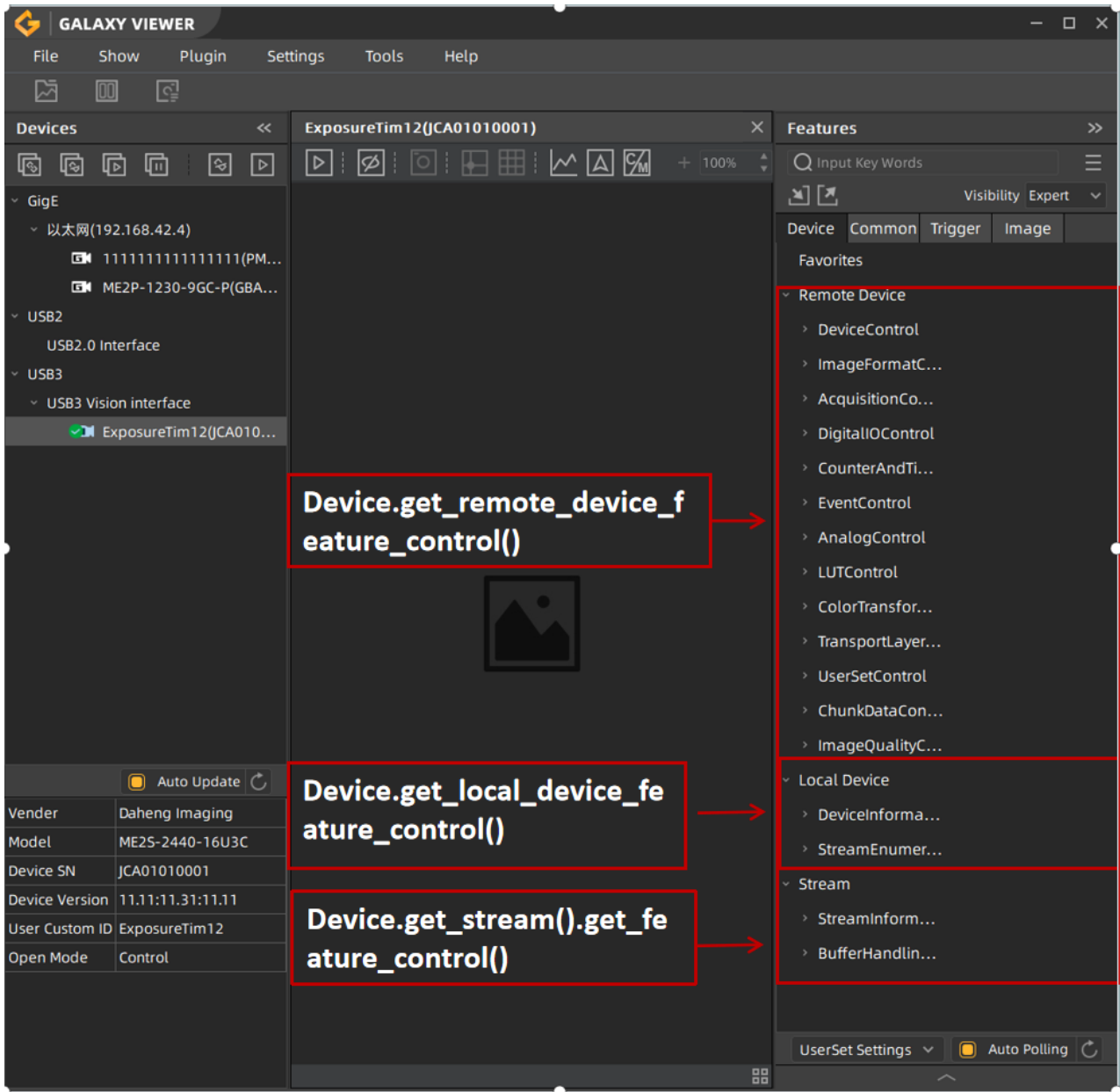


Figure 2-1 Remote, local and stream properties

Click the Interface node, and see the attribute control tree on the right. The corresponding function on the property page is the set of attributes returned by DeviceManagetr.get_interface().get_feature_control() (Figure 2-2).

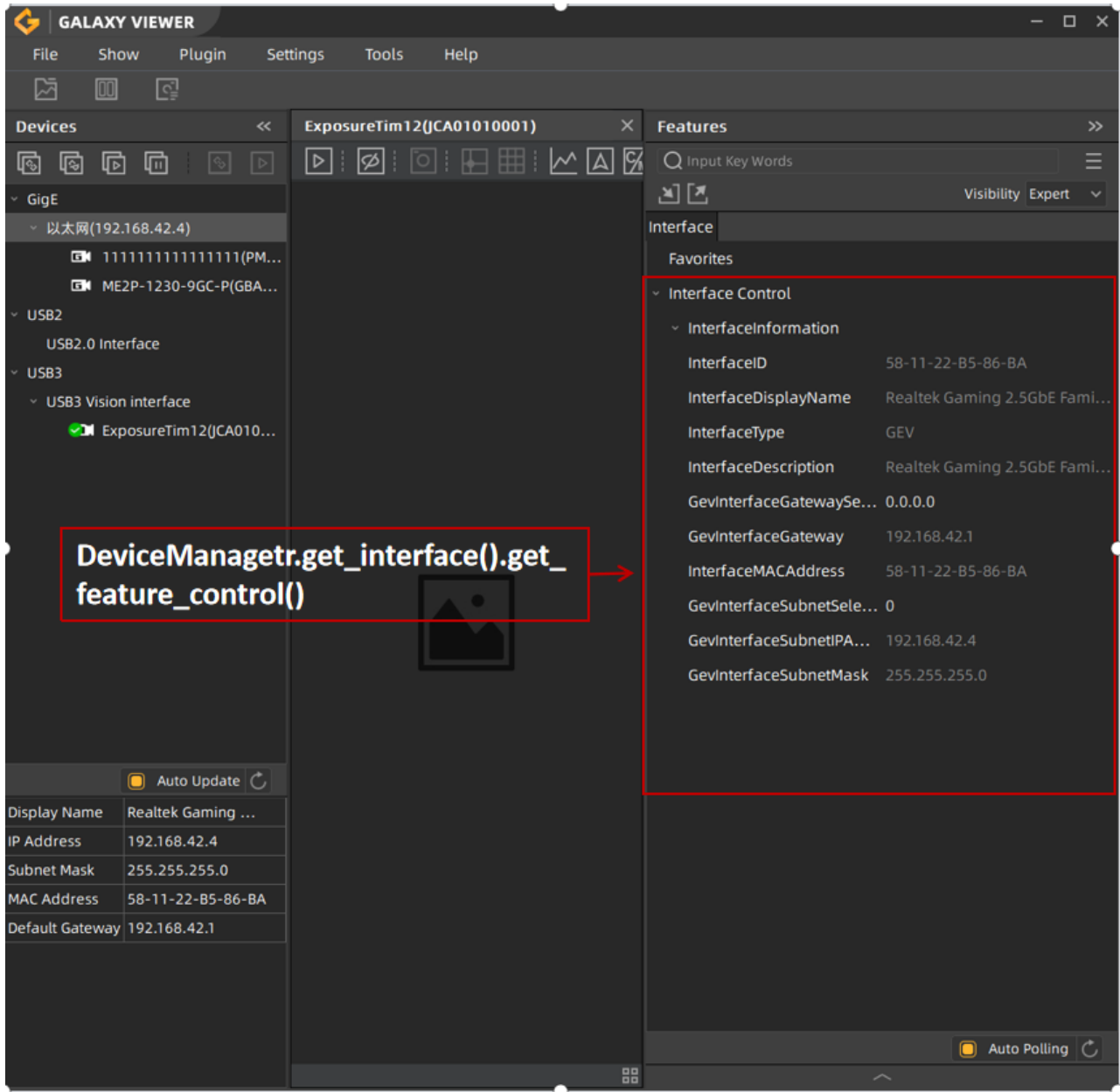


Figure 2-2 Interface Properties

● 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:

FeatureControl.is_implement(feature_name) Does this feature support

FeatureControl.is_readable(feature_name) Does this feature readable

FeatureControl.is_writable(feature_name) Does this feature writable

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

Sample code:

```
# Check if it is supported
remote_device_feature = cam.get_remote_device_feature_control()
is_implemented =
remote_device_feature.is_implemented("PixelFormat")
if is_implemented == True:
# Check if it is writable
is_writable = remote_device_feature.is_writable("PixelFormat")
if is_writable == True:
# Set pixel format
remote_device_feature.get_enum_feature("PixelFormat").set("Mono8")
# Check if it is readable
is_readable = remote_device_feature.is_readable("PixelFormat")
if is_readable == True:
# Print pixel format
value, str_value =
remote_device_feature.get_enum_feature("PixelFormat").get()
print(str_value)
```

Int Type Data Node:

Related interfaces:

```
IntFeature_s.set()           // Set the current value
IntFeature_s.get()           // Get the current value
IntFeature_s.get_range()     // Get the minimum, maximum and step size
```

Sample code:

```
# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()

# Get image width data node object
width_feature = remote_device_feature.get_int_feature("Width")

# Obtain image width with adjustable range
width_range = width_feature.get_range()

# Set the current image width to any value within the range
Width_feature.set(800)

# Get the current image width
int_Width_value = width_feature.get()
```

Float Type Data Node:

Related interfaces:

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

Sample code:

```
# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()

# Get exposure time data node object
exposure_time_feature =
    remote_device_feature.get_float_feature("ExposureTime")

# Obtain exposure values can set the range and maximum value
float_range = rexposure_time_feature.get_range()
float_max = float_range["max"]
# Set any value within the current exposure value range
exposure_time_feature.set(10.0)
# Get the current exposure value
float_exposure_value = exposure_time_feature.get()
```

Enum Type Data Node:

Related interfaces:

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

Sample code:

```
# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()

# Get Pixel Format data node Object
pixel_format_feature =
    remote_device_feature.get_enum_feature("PixelFormat")

# Obtaining enumeration values can set a range
enum_range = pixel_format_feature.get_range()

# Set the current enumeration value
pixel_format_feature.set("Mono8")

# Print the current enumeration value
enum_PixelFormat_value, enum_PixelFormat_key =
    pixel_format_feature.get()
```

Bool Type Data Node:

Related interfaces:

```
BoolFeature_s.set(bool_value)    // Set the current value
BoolFeature_s.get()              // Get the current value
```

Sample code:

```
# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()
```



```
# Get pin level reversal data node object
line_inverter_feature =
    remote_device_feature.get_bool_feature("LineInverter")

# Set current Boolean value
line_inverter_feature.set(True)

# Read the Boolean value
bool_LineInverter_value = line_inverter_feature.get()
```

String Type Data Node:

Related interfaces:

```
StringFeature_s.set(string_value)    // Set the current value

StringFeature_s.get()                // Get the current value

StringFeature_s.get_string_max_length() // Get the maximum length value of string
                                         // feature
```

Sample code:

```
# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()

# Get user-defined name data node object
device_user_id_feature =
    remote_device_feature.get_string_feature("DeviceUserID")

# Read the longest string length
string_max_length = device_user_id_feature.get_string_max_length()

# Read the current string value
current_string = device_user_id_feature.get()

# Set string value
device_user_id_feature.set("MyUserID")
```

Buffer Type Data Node:

Related interfaces:

```
RegisterFeature_s.set_buffer(buf)    // Set the current value

RegisterFeature_s.get_buffer()        // Get the current value

RegisterFeature_s.get_register_length() // Get the length of the buffer type feature
                                         // parameter
```

Sample code:

```
# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()
# Get Register data node Object
data_field_value_all_feature =
    remote_device_feature.get_register_feature(
        "DataFieldValueAll")
```

```
# Read node data length
buffer_length = data_field_value_all_feature.get_register_length()

# Set node data
string_set = "abcd"
set_buffer = gx.Buffer.from_string(string_set_1.encode('utf-8'))
data_field_value_all_feature.set_buffer(string_set)

# Read node data
buffer_data = data_field_value_all_feature.get_buffer()
print("buffer_data: %s" % (buffer_data.get_data().decode()))
```

Command Type Data Node:

Related interface:

CommandFeature_s.send_command // Send command

Sample code:

```
# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()

# Get soft trigger command data node object
trigger_soft_ware_feature =
    remote_device_feature.get_register_feature("TriggerSoftware")
# Send command: soft trigger
trigger_soft_ware_feature.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 device remote attribute controller configuration parameter
# file

# Get remote attribute controller
remote_device_feature = cam.get_remote_device_feature_control()
remote_device_feature.feature_load("import_config_file.txt")

# Export device remote attribute controller configuration parameter
# file
remote_device_feature.feature_save("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
NotImplemented	Unrealized
NotInitApi	Not initialized
Timeout	Timeout
ParameterTypeError	Parameter type error

3. Appendix

3.1. Function class definition

3.1.1. Feature_s (Recommended)

Feature_s is the parent class of the attribute class.: [IntFeature_s](#) / [FloatFeature_s](#) / [EnumFeature_s](#) / [BoolFeature_s](#) / [StringFeature_s](#) / [BufferFeature_s](#) / [CommandFeature_s](#).

Known conditions: The camera has been opened and the [Device](#) object has been obtained through [DeviceManager](#), with the variable name cam.

Step1: Obtain the corresponding feature_control object through Device object, for example:

```
obj_feature_control = cam.get\_remote\_device\_feature\_control\(\).
```

Step2: Obtain the data node object with the specified attribute name through the feature_control object:

For example, accessing the Width of an integer data node:

```
obj_width = remote_device_feature.get\_int\_feature ("Width")
```

```
obj_width.set (2048)
```

Or accessing a float data node Gain

```
obj_gain = remote_device_feature.get\_float\_feature("Gain")
```

```
obj_gain.set(1.0)
```

In summary, users access properties based on their data node name string and corresponding data types (integer, float, enumeration, etc.)

3.1.1.1. IntFeature_s

Responsible for reading, writing, and controlling the integer data node function of the camera, inherited from the [Feature_s](#) class.

Interface List:

get_range()	Get integer data node parameter range
get()	Get current value
set(value)	Set current value

3.1.1.1.1. get_range

Statement: `IntFeature_s.get_range()`

Significance: Get integer data node parameter range.

Return value: Record the integer data node parameter range. The key includes: min, max, step size, and current value.

Exception handling: If the range of the integer data node parameter is not successfully obtained, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.1.2. get

Statement: IntFeature_s.get()

Significance: Get the current value.

Return value: Get the current value corresponding to the integer data node.

Exception handling: If obtaining the value of the integer data node is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.1.3. set

Statement: IntFeature_s.set(int_value)

Significance: Set the current value.

Return value: The integer value set.

Exception handling: If setting the integer data node unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.2. FloatFeature_s

Responsible for viewing and controlling the floating data node, inherited from the [Feature_s](#) class.

Interface List:

get_range()	Get the parameter ranges of float data node
get()	Get the current value
set()	Set the current value

3.1.1.2.1. get_range

Statement: FloatFeature_s.get_range()

Significance: Get float data node parameter ranges.

Return value: Record the range of float data node parameter. The key includes: min, max, inc step size, unit, whether the unit of inc_is_valid is valid, cur_value means the current value.

Exception handling: If obtaining the parameter range of the float data node is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.2.2. get

Statement: FloatFeature_s.get()

Significance: Get the current value.

Return value: The value of the float property parameter obtained.

Exception handling: If obtaining float data node's current value is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.2.3. set

Statement: FloatFeature_s.set(float_value)

Significance: Set current value.

Parameter: [in] float_value Set float value

Exception handling: If setting the float data node's current value is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.3. EnumFeature_s

Responsible for viewing and controlling the enumeration data node, inherited from the [Feature_s](#) class.

Interface List:

get_range()	Get the enumeration type data node parameter range
get()	Get the current enumeration item value
set()	Set the current enumeration item value

3.1.1.3.1. get_range

Statement: EnumFeature_s.get_range()

Significance: Get the enumeration type data node parameter range.

Return value: A dictionary that records the range of enumeration type data node parameters.

Exception handling: If obtaining the parameter range of the enumeration type data node is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.3.2. get

Statement: EnumFeature_s.get()

Significance: Get the current enumeration item value.

Return value:

- 1) The current value of the enumeration type data node.
- 2) The string of the enumeration type data node parameter.

Exception handling: If obtaining the value of an enumeration data node's current value is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.3.3. set

Statement: EnumFeature_s.set(enum_value)

Significance: Set the current enumeration item value.

Parameter: [in] enum_value Set enumeration values
Note: It can be an enumeration value or a string value corresponding to an enumeration type.

Exception handling: If setting the enumeration data node's current value is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.4. BoolFeature_s

Responsible for viewing and controlling the Boolean data node of the camera, inherited from [Feature_s](#) class.

Interface List:

get()	Get the current value
set()	Set the current value

3.1.1.4.1. get

Statement: BoolFeature_s.get()

Significance: Get the current value.

Return value: Get the bool value.

Exception handling: If obtaining the Boolean data node's current value parameter value is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.4.2. set

Statement: BoolFeature_s.set(bool_value)

Significance: Set the current value.

Parameter: [in] bool_value Set the bool type's current value.

Exception handling: If setting the Boolean type data node's current value is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.5. StringFeature_s

Responsible for viewing and controlling the string type data node of the camera, inherited from [Feature_s](#) class.

Interface List:

get_string_max_length()	Obtain the maximum length that can be set, excluding the terminator
---	---

[get\(\)](#) Get the current value

[set\(input_string\)](#) Set the current value

3.1.1.5.1. get_string_max_length

Statement: StringFeature_s.get_string_max_length()

Significance: Obtain the maximum length that can be set, excluding the terminator.

Return value: The maximum length that can be set for string type data node.

Exception handling: If obtaining the maximum length that can be set for string type data node is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.5.2. get

Statement: StringFeature_s.get()

Significance: Get the current value.

Return value: Get the current value of a string type data node.

Exception handling: If obtaining the current value of the string type data node is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.5.3. set

Statement: StringFeature_s.set(input_string)

Significance: Set the current value.

Parameter: [in] input_string Set string type data node's current value

Exception handling: If setting the string type data node's current value is unsuccessful, an exception will be thrown. Please refer to [Error handling](#) for details.

3.1.1.6. RegisterFeature_s

Responsible for viewing and controlling the camera's buffer type data node, inherited from [Feature_s](#) class.

Interface List:

[get_register_length\(\)](#) Get the length of the buffer. Users read this length to read and write the buffer

[get_buffer\(\)](#) Get the buffer value. The length of the input parameter ptrBuffer must be equal to the length obtained by the get_register_length() interface

[set_buffer\(\)](#) Set the buffer value. The length of the input parameter ptrBuffer must be equal to the length obtained by the get_register_length() interface

3.1.1.6.1. get_register_length

Statement: RegisterFeature_s.get_register_length()

Significance: Get the length of the buffer. The user reads this length to read and write the buffer.

Known: The camera has been opened through DeviceManager and the Device object has been obtained with the variable name cam.

If setting an integer data node's current Width: cam.Width.set(2048),

If setting a float data node's current Gain: cam.Gain.set(1.0).

The disadvantage of this approach is that when CAM has new data nodes, it needs to upgrade the Python library to support new features, which is not user-friendly for users of older versions of the Python library. Therefore, the attribute control of this method no longer maintains the upgraded new features, and existing functional objects can still be used.

In the future, it is recommended that users use the feature_control object to obtain data node object access through the attribute string name. This method can be compatible with new camera features without upgrading the Python library.

3.1.2.1. IntFeature

It is responsible for checking and controlling the camera's int type data node, inherited from the [Feature](#) class.Interface list:

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

3.1.2.1.1. is_implemented

See [Feature::is_implemented\(\)](#) for details.

3.1.2.1.2. is_readable

See [Feature::is_readable\(\)](#) for details.

3.1.2.1.3. is_writable

See [Feature::is_writable\(\)](#) for details.

3.1.2.1.4. get_range

Statement: IntFeature.get_range()

Significance: Get the int data node parameter range.

Return value: Record int data node parameter ranges. The key contains: minimum, maximum and step size.

Exception handling:

- 1) If the int data node function 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 data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.1.5. get

Statement: IntFeature.get()**Significance:** Get the int data node value.**Return value:** The int data node value got.**Exception handling:**

- 1) If the int data node is not implemented or is unreadable, then prints the unreadable information of the int data node , and the function returns **None**.
- 2) If getting the int data node value unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.1.6. set

Statement: IntFeature.set(self, int_value)**Significance:** Set the value of int data node.**Formal parameter:** The set int data node value.**Exception handling:**

- 1) If the input parameter is not an int value, a **ParameterTypeError** exception is thrown.
- 2) If the int data node is not implemented or is not writable, then prints the information that the int data node is not writable, and the function returns **None**.
- 3) If the input parameter is not within the range of the int data node, then prints the information that exceeds the range of the int data node and prints the range, and the function returns **None**.
- 4) If setting the int data node value unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.2. 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 data node is implemented
is_readable()	Test if the float data node is readable
is_writable()	Test if the float data node is writable

get_range()	Get the float data node range dictionary
get()	Get the float data node value
set()	Set the float data node value

3.1.2.2.1. is_implemented

See [Feature::is_implemented\(\)](#) for details.

3.1.2.2.2. is_readable

See [Feature::is_readable\(\)](#) for details.

3.1.2.2.3. is_writable

See [Feature::is_writable\(\)](#) for details.

3.1.2.2.4. get_range

Statement: FloatFeature.get_range()

Significance: Get the float data node range.

Return value: Record the float data node ranges. The keys contain: minimum, maximum, inc step size, unit, whether the inc_is_valid unit is valid.

Exception handling:

- 1) If the float data node is not implemented, then prints the information that does not support the float data node to get range, and the function returns **None**.
- 2) If getting the parameter range of the float data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.2.5. get

Statement: FloatFeature.get()

Significance: Get the value of float data node.

Return value: The float data node value got.

Exception handling:

- 1) If the float data node is not implemented or is unreadable, then prints the unreadable information of the float data node, and the function returns **None**.
- 2) If getting the float data node value unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.2.6. set

Statement: FloatFeature.set(float_value)

Significance: Set the value of float data node.

Formal parameter: [in] float_value The set value of float data node

Exception handling:

- 1) If the input parameter is not a float value, throw a **ParameterTypeError** exception.
- 2) If the float data node is not implemented or is not writable, then prints the information that the float data node is not writable, and the function returns **None**.
- 3) If the input parameter is not within the range of the float data node, then prints the information that exceeds the range of the float data node and prints the range, and the function returns **None**.
- 4) If setting the float data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.3. EnumFeature

It is responsible for checking and controlling the camera's enum data node, inherited from the [Feature](#) class. Interface list:

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

3.1.2.3.1. is_implemented

See [Feature::is_implemented\(\)](#) for details.

3.1.2.3.2. is_readable

See [Feature::is_readable\(\)](#) for details.

3.1.2.3.3. is_writable

See [Feature::is_writable\(\)](#) for details.

3.1.2.3.4. get_range

Statement: EnumFeature.get_range()

Significance: Get the enmu data node range.

Return value: Record enum data node range.

Exception handling:

- 1) If the enum data node is not implemented, then prints the information that does not support the enum data node to get range, and the function returns **None**.
- 2) If getting the range of the enum data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.3.5. get

Statement: EnumFeature.get()

Significance: Get the value and string of the enum data node.

Return value:

- 1) The value of the enum data node.
- 2) The string of enum data node.

Exception handling:

- 1) If the enum data node is not implemented or is unreadable, then prints the unreadable information of the enmu data node, and the function returns **None**.
- 2) If getting enmu data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.3.6. set

Statement: EnumFeature.set(enum_value)

Significance: Set the value of enmu data node.

Formal parameter: [in] enum_value The set value of enmu data node

Exception handling:

- 1) If the input parameter is not an int value, throw a **ParameterTypeError** exception.
- 2) If the enmu data node is not implemented or is not writable, then prints the information that the enmu data node is not writable, and the function returns **None**.
- 3) If the input parameter is not within the range of the enmu data node, then prints the information that exceeds the range of the enmu data node and prints the range, and the function returns **None**.
- 4) If setting the enmu data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.4. BoolFeature

It is responsible for checking and controlling the camera's bool data node, inherited from the [Feature](#) class.
Interface list:

is_implemented()	Test if the bool data node is implemented
is_readable()	Test if the bool data node is readable
is_writable()	Test if the bool data node is writable
get()	Get the value of bool data node
set()	Set the bool data node value

3.1.2.4.1. is_implemented

See [Feature::is_implemented\(\)](#) for details.

3.1.2.4.2. is_readable

See [Feature::is_readable\(\)](#) for details.

3.1.2.4.3. is_writable

See [Feature::is_writable\(\)](#) for details.

3.1.2.4.4. get

Statement: BoolFeature.get()

Significance: Get the data node value of bool.

Return value: The bool data node value got.

Exception handling:

- 1) If the bool data node is not implemented or is unreadable, then prints the unreadable information of the bool data node, and the function returns **None**.
- 2) If getting bool data node value unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.4.5. set

Statement: BoolFeature.set(bool_value)

Significance: Set the value of bool data node.

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 data node is not implemented or is not writable, then prints the information that the bool data node is not writable, and the function returns **None**.
- 3) If setting the bool data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.5. StringFeature

It is responsible for checking and controlling the camera's string data node, inherited from the [Feature](#) class. Interface list:

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

3.1.2.5.1. is_implemented

See [Feature::is_implemented\(\)](#) for details.

3.1.2.5.2. is_readable

See [Feature::is_readable\(\)](#) for details.

3.1.2.5.3. is_writable

See [Feature::is_writable\(\)](#) for details.

3.1.2.5.4. get_string_max_length

Statement: StringFeature.get_string_max_length()

Significance: Get the maximum length of string data node can be set.

Return value: The maximum length that string data node can be set.

Exception handling:

- 1) If the string data node 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 data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.5.5. get

Statement: StringFeature.get()

Significance: Get the value of string data node.

Return value: The string data node value got.

Exception handling:

- 1) If the string data node is not implemented or is unreadable, then prints the unreadable information of the string data node, and the function returns **None**.
- 2) If getting the string data node value unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.5.6. set

Statement: StringFeature.set(input_string)

Significance: Set the value of string data node.

Formal parameter: [in] input_string The set value of string data node

Exception handling:

- 1) If the input parameter is not a string value, throw a **ParameterTypeError** exception.
- 2) If the string data node is not implemented or is not writable, then prints the information that the string data node 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 data node length and prints the maximum value, and the function returns **None**.
- 4) If setting the string data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.6. BufferFeature

It is responsible for checking and controlling the camera's register data node, inherited from the [Feature](#) class.

Interface list:

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

3.1.2.6.1. is_implemented

See [Feature::is_implemented\(\)](#) for details.

3.1.2.6.2. is_readable

See [Feature::is_readable\(\)](#) for details.

3.1.2.6.3. is_writable

See [Feature::is_writable\(\)](#) for details.

3.1.2.6.4. get_buffer_length

Statement: BufferFeature.get_buffer_length()

Significance: Get the length of the buffer data node.

Return value: The length of the buffer data node.

Exception handling:

- 1) If the buffer data node is not implemented, then prints the information that does not support the buffer data node to get range, and the function returns **None**.
- 2) If getting the range of the buffer data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.6.5. get_buffer

Statement: BufferFeature.get_buffer()

Significance: Get the data of buffer data node.

Return value: Register type data node object.

Exception handling:

- 1) If the register type data node is not implemented or is unreadable, then prints the unreadable information of the register type data node, and the function returns **None**.
- 2) If getting the register type data node value unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.6.6. set_buffer

Statement: BufferFeature.set_buffer(buf)

Significance: Set the data of register type data node.

Formal parameter: [in] buffer Set the value of register type data node

Exception handling:

- 1) If the input parameter is not a register type, throw a **ParameterTypeError** exception.

- 2) If the register type data node is not implemented or is not writable, then prints the information that the register type data node is not writable, and the function returns **None**.
- 3) If the input register type data node's data length is greater than the maximum length, then prints the information that exceeding the maximum length of the register type data node and prints the maximum value, and the function returns **None**.
- 4) If setting the register type data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.1.2.7. CommandFeature

It is responsible for the command type data node of the camera, inherited from the [Feature](#) class.

Interface list:

is_implemented()	Test if the command type data node is implemented
is_readable()	Test if the command type data node is readable
is_writable()	Test if the command type data node is writable
send_command()	Send command

3.1.2.7.1. is_implemented

See [Feature::is_implemented\(\)](#) for details.

3.1.2.7.2. is_readable

See [Feature::is_readable\(\)](#) for details.

3.1.2.7.3. is_writable

See [Feature::is_writable\(\)](#) for details.

3.1.2.7.4. send_command

Statement: CommandFeature.send_command()

Significance: Send command.

Exception handling:

- 1) If the command type data node is not implemented, then prints the information that does not support the command type data node , and the function returns **None**.
- 2) If sending the command type data node unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.2. Data type definition

3.2.1. GxLogTypeList

Definition	Value	Explanation
GX_LOG_TYPE_OFF	0x00000000	All types sent/not sent
GX_LOG_TYPE_FATAL	0x00000001	FATAL type
GX_LOG_TYPE_ERROR	0x00000010	ERROR type
GX_LOG_TYPE_WARN	0x00000100	WARN type
GX_LOG_TYPE_INFO	0x00001000	INFO type
GX_LOG_TYPE_DEBUG	0x00010000	DEBUG type
GX_LOG_TYPE_TRACE	0x00100000	TRACE type

3.2.2. 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)
SMART	4	Network intelligent camera
CXP	5	CXP camera

3.2.3. 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

3.2.4. 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

3.2.5. GxPixelFormatEntry

Definition	Value	Explanation
UNDEFINED	0x00000000	-
MONO8	0x01080001	Monochrome 8-bit
MONO8_SIGNED	0x01080002	Monochrome 8-bit signed
MONO10	0x01100003	Monochrome 10-bit unpacked
MONO10_P	0x010A0046	Monochrome 10-bit packed
MONO12	0x01100005	Monochrome 12-bit unpacked
MONO12_P	0x010C0047	Monochrome 12-bit packed
MONO14	0x01100025	Monochrome 14-bit unpacked
MONO14_P	0x010E0104	Monochrome 14-bit packed
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_GR10_P	0x010A0056	Bayer Green-Red 10-bit packed
BAYER_RG10	0x0110000D	Bayer Red-Green 10-bit
BAYER_RG10_P	0x010A0058	Bayer Red-Green 10-bit packed
BAYER_GB10	0x0110000E	Bayer Green-Blue 10-bit
BAYER_GB10_P	0x010A0054	Bayer Green-Blue 10-bit packed
BAYER_BG10	0x0110000F	Bayer Blue-Green 10-bit

BAYER_BG10_P	0x010A0052	Bayer Blue-Green 10-bit packed
BAYER_GR12	0x01100010	Bayer Green-Red 12-bit
BAYER_GR12_P	0x010C0057	Bayer Green-Red 12-bit packed
BAYER_RG12	0x01100011	Bayer Red-Green 12-bit
BAYER_RG12_P	0x010C0059	Bayer Red-Green 12-bit packed
BAYER_GB12	0x01100012	Bayer Green-Blue 12-bit
BAYER_GB12_P	0x010C0055	Bayer Green-Blue 12-bit packed
BAYER_BG12	0x01100013	Bayer Blue-Green 12-bit
BAYER_BG12_P	0x010C0053	Bayer Blue-Green 12-bit packed
BAYER_GR14	0x01100109	Bayer Green-Red 14-bit
BAYER_GR14_P	0x010E0105	Bayer Green-Red 14-bit packed
BAYER_RG14	0x0110010A	Bayer Red-Green 14-bit
BAYER_RG14_P	0x010E0106	Bayer Red-Green 14-bit packed
BAYER_GB14	0x0110010B	Bayer Green-Blue 14-bit
BAYER_GB14_P	0x010E0107	Bayer Green-Blue 14-bit packed
BAYER_BG14	0x0110010C	Bayer Blue-Green 14-bit
BAYER_BG14_P	0x010E0108	Bayer Blue-Green 14-bit packed
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 unpacked
RGB12_PLANAR	0x02300023	Red-Green-Blue 12-bit unpacked
RGB16_PLANAR	0x02300024	Red-Green-Blue 16-bit planar
RGB8	0x2180014	Red-Green-Blue 8-bit
RGB10	0x2300018	Red-Green-Blue 10-bit
RGB12	0x230001A	Red-Green-Blue 12-bit
RGB14	0x230005E	Red-Green-Blue 14-bit
RGB16	0x2300033	Red-Green-Blue 16-bit

BGR8	0x2180015	Blue-Green-Red 8-bit
BGR10	0x2300019	Blue-Green-Red 10-bit
BGR12	0x230001B	Blue-Green-Red 12-bit
BGR14	0x230004A	Blue-Green-Red 14-bit
BGR16	0x230004B	Blue-Green-Red 16-bit
RGBA8	0x2200016	Red-Green-Blue-Alpha 8-bit
BGRA8	0x2200017	Blue-Green-Red-Alpha 8-bit
ARGB8	0x2200018	Alpha-Red-Green-Blue 8-bit
ABGR8	0x2200019	Alpha-Blue-Green-Red 8-bit
R8	0x010800C9	Red 8-bit
G8	0x010800CD	Green 8-bit
B8	0x010800D1	Blue 8-bit
COORD3D_ABC32F	0x026000C0	3D coordinate A-B-C 32-bit floating point
COORD3D_ABC32F_PLANAR	0x026000C1	3D coordinate A-B-C 32-bit floating point planar
COORD3D_C16	0x011000B8	3D coordinate C 16-bit
COORD3D_C16_I16	0x0110FF02	custom pixel format
COORD3D_C16_S16	0x0110FF03	custom pixel format
COORD3D_C16_I16_S16	0x0110FF04	custom pixel format
YUV444_8	0x2180020	YUV444 8-bit
YUV422_8	0x2100032	YUV422 8-bit
YUV411_8	0x20C001E	YUV411 8-bit
YUV420_8_PLANAR	0x20C0040	YUV420 8-bit planar
YCBCR444_8	0x218005B	YCBCR 444 8-bit
YCBCR422_8	0x210003B	YCBCR 422 8-bit
YCBCR411_8	0x20C005A	YCBCR 411 8-bit
YCBCR601_444_8	0x218003D	YCBCR601 444 8-bit
YCBCR601_422_8	0x210003E	YCBCR601 422 8-bit
YCBCR601_411_8	0x20C003F	YCBCR601 411 8-bit
YCBCR709_444_8	0x2180040	YCBCR709 444 8-bit
YCBCR709_422_8	0x2100041	YCBCR709 422 8-bit

YCBCR709_411_8	0x20C0042	YCBCR709 411 8-bit
MONO10_PACKED	0x010C0004	GigE Vision specific format, Mono 10-bit packed
MONO12_PACKED	0x010C0006	GigE Vision specific format, Mono 12-bit packed
BAYER_BG10_PACKED	0x010C0029	GigE Vision specific format, Bayer Blue-Green 10-bit packed
BAYER_BG12_PACKED	0x010C002D	GigE Vision specific format, Bayer Blue-Green 12-bit packed
BAYER_GB10_PACKED	0x010C0028	GigE Vision specific format, Bayer Green-Blue 10-bit packed
BAYER_GB12_PACKED	0x010C002C	GigE Vision specific format, Bayer Green-Blue 12-bit packed
BAYER_GR10_PACKED	0x010C0026	GigE Vision specific format, Bayer Green-Red 10-bit packed
BAYER_GR12_PACKED	0x010C002A	GigE Vision specific format, Bayer Green-Red 12-bit packed
BAYER_RG10_PACKED	0x010C0027	GigE Vision specific format, Bayer Red-Green 10-bit packed
BAYER_RG12_PACKED	0x010C002B	GigE Vision specific format, Bayer Red-Green 12-bit packed

3.2.6. GxFrameStatusList

Definition	Value	Explanation
SUCCESS	0	Normal frame
IMCOMPLETE	-1	Incomplete frame
INVALID_IMAGE_INFO	-2	Invalid image information

3.2.7. GxDeviceTemperatureSelectorEntry

Definition	Value	Explanation
SENSOR	1	Sensor
MAINBOARD	2	Mainboard

3.2.8. 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

3.2.9. 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

3.2.10. GxAcquisitionModeEntry

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

3.2.11. 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
TRIGGER	6	Triggering signal
MULTISOURCE	7	Multi source triggering
CXPTRIGGER0	8	CXP trigger source 0
CXPTRIGGER1	9	CXP trigger source 1

3.2.12. GxTriggerActivationEntry

Definition	Value	Explanation
FALLING_EDGE	0	Falling edge trigger
RISING_EDGE	1	Rising edge trigger
ANYEDGE	2	Rising or falling edge triggering
LEVELHIGH	3	High level trigger
LEVELLOW	4	Low level trigger

3.2.13. GxExposureModeEntry

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

3.2.14. GxUserOutputSelectorEntry

Definition	Value	Explanation
OUTPUT0	1	Output 0
OUTPUT1	2	Output 1
OUTPUT2	4	Output 2
OUTPUT3	5	Output 3
OUTPUT4	6	Output 4

OUTPUT5	7	Output 5
OUTPUT6	8	Output 6

3.2.15. GxUserOutputModeEntry

Definition	Value	Explanation
STROBE	0	Strobe
USER_DEFINED	1	User defined

3.2.16. GxGainSelectorEntry

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

3.2.17. 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

3.2.18. GxBalanceRatioSelectorEntry

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

3.2.19. GxAALightEnvironmentEntry

Definition	Value	Explanation
NATURE_LIGHT	0	Natural light

AC50HZ	1	50 Hz fluorescent lamp
AC60HZ	2	60 Hz fluorescent lamp

3.2.20. GxUserSetEntry

Definition	Value	Explanation
DEFAULT	0	Default parameter set
USER_SET0	1	User parameter set 0
USER_SET1	2	User parameter set 1

3.2.21. 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

3.2.22. 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

3.2.23. 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

3.2.24. GxTriggerSelectorEntry

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

3.2.25. 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
LINE11	12	Pin 11
LINE12	13	Pin 12
LINE13	14	Pin 13
LINE14	15	Pin 14
TRIGGER	16	Hardware triggered input
IO1	17	GPIO input
IO2	18	GPIO input
FLASH_P	19	Flash_B output
FLASH_W	20	Flash_W output

3.2.26. GxDeviceSerialPortBaudRateEntry

Definition	Value	Explanation
Baud9600	5	The serial port baud rate is 9600Hz
Baud19200	6	The serial port baud rate is 19200Hz
Baud38400	7	The serial port baud rate is 38400Hz
Baud76800	8	The serial port baud rate is 76800Hz
Baud115200	9	The serial port baud rate is 115200Hz

3.2.27. GxSerialPortStopBitsEntry

Definition	Value	Explanation
Bits1	0	Bit1
Bits1AndAHalf	1	Bit1AndHalf
Bits2	2	Bit2

3.2.28. GxLineModeEntry

Definition	Value	Explanation
INPUT	0	Input
OUTPUT	1	Output

3.2.29. 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
TIMER2_ACTIVE	13	Timer 2 activated
TIMER3_ACTIVE	14	Timer 3 activated
FRAME_TRIGGER	15	Frame trigger
Flash_W	16	Flash_w
Flash_P	17	Flash_P
SERIAL_PORT_0	18	SerialPort0

3.2.30. 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

3.2.31. GxLutSelectorEntry

Definition	Value	Explanation
LUMINANCE	0	Luminance

3.2.32. GxTransferControlModeEntry

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

3.2.33. GxTransferOperationModeEntry

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

3.2.34. GxTestPatternGeneratorSelectorEntry

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

3.2.35. GxChunkSelectorEntry

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

3.2.36. 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

3.2.37. 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

3.2.38. 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

3.2.39. GxAcquisitionStatusSelectorEntry

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

3.2.40. GxExposureTimeModeEntry

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

3.2.41. GxGammaModeEntry

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

3.2.42. 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)

3.2.43. GxColorTransformationModeEntry

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

3.2.44. 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

3.2.45. GxAutoEntry

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

3.2.46. GxSwitchEntry

Definition	Value	Explanation
OFF	0	Off
ON	1	On

3.2.47. GxSensorBitDepthEntry

Definition	Value	Explanation
BPP8	8	Bit 8
BPP10	10	Bit 10
BPP12	12	Bit 12

3.2.48. GxMultisourceSelectorEntry

Definition	Value	Explanation
Software	0	Software trigger
LINE0	1	Multiple trigger selection 0
LINE2	3	Multiple trigger selection 2
LINE3	4	Multiple trigger selection 3

3.2.49. GxDeviceTapGeometryEntry

Definition	Value	Explanation
GEOMETRY_1X_1Y	0	Geometry_1X_1Y
GEOMETRY1_X1_Y2	19	Geometry_1X_1Y2
GEOMETRY1_X2_YE	10	Geometry_1X_2YE

3.2.50. GxEncoderSelectorEntry

Definition	Value	Explanation
ENCODER0	0	Encoder selector 0
ENCODER1	1	Encoder selector 1
ENCODER2	2	Encoder selector 2

3.2.51. GxEncoderSourceAEntry

Definition	Value	Explanation
OFF	0	Encoder A related closed input
LINE0	1	Encoder A phase input Line0
LINE1	2	Encoder A phase input Line1
LINE2	3	Encoder A phase input Line2
LINE3	4	Encoder A phase input Line3
LINE4	5	Encoder A phase input Line4
LINE5	6	Encoder A phase input Line5

3.2.52. GxEncoderSourceBEntry

Definition	Value	Explanation
OFF	0	Encoder B related closed input
LINE0	1	Encoder B phase input Line0

LINE1	2	Encoder B phase input Line1
LINE2	3	Encoder B phase input Line2
LINE3	4	Encoder B phase input Line3
LINE4	5	Encoder B phase input Line4
LINE5	6	Encoder B phase input Line5

3.2.53. GxEncoderModeEntry

Definition	Value	Explanation
HIGH_RESOLUTION	0	Encoder Mode

3.2.54. GxEncoderDirectionEntry

Definition	Value	Explanation
FORWARD	0	Encoder direction forward
BACKWARD	1	Encoder direction backward

3.2.55. GxRegionSendModeEntry

Definition	Value	Explanation
SINGLE_ROI	0	Single ROI
MULTI_ROI	1	Multiple ROI

3.2.56. GxShadingCorrectionModeEntry

Definition	Value	Explanation
FLAT_FIELD_CORRECTION	0	Flat Field Correction
PARALLAX_CORRECTION	1	Parallax correction
TAILOR_FLAT_FIELD_CORRECTION	2	Customized flat field calibration
DEVICE_FLAT_FIELD_CORRECTION	3	Equipment flat field calibration

3.2.57. GxFFCGenerateStatusEntry

Definition	Value	Explanation
IDLE	0	Idle
WAITING_IMAGE	1	Waiting for image
FINISH	2	Finished

3.2.58. GxFFCCoefficientEntry

Definition	Value	Explanation
SET0	0	FFC coefficient Set0
SET1	1	FFC coefficient Set1
SET2	2	FFC coefficient Set2
SET3	3	FFC coefficient Set3
SET4	4	FFC coefficient Set4
SET5	5	FFC coefficient Set5
SET6	6	FFC coefficient Set6
SET7	7	FFC coefficient Set7
SET8	8	FFC coefficient Set8
SET9	9	FFC coefficient Set9
SET10	10	FFC coefficient Set10
SET11	11	FFC coefficient Set11
SET12	12	FFC coefficient Set12
SET13	13	FFC coefficient Set13
SET14	14	FFC coefficient Set14
SET15	15	FFC coefficient Set15

3.2.59. GxDSNUSelectorEntry

Definition	Value	Explanation
DEFAULT	0	Dark Signal Non-Uniformity coefficient Default
SET0	1	Dark Signal Non-Uniformity coefficient Set0
SET1	2	Dark Signal Non-Uniformity coefficient Set1
SET2	3	Dark Signal Non-Uniformity coefficient Set2
SET3	4	Dark Signal Non-Uniformity coefficient Set3
SET4	5	Dark Signal Non-Uniformity coefficient Set4
SET5	6	Dark Signal Non-Uniformity coefficient Set5
SET6	7	Dark Signal Non-Uniformity coefficient Set6
SET7	8	Dark Signal Non-Uniformity coefficient Set7
SET8	9	Dark Signal Non-Uniformity coefficient Set8
SET9	10	Dark Signal Non-Uniformity coefficient Set9
SET10	11	Dark Signal Non-Uniformity coefficient Set10

SET11	12	Dark Signal Non-Uniformity coefficient Set11
SET12	13	Dark Signal Non-Uniformity coefficient Set12
SET13	14	Dark Signal Non-Uniformity coefficient Set13
SET14	15	Dark Signal Non-Uniformity coefficient Set14
SET15	16	Dark Signal Non-Uniformity coefficient Set15

3.2.60. GxDSNUGenerateStatusEntry

Definition	Value	Explanation
IDLE	0	Idle
WAITING_IMAGE	1	Waiting for image
FINISH	2	Finished

3.2.61. GxPRNUSelectorEntry

Definition	Value	Explanation
DEFAULT	0	Photo Response Non-Uniformity coefficient Default
SET0	1	Photo Response Non-Uniformity coefficient Set0
SET1	2	Photo Response Non-Uniformity coefficient Set1
SET2	3	Photo Response Non-Uniformity coefficient Set2
SET3	4	Photo Response Non-Uniformity coefficient Set3
SET4	5	Photo Response Non-Uniformity coefficient Set4
SET5	6	Photo Response Non-Uniformity coefficient Set5
SET6	7	Photo Response Non-Uniformity coefficient Set6
SET7	8	Photo Response Non-Uniformity coefficient Set7
SET8	9	Photo Response Non-Uniformity coefficient Set8
SET9	10	Photo Response Non-Uniformity coefficient Set9
SET10	11	Photo Response Non-Uniformity coefficient Set10
SET11	12	Photo Response Non-Uniformity coefficient Set11
SET12	13	Photo Response Non-Uniformity coefficient Set12
SET13	14	Photo Response Non-Uniformity coefficient Set13
SET14	15	Photo Response Non-Uniformity coefficient Set14
SET15	16	Photo Response Non-Uniformity coefficient Set15

3.2.62. GxPRNUGenerateStatusEntry

Definition	Value	Explanation
IDLE	0	Idle
WAITING_IMAGE	1	Waiting for image
FINISH	2	Finished

3.2.63. GxCXPLinkConfigurationEntry

Definition	Value	Explanation
CXP6_X1	0x00010048	CXP connection configuration CXP6_X1
CXP12_X1	0x00010058	CXP connection configuration CXP12_X1
CXP6_X2	0x00020048	CXP connection configuration CXP6_X2
CXP12_X2	0x00020058	CXP connection configuration CXP12_X2
CXP6_X4	0x00040048	CXP connection configuration CXP6_X4
CXP12_X4	0x00040058	CXP connection configuration CXP12_X4
CXP3_X1	0x00010038	CXP connection configuration CXP3_X1
CXP3_X2	0x00020038	CXP connection configuration CXP3_X2
CXP3_X4	0x00040038	CXP connection configuration CXP3_X4

3.2.64. GxCXPLinkConfigurationPreferredEntry

Definition	Value	Explanation
CXP12_X4	0x00040058	Preset connection configuration CXP12_X4

3.2.65. GxCXPLinkConfigurationStatusEntry

Definition	Value	Explanation
CXP6_X1	0x00010048	CXP connection configuration statusCXP6_X1
CXP12_X1	0x00010058	CXP connection configuration status CXP12_X1
CXP6_X2	0x00020048	CXP connection configuration status CXP6_X2
CXP12_X2	0x00020058	CXP connection configuration status CXP12_X2
CXP6_X4	0x00040048	CXP connection configuration status CXP6_X4
CXP12_X4	0x00040058	CXP connection configuration status CXP12_X4
CXP3_X1	0x00010038	CXP connection configuration status CXP3_X1
CXP3_X2	0x00020038	CXP connection configuration status CXP3_X2
CXP3_X4	0x00040038	CXP connection configuration status CXP3_X4

3.2.66. GxCXPConectionSelectorEntry

Definition	Value	Explanation
SELECTOR0	0	Connection selection 0
SELECTOR1	1	Connection selection 1
SELECTOR2	2	Connection selection 2
SELECTOR3	3	Connection selection 3

3.2.67. GxCXPConectionTestModeEntry

Definition	Value	Explanation
OFF	0	Close the connection test mode
MODE1	1	Trigger the connection test mode

3.2.68. GxSequencerFratureSelectorEntry

Definition	Value	Explanation
FLAT_FIELD_CORRECTION	0	Sequence function selection

3.2.69. GxSequencerTriggerSourceEntry

Definition	Value	Explanation
FRAME_START	7	Sequence trigger source FrameStart

3.2.70. 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

3.2.71. GxTimerSelectorEntry

Definition	Value	Explanation
TIMER1	1	Timer 1
TIMER2	2	Timer 2
TIMER3	3	Timer 3

3.2.72. GxTimerTriggerSourceEntry

Definition	Value	Explanation
EXPOSURE_START	1	Exposure start
LINE10	10	Start when receive pin 10 signal
LINE14	14	Start when receive pin 14 signal
STROBE	16	Start when receive strobe signal

3.2.73. GxNoiseReductionModeEntry

Definition	Value	Explanation
OFF	0	Turn off 2D noise reduction mode
LOW	1	Low
MIDDLE	2	Middle
HIGH	3	High

3.2.74. GxHDRModeEntry

Definition	Value	Explanation
OFF	0	Turn off HDR mode
CONTINUOUS	1	Continuous HDR mode

3.2.75. GxMGCMModeEntry

Definition	Value	Explanation
OFF	0	Disable multi frame grayscale control mode
TWO_FRAME	1	Two frame grayscale control mode
FOUR_FRAME	2	Four frame grayscale control mode

3.2.76. GxAcquisitionBurstModeEntry

Definition	Value	Explanation
STANDARD	0	Standard mode
HIGH_SPEED	1	High speed mode

3.2.77. GxSensorSelectorEntry

Definition	Value	Explanation
CMOS1	0	Select CMOS1 sensor
CCD1	1	Select CCD1 sensor

3.2.78. GxIMUConfigAccRangeEntry

Definition	Value	Explanation
ACC_16G	2	The accelerometer has a measurement range of 16g
ACC_8G	3	The accelerometer has a measurement range of 8g
ACC_4G	4	The accelerometer has a measurement range of 4g
ACC_2G	5	The accelerometer has a measurement range of 2g

3.2.79. GxIMUConfigAccOdrEntry

Definition	Value	Explanation
ODR_1000HZ	0	The accelerometer output data rate is 1000Hz
ODR_500HZ	1	The accelerometer output data rate is 500Hz
ODR_250HZ	2	The accelerometer output data rate is 250Hz
ODR_125HZ	3	The accelerometer output data rate is 125Hz
ODR_63HZ	4	The accelerometer output data rate is 63Hz
ODR_31HZ	5	The accelerometer output data rate is 31Hz
ODR_16HZ	6	The accelerometer output data rate is 16Hz
ODR_2000HZ	8	The accelerometer output data rate is 2000Hz
ODR_4000HZ	9	The accelerometer output data rate is 4000Hz
ODR_8000HZ	10	The accelerometer output data rate is 8000Hz

3.2.80. GxIMUConfigAccOdrLowPassFilterFrequencyEntry

Definition	Value	Explanation
ODR040	0	The low-pass cut-off frequency of the accelerometer is $ODR \times 0.40$

ODR025	1	The low-pass cut-off frequency of the accelerometer is $ODR \times 0.25$
ODR011	2	The low-pass cut-off frequency of the accelerometer is $ODR \times 0.11$
ODR004	3	The low-pass cut-off frequency of the accelerometer is $ODR \times 0.04$
ODR002	4	The low-pass cut-off frequency of the accelerometer is $ODR \times 0.02$

3.2.81. GxIMUConfigGyroRangeEntry

Definition	Value	Explanation
RANGE_125DPS	2	The measurement range of gyroscope in the X direction is 125dps
RANGE_250DPS	3	The measurement range of gyroscope in the X direction is 250dps
RANGE_500DPS	4	The measurement range of gyroscope in the X direction is 500dps
RANGE_1000DPS	5	The measurement range of gyroscope in the X direction is 1000dps
RANGE_2000DPS	6	The measurement range of gyroscope in the X direction is 2000dps

3.2.82. GxIMUConfigGyroOdrEntry

Definition	Value	Explanation
ODR_1000HZ	0	The accelerometer output data rate is 1000Hz
ODR_500HZ	1	The accelerometer output data rate is 500Hz
ODR_250HZ	2	The accelerometer output data rate is 250Hz
ODR_125HZ	3	The accelerometer output data rate is 125Hz
ODR_63HZ	4	The accelerometer output data rate is 63Hz
ODR_31HZ	5	The accelerometer output data rate is 31Hz
ODR_4KHZ	9	The accelerometer output data rate is 4KHz
ODR_8KHZ	10	The accelerometer output data rate is 8KHz
ODR_16KHZ	11	The accelerometer output data rate is 16KHz
ODR_32KHZ	12	The accelerometer output data rate is 32KHz

3.2.83. GxIMUConfigGyroOdrLowPassFilterFrequencyEntry

Definition	Value	Explanation
GYROLPF2000HZ	2000	The low-pass cutoff frequency of the accelerometer is 2000Hz
GYROLPF1600HZ	1600	The low-pass cutoff frequency of the accelerometer is 1600Hz
GYROLPF1525HZ	1525	The low-pass cutoff frequency of the accelerometer is 1525Hz
GYROLPF1313HZ	1313	The low-pass cutoff frequency of the accelerometer is 1313Hz

GYROLPF1138HZ	1138	The low-pass cutoff frequency of the accelerometer is 1138Hz
GYROLPF1000HZ	1000	The low-pass cutoff frequency of the accelerometer is 1000Hz
GYROLPF863HZ	863	The low-pass cutoff frequency of the accelerometer is 863Hz
GYROLPF638HZ	638	The low-pass cutoff frequency of the accelerometer is 638Hz
GYROLPF438HZ	438	The low-pass cutoff frequency of the accelerometer is 438Hz
GYROLPF313HZ	313	The low-pass cutoff frequency of the accelerometer is 313Hz
GYROLPF213HZ	213	The low-pass cutoff frequency of the accelerometer is 213Hz
GYROLPF219HZ	219	The low-pass cutoff frequency of the accelerometer is 219Hz
GYROLPF363HZ	363	The low-pass cutoff frequency of the accelerometer is 363Hz
GYROLPF320HZ	320	The low-pass cutoff frequency of the accelerometer is 320Hz
GYROLPF250HZ	250	The low-pass cutoff frequency of the accelerometer is 250Hz
GYROLPF200HZ	200	The low-pass cutoff frequency of the accelerometer is 200Hz
GYROLPF181HZ	181	The low-pass cutoff frequency of the accelerometer is 181Hz
GYROLPF160HZ	160	The low-pass cutoff frequency of the accelerometer is 160Hz
GYROLPF125HZ	125	The low-pass cutoff frequency of the accelerometer is 125Hz
GYROLPF100HZ	100	The low-pass cutoff frequency of the accelerometer is 100Hz
GYROLPF90HZ	90	The low-pass cutoff frequency of the accelerometer is 90Hz
GYROLPF80HZ	80	The low-pass cutoff frequency of the accelerometer is 80Hz
GYROLPF63HZ	63	The low-pass cutoff frequency of the accelerometer is 63Hz
GYROLPF50HZ	50	The low-pass cutoff frequency of the accelerometer is 50Hz
GYROLPF45HZ	45	The low-pass cutoff frequency of the accelerometer is 45Hz
GYROLPF40HZ	40	The low-pass cutoff frequency of the accelerometer is 40Hz
GYROLPF31HZ	31	The low-pass cutoff frequency of the accelerometer is 31Hz
GYROLPF25HZ	25	The low-pass cutoff frequency of the accelerometer is 25Hz
GYROLPF23HZ	23	The low-pass cutoff frequency of the accelerometer is 23Hz
GYROLPF20HZ	20	The low-pass cutoff frequency of the accelerometer is 20Hz
GYROLPF15HZ	15	The low-pass cutoff frequency of the accelerometer is 15Hz
GYROLPF13HZ	13	The low-pass cutoff frequency of the accelerometer is 13Hz
GYROLPF11HZ	11	The low-pass cutoff frequency of the accelerometer is 11Hz
GYROLPF10HZ	10	The low-pass cutoff frequency of the accelerometer is 10Hz

GYROLPF8HZ	8	The low-pass cutoff frequency of the accelerometer is 8Hz
GYROLPF6HZ	6	The low-pass cutoff frequency of the accelerometer is 6Hz

3.2.84. GxIMUTemperatureOdrEntry

Definition	Value	Explanation
ODR_500HZ	0	The output data rate of the thermometer is 500Hz
ODR_250HZ	1	The output data rate of the thermometer is 250Hz
ODR_125HZ	2	The output data rate of the thermometer is 125Hz
ODR_63HZ	3	The output data rate of the thermometer is 63Hz

3.2.85. GxSerialportSelectorEntry

Definition	Value	Explanation
SERIALPOR0	0	Serial port 0

3.2.86. GxSerialportSourceEntry

Definition	Value	Explanation
OFF	0	Serial input source switch
LINE0	1	Serial input source 0
LINE1	2	Serial input source 1
LINE2	3	Serial input source 2
LINE3	4	Serial input source 3

3.2.87. GxSerialportBaundrateEntry

Definition	Value	Explanation
BAUNDRATE_9600	5	The serial port baud rate is 9600Hz
BAUNDRATE_19200	6	The serial port baud rate is 19200Hz
BAUNDRATE_38400	7	The serial port baud rate is 38400Hz
BAUNDRATE_76800	8	The serial port baud rate is 76800Hz
BAUNDRATE_115200	9	The serial port baud rate is 115200Hz

3.2.88. GxSerialporeStopBitsEntry

Definition	Value	Explanation
ONE	0	Bit1

ONEANDHALF	1	Bit1AndHalf
TWO	2	Bit2

3.2.89. GxSerialportParityEntry

Definition	Value	Explanation
NONE	0	None
ODD	1	Odd
EVEN	2	Even
MARK	3	Mark
SPACE	4	Space

3.2.90. GxCounterSelectorEntry

Definition	Value	Explanation
COUNTER1	1	Counter 1
COUNTER2	2	Counter 2

3.2.91. 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

3.2.92. 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
CXPTRIGGER0	8	CxpTrigger0
CXPTRIGGER1	9	CxpTrigger1

3.2.93. GxCounterResetActivationEntry

Definition	Value	Explanation
RISINGEDGE	1	Rising edge counter reset

3.2.94. 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

3.2.95. GxTimerTriggerActivationEntry

Definition	Value	Explanation
RISINGEDGE	1	Rising edge counter reset

3.2.96. GxStopAcquisitionModeEntry

Definition	Value	Explanation
GENERAL	0	General
LIGHT	1	Light

3.2.97. GxDSSStreamBufferHandlingModeEntry

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

3.2.98. GxResetDeviceModeEntry

Definition	Value	Explanation
RECONNECT	1	Reconnect device
RESET	2	Reset device

3.2.99. Dx BayerConvertType

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

3.2.100. 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

BIT4_11	4	4-11 bits
BIT5_12	5	5-12 bits
BIT6_13	6	6-13 bits
BIT7_14	7	7-14 bits
BIT8_15	8	8-15 bits

3.2.101. DxImageMirrorMode

Definition	Value	Explanation
HORIZONTAL_MIRROR	0	Horizontal mirroring
VERTICAL_MIRROR	1	Vertical mirroring

3.2.102. DxRGBChannelOrder

Definition	Value	Explanation
ORDER_RGB	0	RGB channel
ORDER_BGR	1	BGR channel

3.2.103. GxTLClassList

Definition	Value	Explanation
TL_TYPE_UNKNOWN	0	Unknown TL type
TL_TYPE_USB	1	USB TL type
TL_TYPE_GEV	2	GEV TL type
TL_TYPE_U3V	4	U3V TL type
TL_TYPE_CXP	8	CXP TL type

3.2.104. GxImageInfo

Definition	Explanation
image_width	Image width
image_height	Image height
image_buf	Image buffer
image_pixel_format	Image format

3.2.105. GxIPConfigureModeList

Definition	Explanation
DHCP	Enable DHCP to allocate IP through DHCP service
LLA	Enable LLA mode for IP allocation
STATIC_IP	Enable static IP mode to configure IP addresses
DEFAULT	Enable default mode to configure IP addresses

3.2.106. GxActionCommandResult

Definition	Explanation
device_ip	The current device IP that returns ack (Dotted Decimal Notation IPv4)
status	<p>The ACTION status returned by the device, the error originates from the GigaVision protocol</p> <p>0: Command sent successfully.</p> <p>0x8013: The device is not time synchronized with the master clock. Before executing this interface, "PtpEnable" must be enabled and "PtpStatus" must be set to "Master" or "Slave" (indicating synchronization with the master clock).</p> <p>0x8015: The device queue or packet data has overflowed. When the device corresponding to strDeviceAddress is executing the previous issue_scheduled_action_command request and receives a new issue_scheduled_action_command, this error will be returned.</p> <p>0x8016: The Scheduled Action Command issued by issue_scheduled_action_command is outdated.</p>

3.2.107. GxRegisterStackEntry

Definition	Definition
address	Register address
buffer	Register memory pointer
size	Register memory size

3.2.108. ColorTransformFactor

Definition	Default	Range	Definition
fGain00	1.0	-4.0~4.0	Red channel gain applied to red pixels
fGain01	0.0	-4.0~4.0	Green channel gain applied to red pixels
fGain02	0.0	-4.0~4.0	Blue channel gain applied to red pixels
fGain10	0.0	-4.0~4.0	Red channel gain applied to green pixels
fGain11	1.0	-4.0~4.0	Green channel gain applied to green pixels
fGain12	0.0	-4.0~4.0	Blue channel gain applied to green pixels

fGain20	0.0	-4.0~4.0	Red channel gain applied to blue pixels
fGain21	0.0	-4.0~4.0	Green channel gain applied to blue pixels
fGain22	1.0	-4.0~4.0	Blue channel gain applied to blue pixels

3.3. Module interface definition

3.3.1. DeviceManager

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

Interface list:

set_log_type	Set the type of log generated
get_log_type	Get the generated log type
update_device_list	Enumerate devices on the same network segment
update_all_device_list	Enumerate devices on different network segments
update_device_list_ex	Enumerate corresponding devices according to the specified TL type
get_device_number()	Get the number of devices
get_device_info()	Get the information of devices
get_interface_number()	Get the number of Interface
get_interface_info()	Get the information of Interface
get_interface()	Get Interface Object
open_device_by_sn	Open the device by serial number
open_device_by_user_id	Open the device by user ID
open_device_by_index	Open the device by device index
open_device_by_ip	Open the device by IP address
open_device_by_mac	Open the device by mac address
gige_reset_device	Reconnect device or reset device
gige_force_ip	Set device ForceIP
gige_ip_configuration	Configure IP
create_image_format_convert()	Create image pixel format conversion objects
create_image_process()	Create image processing objects
issue_action_command	Send regular ACTION
issue_scheduled_action_command	Send scheduled ACTION

3.3.1.1. set_log_type

Statement: DeviceManager.set_log_type (log_type)

Significance: Set the type of log generated.

Formal parameter: [in] log_type The type of log to be set, refer to [GxLogTypeList](#)
Eg: GX_LOG_TYPE_INFO | GX_LOG_TYPE_WARN.

Return value: None.

Exception handling: If the input parameter is not an integer value, throw ParameterTypeError exception.

3.3.1.2. get_log_type

Statement: DeviceManager.get_log_type ()

Significance: Get the generated log type.

Formal parameter: None.

Return value: Return the type of log obtained, please refer to [GxLogTypeList](#).
Eg: GX_LOG_TYPE_INFO | GX_LOG_TYPE_WARN.

Exception handling: None.

3.3.1.3. update_device_list

Statement: DeviceManager.update_device_list(timeout=200)

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

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

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

Exception handling:

- 1) If the input parameter is not an int value, a ParameterTypeError exception is thrown.
- 2) If the input parameter is less than 0 or greater than the maximum value of the unsigned int, then prints **DeviceManager.update_device_list: Out of bounds, timeout:minimum=0, maximum= 0xffffffff**, and the function returns **None**.
- 3) If the enumeration of device on the same segment fails, an exception is thrown. For details, see [Error handling](#).

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

3.3.1.4. 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](#).

3.3.1.5. update_device_list_ex

Statement: DeviceManager.update_device_list_ex (tl_type, timeout=2000)

Significance: Enumerate the corresponding devices according to the specified TL type.

Parameter:

[in] tl_type	TL type, see details in GxTLClassList
[in] timeout	Enumeration timeout [0, 0xffffffff], 2000 (ms) by default

Return value: Enumerate the number of devices and the 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 dictionary, and the key names in the dictionary can be found in [Enumeration device](#).

Exception handling: If obtaining basic information of all devices is unsuccessful, an exception will be thrown, and the type of exception can be found in [Error handling](#).

3.3.1.6. get_device_number

Statement: DeviceManager.get_device_number()

Significance: Get the number of devices.

Return value: The number of devices.

3.3.1.7. 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](#).

3.3.1.8. get_interface_number

Statement: DeviceManager.get_interface_number ()

Significance: Get the number of Interface.

Return value: The number of Interface.

3.3.1.9. get_interface_info

Statement: DeviceManager.get_interface_info()

Significance: Get the information of Interface.

Return value: Interface information list. The number of elements in the device information list is the number of interfaces enumerated. The data type of the elements in the list is dictionary, and the keys include:

Key Name	Description	Type
type	TL type	GxTLClassList
display_name	Display name	String
interface_id	Interface ID	String
serial_number	Device serial number	String
description	Device display name	String
init_flag	Initial identification (Only for CXP frame grabbers)	Int
reserved	User defined name	String array

3.3.1.10. get_interface

Statement: DeviceManager.get_interface()

Significance: Get Interface object.

Return value: Interface object, see details in [Interface](#).

3.3.1.11. 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 GxAccessMode.CONTROL , check GxAccessMode

Return value: Device object.

Exception handling:

- 1) If the input parameter 1 is not a string, a **ParameterTypeError** exception is thrown.
- 2) If the input parameter 2 is not an int, a **ParameterTypeError** exception is thrown.
- 3) If input parameter 2 is not in device mode [GxAccessMode](#), then prints interface name, open device mode is not in the range and the enumeration value information supported by the current parameter, and the function returns **None**.
- 4) If repeatedly getting the device class unsuccessfully, a **NotFoundDevice** exception is thrown.
- 5) If opening the device unsuccessfully, an exception is thrown. For details, see [Error handling](#).
- 6) If the got device is not one of the U3V/USB2/GEV classes, a **NotFoundDevice** exception is thrown.

3.3.1.12. 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.

3.3.1.13. 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.

3.3.1.14. 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](#).

3.3.1.15. open_device_by_mac

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

Significance: Open the GigE Vision camera by device mac address.

Formal parameter:

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

Return value: Device object.

Exception handling:

- 1) If the input parameter 1 is not a string, a **ParameterTypeError** exception is thrown.
- 2) If the input parameter 2 is not an int, a **ParameterTypeError** exception is thrown.
- 3) If the input parameter 2 is not in device mode [GxAccessMode](#), then prints interface name, open device mode is not in the range and the enumeration value information supported by the current parameter, the function returns **None**.
- 4) If opening the device unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.3.1.16. 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](#).

3.3.1.17. gige_force_ip

Statement: DeviceManager.gige_force_ip (mac_address, ip_address, subnet_mask, default_gate_way)

Significance: Execute force ip operation.

Parameter:

[in]	mac_address	Mac address
[in]	ip_address	IP address
[in]	subnet_mask	Subnet mask
[in]	default_gate_way	Default gateway

Exception handling: If the call is unsuccessful, an exception is thrown. For details, see [Error handling](#).

3.3.1.18. gige_ip_configuration

Statement: DeviceManager.gige_ip_configuration(mac_address, ipconfig_flag,
ip_address, subnet_mask,
default_gateway, user_id)

Significance: Perform IP configuration operations.

Parameter:

[in]	mac_address	Mac address
------	-------------	-------------

[in]	ip_address	IP address
[in]	subnet_mask	Subnet mask
[in]	default_gate_way	Default gateway
[in]	ipconfig_flag	IP configuration mode, see details in GxIPConfigureModeList
[in]	user_id	User ID

Exception handling: If the call is unsuccessful, an exception is thrown. For details, see [Error handling](#).

3.3.1.19. create_image_format_convert

Statement: DeviceManager.create_image_format_convert()

Significance: Create an image pixel format conversion object.

Return value: Pixel format conversion object, see details in [ImageFormatConvert](#).

3.3.1.20. create_image_process

Statement: DeviceManager.create_image_process()

Significance: Create image processing objects.

Return value: Image processing object, see details in [ImageProcess](#).

3.3.1.21. issue_action_command

Statement: DeviceManager.issue_action_command(self, device_key, group_key,
group_mask, broadcast_address,
special_address, time_out,
expect_ack_number_res)

Significance: All cameras on the network execute “action” simultaneously by sending ActionCommand.

Return value: Actual returned ACK list, see details in [GxActionCommandResult](#).

Parameter:

[in]	device_key	Device secret key, corresponding to the “ActionDeviceKey” attribute of the device
[in]	group_key	Group secret key, corresponding to the “ActionGroupKey” attribute of the device
[in]	group_mask	Group mask, corresponding to the “ActionGroupMask” attribute of the device, can not be 0
[in]	broadcast_address	The destination IP for sending the “action” command can be broadcast, subnet broadcast, or unicast
[in]	special_address	The source IP address for sending the “action” command is used to clearly indicate which network adapter is sending the command from. If not specified, each IP of all network adapters will send the current “action” command. Please fill in an empty string without specifying the network port
[in]	time_out	0 means there is no need to wait for ack to return; Non zero indicates the maximum waiting time for ack return, and after

		the time is up, the parameter pNumResults returns the actual number of ack received
[in]]	expect_ack_number_res	Indicate the expected number of ack returns. This value represents the expected number of ACKs that the user expects to receive in feedback from the device executing the “action” command. If time_out is 0, ignore this parameter. Therefore, if time_out is 0, this parameter can be NULL

Exception handling: If the call is unsuccessful, an exception is thrown. For details, see [Error handling](#).

3.3.1.22. issue_scheduled_action_command

Statement: DeviceManager.issue_scheduled_action_command(self, device_key, group_key, group_mask, action_time, broadcast_address, special_address, time_out, expect_ack_number_res)

Significance: By sending ActionCommand at an absolute point in time, all cameras on the network can simultaneously execute “action”.

Return value: Actual returned ACK list, see details in [GxActionCommandResult](#).

Parameter:

[in]	device_key	Device secret key, corresponding to the “ActionDeviceKey” attribute of the device
[in]	group_key	Group secret key, corresponding to the “ActionGroupKey” attribute of the device
[in]	group_mask	Group mask, corresponding to the “ActionGroupMask” attribute of the device, can not be 0
[in]	action_time	Execution time (in nanoseconds). The actual value can be the master clock value used plus the expected delay time. The master clock value of a set of synchronized camera devices can be obtained by locking the timestamp value get_command_feature("TimestampLatch").send_command () from a set of camera devices and reading the timestamp value get_int_feature("TimestampLatchValue").get()
[in]	broadcast_address	The destination IP for sending the “action” command can be broadcast, subnet broadcast, or unicast
[in]	special_address	The source IP address for sending the “action” command is used to clearly indicate which network adapter is sending the command from. If not specified, each IP of all network adapters will send the current “action” command. Please fill in an empty string without specifying the network port
[in]	time_out	0 means there is no need to wait for ack to return; Non zero indicates the maximum waiting time for ack return, and after the time is up, the parameter pNumResults returns the actual number of ack received
[in]	expect_ack_number_res	Indicate the expected number of ack returns. This value

represents the expected number of ACKs that the user expects to receive in feedback from the device executing the “action” command. If time_out is 0, ignore this parameter. Therefore, if time_out is 0, this parameter can be NULL

Exception handling: If the call is unsuccessful, an exception is thrown. For details, see [Error handling](#).

3.3.2. 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()	Obtain the number of streaming channels supported by the current device
get_parent_interface	Retrieve the interface handle to which the device belongs
stream_on()	Send start command, camera starts transmitting image data
stream_off()	Send end command, camera ends transmitting image data
export_config_file()	Export the current configuration file
import_config_file()	Import the current configuration file
close_device()	Close the device, destroy the device handle, and leave the handle empty
register_device_offline_callback()	Registering a callback function for disconnections
unregister_device_offline_callback()	Cancel device disconnection callback function
get_stream_channel_num()	Obtain the number of flow channels
get_stream()	Get stream object
get_local_device_feature_control()	Get local attribute controller
get_remote_device_feature_control()	Get remote attribute controller
register_device_feature_callback_by_string()	Update callback function through string registration function properties
unregister_device_feature_callback_by_string()	Update callback function through string deregistration function properties
create_image_process_config()	Create image processing configuration objects
set_device_persistent_ip_address	Set the permanent IP information of the device
get_device_persistent_ip_address	Get the permanent IP information of the device
read_remote_device_port()	Read remote register address value (no longer maintained)
write_remote_device_port()	Write remote register address value (no longer maintained)

	maintained)
read_remote_device_port_stacked()	Batch reading of user specified register values (no longer maintained)
write_remote_device_port_stacked()	Batch write user specified register values (no longer maintained)
register_device_feature_callback()	Register function properties through function code to update callback function (no longer maintained)
unregister_device_feature_callback()	Update callback function through function code to cancel function properties (no longer maintained)

3.3.2.1. 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.

3.3.2.2. get_parent_interface

Statement: Device.get_parent_interface()

Significance: Retrieve the interface handle to which the device belongs.

Return value: The interface handle to which the device belongs.

Exception handling: If sending start command unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.3.2.3. stream_on

Statement: Device.stream_on()

Significance: Send a start command, the camera starts transmitting image data.

Return value: None.

Exception handling: If sending start command unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.3.2.4. 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](#).

3.3.2.5. 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](#).

3.3.2.6. 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](#).

3.3.2.7. 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: 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.

3.3.2.8. 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](#).

3.3.2.9. unregister_device_offline_callback

Statement: Device.unregister_device_offline_callback()

Significance: Unregister the device offline callback.

Return value: None.

Exception handling: If unregister offline callback unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.3.2.10. get_stream_channel_num

Statement: Device.get_stream_channel_num()

Significance: Obtain the number of flow channels.

Return value: Number of flow channels.

3.3.2.11. get_stream

Statement: Device.get_stream(stream_index)

Significance: Get the stream object.

Parameter: [in] stream_index Stream array index value, starting from 1

Return value: Stream object, see details in [DataStream](#).

Exception handling: If the call is unsuccessful, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.12. get_local_device_feature_control

Statement: Device.get_local_device_feature_control()

Significance: Get the local property controller.

Return value: Attribute controller object, see details in [FeatureControl](#).

Exception handling: If the call is unsuccessful, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.13. get_remote_device_feature_control

Statement: Device.get_remote_device_feature_control()

Significance: Get the remote property controller.

Return value: Attribute controller object, see details in [FeatureControl](#).

3.3.2.14. register_device_feature_callback_by_string

Statement: Device.register_device_feature_callback_by_string(callback_func, feature_name, args)

Significance: Update callback functions through string registration function properties.

Parameter:

[in]	callback_func	Property update callback function
[in]	feature_name	Function string node
[in]	args	The user parameter can be None

Return value: Callback function handle.

Exception handling: If the registration callback function fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.15. unregister_device_feature_callback_by_string

Statement: Device.unregister_device_feature_callback_by_string(feature_name,
feature_callback_handle)

Significance: Update the callback function by logging out the function attribute through the function code.

Parameter:

[in]	feature_name	Function string node
[in]	feature_callback_handle	Callback function handle

Return value: None.

Exception handling: If the logout callback function fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.16. create_image_process_config()

Statement: Device.create_image_process_config()

Significance: Create an image processing configuration object.

Return value: Image processing configuration object, see details in [ImageProcessConfig](#).

Exception handling: If the input parameter does not support the function node "ColorCorrectionParam", or supports but is not readable, an UnexpectedError exception is thrown.

3.3.2.17. set_device_persistent_ip_address

Statement: Device.set_device_persistent_ip_address(ip, subnet_mask, default_gate_way)

Significance: Set the permanent IP information of the device.

Parameter:

[in]	ip	Ip address
[in]	subnet_mask	Subnet Mask
[in]	default_gate_way	Default gateway

Return value: None.

Exception handling: If setting the IP fails, throw an exception, and the type of exception can be found in [Error handling](#).

3.3.2.18. get_device_persistent_ip_address

Statement: Device.get_device_persistent_ip_address()

Significance: Obtain the permanent IP information of the device.

Return value: The IP address, subnet mask, and gateway of the device.

Exception handling: If setting the IP fails, throw an exception, and the type of exception can be found in [Error handling](#).

3.3.2.19. read_remote_device_port (No longer maintained)

Recommend using [FeatureControl.read_port\(\)](#)

Statement: Device.read_remote_device_port(address, buff, size)

Significance: Read the value of the user specified register.

Parameter:

[in]	address	Register Address
[out]	buff	Return the cache address of the register content
[in out]	size	The buffer size requested by the user, and the actual size will be returned after the call is completed

Exception handling: If calling the function fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.20. write_remote_device_port (No longer maintained)

Recommend using [FeatureControl.write_port\(\)](#)

Statement: Device.write_remote_device_port(address, buf, size)

Significance: Write the user specified data to the user specified register.

Parameter:

[in]	address	Register address
[in]	buf	The cache address of the register content to be written
[in]	size	The buffer size requested by the user, and the actual size will be returned after the call is completed

Exception handling: If the function call fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.21. read_remote_device_port_stacked (No longer maintained)

Recommend using [FeatureControl.read_port_stacked\(\)](#)

Statement: Device.read_remote_device_port_stacked(entries, size)

Significance: Batch read the value of the user specified register (only for registers with a command value of 4 bytes in length).

Parameter:

[in]	entries	Batch read register structure address
[in out]	size	Read the number of device registers

Exception handling: If the function call fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.22. write_remote_device_port_stacked (No longer maintained)

Recommend using [FeatureControl.write_port_stacked\(\)](#)

Statement: Device.write_remote_device_port_stacked(entries, size)

Significance: Batch write user specified data to user specified registers (limited to registers with a command value of 4 bytes in length)

Parameter:

[in]	entries	Batch write register structure address
[in]	size	Write the number of device registers

Exception handling: If the function call fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.23. register_device_feature_callback (No longer maintained)

Recommend using [register_device_feature_callback_by_string](#)

Statement: Device.register_device_feature_callback(callback_func, feature_id, args)

Significance: Register function properties through function codes to update callback functions.

Parameter:

[in]	callback_func	Property update callback function
[in]	feature_id	Function code node
[in]	args	The user parameter can be None

Return value: Callback function handle.

Exception handling: If the function call fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.2.24. unregister_device_feature_callback (No longer maintained)

Recommend using [unregister_device_feature_callback_by_string](#)

Statement: Device.unregister_device_feature_callback(feature_id, feature_callback_handle)

Significance: Update the callback function by logging out the function attribute through the function code.

Parameter:

[in]	feature_id	Function code node
[in]	feature_callback_handle	Callback function handle

Return value: None.

Exception handling: If the logout callback function fails, an exception is thrown, and the type of exception can be found in [Error handling](#).

3.3.3. Interface

Encapsulate the Interface class.

Interface List:

interface_info()	Interface information
feature_control()	Interface property control object

3.3.3.1. get_interface_info

Statement: get_interface_info()

Significance: Get Interface information.

Return value: Return the Interface information dictionary, where the keys include:

Key Name	Description	Type
type	TL type	GxTLClassList
display_name	Display name	String
interface_id	Interface ID	String
serial_number	Device serial number	String
description	Device display name	String
init_flag	Initial identification (only for CXP frame grabber)	Int
reserved	User defined name	String array

3.3.3.2. get_feature_control

Statement: get_feature_control()

Significance: Get the Interface property control object.

Return value: Interface property control object, see details in [FeatureControl](#).

3.3.4. DataStream

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

Interface list:

set_acquisition_buffer_number()	Set the size of the acquisition buffer
get_image()	Get the image and successfully create the image class object
dq_buf()	Zero copy for getting images
q_buf()	Return the image object to the acquisition system
flush_queue()	Clear camera acquire buffer queue
register_capture_callback()	Register capture callback function
unregister_capture_callback()	Unregister capture callback function
get_featrue_control()	Get property control object
get_payload_size	Get payloadsize
register_buffer()	The buffer applied by the registered user is used for internal collection
unregister_buffer()	Unregister the buffer registered using the registrant buffer() interface

3.3.4.1. 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](#).

3.3.4.2. 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**.

3.3.4.3. dq_buf

Statement: `DataStream.dq_buf(timeout=1000)`

Significance: Zero copy image acquisition, successfully created image class object.

Formal parameter: [in] `timeout` Get timeout [0, 0xffffffff], default value: 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, dq_buf**, and the function returns **None**.

3.3.4.4. q_buf

Statement: DataStream.q_buf(image)

Significance: Return the image object to the acquisition system.

Formal parameter: [in] image image object obtained by dq_buf

Return value: None.

Exception handling:

- 1) If the input parameter is not RawImage, a **ParameterTypeError** exception is thrown.
- 2) If the image acquisition has not started yet, print an error message and return directly.
- 3) If the callback function is already registered, throw an InvalidCall exception.

3.3.4.5. flush_queue

Statement: DataStream.flush_queue()

Significance: Clear the camera acquisition buffer queue

Exception handling: If clearing the camera acquisition buffer queue unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.3.4.6. 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.

3.3.4.7. unregister_capture_callback

Statement: Device.unregister_capture_callback()

Significance: Unregister the capture callback. [Using callback](#)

Return value: None.

Exception handling: If unregister capture callback unsuccessfully, an exception is thrown. For details, see [Error handling](#).

3.3.4.8. get_featrue_control

Statement: DataStream.get_featrue_control()

Significance: Get the flow property control object.

Return value: Flow attribute control object, see details in [FeatureControl](#).

3.3.4.9. get_payload_size

Statement: DataStream.get_payload_size

Significance: Get payloadsize.

Return value: Return payloadsize.

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.4.10. register_buffer

Statement: DataStream.register_buffer(user_buf, user_param)

Significance: The buffer applied by registered users is used for internal collection, and users can independently manage the buffer memory.

Parameter:

[in]	user_buf	The buffer pointer requested by the user
[in]	user_param	User defined parameters, which can be get from the RawImage.get_user_param()

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

Note:

- 1) The buffer size requested by the user cannot be calculated by itself and needs to be obtained from the [get_payload_size\(\)](#) interface, otherwise it may cause exceptions. Products such as POLS data frame grabber require alignment of memory size and first address for data acquisition based on performance considerations. Therefore, user needs to align the starting address and size according to the alignment byte requirements specified by the `StreamBufferAlignment` attribute when using this interface.
- 2) When the number of registered buffers is small, the acquisition driver completes a single acquisition (buffer is in the consumer queue), and no buffer available in the driver may result in frame loss. Therefore, it is recommended to register at least 3 buffers and complete the use of the image as soon as possible. In addition, hardware such as POLS requires a minimum `AnnounceBuffer` quantity of 3 or more. If the number of buffers registered by the user is insufficient, it will result in acquisition failure.
- 3) After calling the [register_buffer\(\)](#) interface, if the user modifies the parameters of the camera or frame grabber, resulting in a change in the `PayloadSize` output, the registered buffer needs to be deregistered, and memory needs to be reapplied and registered based on the new length.
- 4) [register_buffer\(\)](#) is an optional interface. If this interface is not called to register the buffer, the API library will automatically apply for and mine the buffer after mining, without affecting the collection process.

Sample code:

```
# Create a byte-aligned buffer
def create_aligned_buffer(size, alignment):
    buf = mmap.mmap(-1, size + alignment, access=mmap.ACCESS_WRITE)
    address = (ctypes.c_char * size).from_buffer(buf)
    return address

# Get the buffer size and alignment
payload_size = cam.data_stream[0].get_payload_size()
stream_feature_control = cam.data_stream[0].get_feature_control()
alignment=stream_feature_control.get_int_feature(
    "StreamBufferAlignment").get()

# Create the buffer and register it with the capture system.
buf1 = create_aligned_buffer(payload_size, alignment)
buf2 = create_aligned_buffer(payload_size, alignment)
buf3 = create_aligned_buffer(payload_size, alignment)
cam.data_stream[0].register_buffer(buf1, 1)
cam.data_stream[0].register_buffer(buf2, 3.14)
cam.data_stream[0].register_buffer(buf3, "test")

# Start acquisition
cam.stream_on()
num = 3
for i in range(num):
    # Open the data stream of channel 0
    raw_image = cam.data_stream[0].get_image()
    print('user_param: ', raw_image.get_user_param())

# Stop acquisition
cam.stream_off()
```

```
# Unregister the capture buffer.  
cam.data_stream[0].unregister_buffer(buf1)  
cam.data_stream[0].unregister_buffer(buf2)  
cam.data_stream[0].unregister_buffer(buf3)
```

3.3.4.11. unregister_buffer

Statement: `DataStream.unregister_buffer (user_buf)`

Significance: Unregister the buffer registered by the user using the [register_buffer\(\)](#) interface.

Parameter:

[in] user_buf The buffer pointer requested by the user

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

Note: This interface needs to be called after the collection is stopped. Calling it during the mining process will affect the normal collection process.

3.3.5. FeatureControl

Responsible for querying whether the access properties of each node are supported, readable, and writable, as well as setting and retrieving data.

Interface List:

is_implemented()	Check if the current function is supported
is_readable()	Check if the current function is readable
is_writable()	Check if the current function is writable
get_int_feature()	Get an integer data node object
get_enum_feature()	Get an enumeration data node object
get_float_feature()	Get a float type data node object
get_bool_feature()	Get a boolean data node object
get_string_feature()	Get a string data node object
get_command_feature()	Get a command data node object
get_register_feature()	Get a register data node object
feature_save()	Save user parameter group
feature_load()	Load user parameter groups
read_port()	Read the value of the specified register
write_port()	Write the value of the specified register
read_port_stacked()	Batch read specified register values
write_port_stacked()	Batch write specified register values

3.3.5.1. is_implemented

Statement: `is_implemented(feature_name)`

Significance: Check if the current function is supported.

Parameter: [in] feature_name Property string

Return value: Support returning true, otherwise return false.

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.2. is_readable

Statement: is_readable(feature_name)

Significance: Check if the current function is readable.

Parameter: [in] feature_name Property string

Return value: Readable returns true, otherwise returns false.

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.3. is_writable

Statement: is_writable(feature_name)

Significance: Check if the current function is writable.

Parameter: [in] feature_name Property string

Return value: Writable returns true, otherwise returns false.

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.4. get_int_feature

Statement: get_int_feature(feature_name)

Significance: Get an integer data node object.

Parameter: [in] feature_name Property string

Return value: Returns an integer data node object, see details in [IntFeature_s](#).

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.5. get_enum_feature

Statement: get_enum_feature (feature_name)

Significance: Get an enumeration type data node object.

Parameter: [in] feature_name Property string

Return value: Get an enumeration type data node object, see details in [EnumFeature_s](#).

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.6. get_float_feature

Statement: get_float_feature(feature_name)

Significance: Get a float data node object.

Parameter: [in] feature_name Property string

Return value: Return float data node object, see details in [EnumFeature_s](#).

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.7. get_bool_feature

Statement: get_bool_feature(feature_name)

Significance: Get a Boolean data node object.

Parameter: [in] feature_name Property string

Return value: Returns a boolean data node object, see details in [BoolFeature_s](#).

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.8. get_string_feature

Statement: get_string_feature(feature_name)

Significance: Get a string data node object.

Parameter: [in] feature_name Property string

Return value: Returns a string data node object, see details in [StringFeature_s](#).

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.9. get_command_feature

Statement: get_command_feature(feature_name)

Significance: Get a command type data node object, see details in [CommandFeature_s](#).

Parameter: [in] feature_name Property string

Return value: Returns a command type data node object.

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.10. get_register_feature

Statement: get_register_feature(feature_name)

Significance: Get a register type data node object, see details in [RegisterFeature_s](#).

Parameter: [in] feature_name Property string

Return value: Returns a register type data node object.

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.11. feature_save

Statement: feature_save(file_path)

Significance: Save the current device configuration parameters to a text file.

Parameter: [in] file_path Export file path name

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.12. feature_load

Statement: feature_load(file_path, verify=False)

Significance: Load the parameters in the configuration file onto the device.

Parameter: [in] file_path Load file path name

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.13. read_port

Statement: read_port(address, size)

Significance: Read the value of the camera's specified register.

Note: The attribute value of the gigabit network camera was read as big endian data.

Parameter:

[in] address Property register address

[in] size Byte size to read attribute

Return value: Returns the value of the specified register.

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.14. write_port

Statement: write_port(address, buff, size)

Significance: Set the value of the camera's specified register.

Note:

- 1) The attribute values set for GigE cameras need to be converted to big endian settings.
- 2) After calling the current interface, the value obtained by [get_enum_feature](#), [get_int_feature](#) Interface are still the the value before modification. Users can get the latest property value by [read_port](#) or [read_port_stacked](#).

Parameter:

[in]	address	Property register address
[out]	buff	The attribute cache to be written
[in]	size	The size of the attribute to be written

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

3.3.5.15. read_port_stacked

Statement: read_port_stacked(entries, size)

Significance: Batch reading of values from multiple registers specified by the camera, currently only supports integers (4 bytes).

Note: Obtain the attribute values of GigE cameras as big endian data.

Parameter:

[in]	entries	Pointer array to GxRegisterStackEntry type
[in]	size	Array size

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

Sample Code:

```
# Get Property Controller
remote_device_feature = cam.get_remote_device_feature_control()

# Number of structural elements
i = 2

# Requesting memory for the buffer
outbuffer = 64
buffer = c_void_p()
buffer.value = id(outbuffer)

outbuffer1 = 1024
buffer1 = c_void_p()
buffer1.value = id(outbuffer1)

# Define structure list
struct_list = [c_ulonglong(0x00900018), buffer, c_uint(4)]
struct_list1 = [c_ulonglong(0x00900008), buffer1, c_uint(4)]

# Structure initialization
struct_Reg = GxRegisterStackEntry(*struct_list)
struct_Reg1 = GxRegisterStackEntry(*struct_list1)

# Create a multi linked list
list = [struct_Reg, struct_Reg1]

# Define a structural array of type GxRegisterStackEntry with a length
```

```
# of 2
struct_array = GxRegisterStackEntry * 2

# Structure array initialization
array_instance = struct_array(*list)

# Convert to pointer batch read register
struct = pointer(array_instance)
status = remote_device_feature.read_port_stacked(struct, i)
assert status == 0

# Get data in memory
length = c_int
width_value = cast(array_instance[0].buffer,
                    POINTER(length)).contents.value
width_value1 = cast(array_instance[1].buffer,
                     POINTER(length)).contents.value
```

3.3.5.16. write_port_stacked

Statement: write_port_stacked(entries, size)

Significance: Batch setting of values for multiple registers specified by the camera, currently only supports integers (4 bytes).

Note:

- 1) The attribute values set for GigE cameras need to be converted to big endian settings.
- 2) After calling the current interface, the value obtained by [get_enum_feature\(\)](#), [get_bool_feature\(\)](#), GetBoolFeature Interface are still the the value before modification. Users can get the latest property value by [read_port\(\)](#) or [read_port_stacked\(\)](#).

Parameter:

[in]	entries	Pointer array to GxRegisterStackEntry type
[in]	size	Array size

Exception handling: If the function call fails, an exception is thrown. For details, see [Error handling](#).

Sample Code:

```
# Get Property Controller
remote_device_feature = cam.get_remote_device_feature_control()

# Number of structural elements
i = 2

size = 2
value = c_int(64)
point = byref(value)
bufer = cast(point, c_void_p)
buffer_c = c_void_p()
buffer_c = bufer

# Define structure list
```



```
struct_list = [c_ulonglong(0x00900008), buffer_c, c_uint(4)]
struct_list1 = [c_ulonglong(0x0090000C), buffer_c, c_uint(4)]

struct_Reg = GxRegisterStackEntry(*struct_list)
struct_Reg1 = GxRegisterStackEntry(*struct_list1)
list = [struct_Reg, struct_Reg1]

struct_array = GxRegisterStackEntry * 2
array_instance = struct_array(*list)

struct = pointer(array_instance)

# Batch write register
status = remote_device_feature.write_port_stacked(struct, size)
assert status == 0
assert status == 0
```

3.3.6. RGBImage (No longer maintained)

Recommend using ImageProcess.

It is responsible for the operation of RGB image.

Interface list:

image_improvement()	Improve image quality
brightness()	Image brightness adjustment
contrast()	Image contrast adjustment
saturation()	Image saturation adjustment
sharpen()	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

3.3.6.1. 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.

3.3.6.2. 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**.

3.3.6.3. 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**.

3.3.6.4. 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:** If the image saturation adjustment fails, throw the exception **UnexpectedError**.

3.3.6.5. 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:** If the image sharpening adjustment fails, throw the exception **UnexpectedError**.

3.3.6.6. get_white_balance_ratio

Statement: `RGBImage.get_white_balance_ratio()`**Significance:** Get the white balance ratio.**Return value:** RGB component coefficient tuple.

3.3.6.7. get_numpy_array

Statement: `RGBImage.get_numpy_array()`**Significance:** Get numpy object.**Return value:** Numpy object.

3.3.6.8. get_image_size

Statement: `RGBImage.get_image_size()`

Significance: Get the data size of RGB.

Return value: The size of RGB image.

3.3.7. RawImage

It is responsible for the operation of Raw image.

Interface list:

<code>defective_pixel_correct()</code>	Image defective pixel correction
<code>raw8_rotate_90_cw()</code>	8-bit image 90 degree clockwise rotate
<code>raw8_rotate_90_ccw()</code>	8-bit image 90 degree counterclockwise rotate
<code>mirror()</code>	Mirroring 8-bit image
<code>get_ffc_coefficients()</code>	Calculate Flat Field Correction (FFC) coefficients
<code>flat_field_correction()</code>	Execute FFC
<code>get_numpy_array()</code>	Convert raw data to numpy
<code>get_data()</code>	Get raw data
<code>save_raw()</code>	Save the data of raw image
<code>get_status()</code>	Get the status of raw image
<code>get_width()</code>	Get the width of raw image
<code>get_height()</code>	Get the height of raw image
<code>get_pixel_format()</code>	Get the pixel format of image
<code>get_image_size()</code>	Get the data size of raw image
<code>get_frame_id()</code>	Get the frame ID
<code>get_timestamp()</code>	Get the timestamp of raw image
<code>rgb8_to_numpy_array()</code>	Convert RGB8 data to numpy objects
<code>is_color_cam()</code>	Is it a color camera
<code>get_output_pixel_format()</code>	Get output pixel format
<code>get_chunkdata()</code>	Obtain frame information data
<code>get_user_param()</code>	Get user-defined parameters
<code>convert()</code>	Image format conversion (Stop maintenance)
<code>brightness()</code>	8-bit gray scale image brightness adjustment (Stop maintenance)
<code>contrast()</code>	8-bit gray scale image contrast adjustment (Stop maintenance)

3.3.7.1. defective_pixel_correct

Statement: RawImage.defective_pixel_correct()

Significance: Defective pixel correction on raw data.

Return value: None.

Exception handling: If the defective pixel correction is unsuccessful, throw the exception **UnexpectedError**.

3.3.7.2. 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: If the rotate is unsuccessful, throw the exception **UnexpectedError**.

3.3.7.3. 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: If the rotate is unsuccessful, throw the exception **UnexpectedError**.

3.3.7.4. 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**.

3.3.7.5. 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**.

3.3.7.6. 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**.

3.3.7.7. 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**.

3.3.7.8. get_data

Statement: RawImage.get_data()**Significance:** Get raw data.**Return value:** Raw data [string type]

3.3.7.9. 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**.

3.3.7.10. get_status

Statement: RawImage.get_status()**Significance:** Get the status of raw image.**Return value:** The status of raw image, data type reference [GxFrameStatusList](#).

3.3.7.11. get_width

Statement: RawImage.get_width()**Significance:** Get the width of raw image.**Return value:** The width of raw image.

3.3.7.12. get_height

Statement: RawImage.get_height()**Significance:** Get the height of raw image.**Return value:** The height of raw image.

3.3.7.13. get_pixel_format

Statement: RawImage.get_pixel_format()

Significance: Get the pixel format of image.

Return value: The pixel format.

3.3.7.14. get_image_size

Statement: RawImage.get_image_size()

Significance: Get the data size of raw image.

Return value: The size of raw image.

3.3.7.15. get_frame_id

Statement: RawImage.get_frame_id()

Significance: Get frame ID.

Return value: Frame ID

3.3.7.16. get_timestamp

Statement: RawImage.get_timestamp()

Significance: Get timestamp.

Return value: Timestamp.

3.3.7.17. rgb8_to_numpy_array

Statement: RawImage.rgb8_to_numpy_array()

Significance: Convert rgb8 data to numpy objects.

Return value:

numpyobject: Success

None: Fail

3.3.7.18. is_color_cam

Statement: RawImage.is_color_cam()

Significance: Is it a color camera.

Return value: Is it a color camera.

3.3.7.19. get_output_pixel_format

Statement: RawImage.get_output_pixel_format()

Significance: Get the output pixel format.

Return value: Output pixel format.

3.3.7.20. get_chunkdata

Statement: RawImage.get_chunkdata()

Significance: Obtain frame information data.

Return value: Frame information data.

3.3.7.21. get_user_param

Statement: RawImage.get_user_param()

Significance: Retrieve the user parameters set by the user through [register_buffer\(\)](#). If the interface is not called, return a value of null.

Return value: Return user-defined parameters.

3.3.7.22. convert (Stop maintenance)

Recommend using [ImageFormatConvert.convert\(\)](#) or [ImageFormatConvert.convert_ex\(\)](#).

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**.

3.3.7.23. brightness (Stop maintenance)

Recommend using [ImageProcessConfig.set_contrast_param\(\)](#) for setting, [ImageProcess.image_improvement\(\)](#) for processing.

Statement: RawImage.brightness(factor)

Significance: Brightness adjustment on 8-bit gray scale images.

Formal parameter: [in] factor	Brightness adjustment factor, range: -150~150 0: unchanged. >0: Brightness increase <0: Brightness decrease
---	--

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**.

3.3.7.24. contrast (Stop maintenance)

Recommend using [ImageProcessConfig.set_lightness_param\(\)](#) for setting and [ImageProcess.image_improvement\(\)](#) for processing.

Statement: RawImage.contrast(factor)

Significance: Contrast adjustment on 8-bit gray scale images.

Formal parameter: [in] factor Contrast adjustment factor, range: -50~150
0: unchanged.
>0: Contrast increase
<0: Contrast decrease

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**.

3.3.8. 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()	Get buffer object from a file
from_string()	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

3.3.8.1. 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.

3.3.8.2. 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.

3.3.8.3. 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.

3.3.8.4. 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]

3.3.8.5. 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]

3.3.8.6. 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.

3.3.9. ImageProcessConfig

Responsible for configuring image processing properties.

Interface List:

set_valid_bits()	Select to obtain the specified 8 valid data bits, and this interface sets which 8 bits are specified
get_valid_bits()	Query which 8 valid bits are currently specified
enable_defective_pixel_correct()	Enable defect pixel correct
is_defective_pixel_correct()	Check if bad point correction is currently enabled
enable_sharpen()	Enable sharpen
is_sharpen()	Query whether sharpening is currently enabled
set_sharpen_param()	Set sharpening intensity factor
get_sharpen_param()	Query the current sharpening intensity factor used
set_contrast_param()	Set contrast adjustment parameters
get_contrast_param()	Query the currently used contrast adjustment parameters
set_gamma_param()	Set Gamma coefficient
get_gamma_param()	Get Gamma coefficient
set_lightness_param()	Set brightness adjustment parameters
get_lightness_param()	Obtain brightness adjustment parameters
enable_denoise()	Enable denoise
is_denoise()	Query whether noise reduction is currently enabled
set_saturation_param()	Set saturation coefficient
get_saturation_param()	Get saturation coefficient
set_convert_type()	Set image format conversion algorithm
get_convert_type()	Get image format conversion algorithm
enable_convert_flip()	Enable image format conversion flipping
is_convert_flip()	Query whether image format conversion flipping is currently enabled
enable_accelerate()	Enable accelerated image processing
is_accelerate()	Check if the current work is in the acceleration enable state
enable_color_correction()	Enable color correction
is_color_correction()	Check if color correction is enabled
enable_user_set_ccparam()	Enable color correction in user settings mode
is_user_set_ccparam()	Check if color correction is in user set mode
set_user_ccparam()	Set color conversion factor structure
get_user_ccparam()	Get color conversion factor structure
reset()	Restore default tuning parameters

3.3.9.1. set_valid_bits

Statement: set_valid_bits(valid_bits)

Significance: Select to obtain the specified 8-bit valid data bits, this interface is set up for non 8-bit raw data.

Parameter: [in] param valid bit, default to the top 8 bits of the current pixel format, refer to [DxValidBit](#)

Exception handling: If the valid bit setting is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.2. get_valid_bits()

Statement: get_valid_bits()

Significance: Obtain a specified 8-bit valid data bit, this interface is set up for non 8-bit raw data.

Return value: Valid bits, see details in [DxValidBit](#).

3.3.9.3. enable_defective_pixel_correct

Statement: enable_defective_pixel_correct(enable)

Significance: Enable defect pixel correction function

Parameter: [in] enable Enable defect pixel correction function, true is enable, false is disable

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.4. is_defective_pixel_correct

Statement: is_defective_pixel_correct()

Significance: Obtain the defect pixel correction status.

Return value: Defect pixel correction status.

3.3.9.5. enable_sharpen

Statement: enable_sharpen(enable)

Significance: Enable sharpening

Parameter: [in] enable Enable sharpening, If bEnable is true, enable sharpening
If false, disable sharpening

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.6. is_sharpen

Statement: is_sharpen()

Significance: Check if sharpening is currently enabled.

Return value: Enable sharpening state.

3.3.9.7. set_sharpen_param

Statement: set_sharpen_param(param)

Significance: Set the sharpening intensity factor.

Parameter: [in] param Sharpening intensity factor.

Exception handling:

- 1) If the input parameter is not a floating-point type, throw the ParameterTypeError exception.
- 2) If the input parameter range is not between 0.1 and 5.0, an UnexpectedError exception is thrown.

3.3.9.8. get_sharpen_param

Statement: get_sharpen_param()

Significance: Queries the sharpening intensity factor currently in use.

Return value: Returns the sharpening intensity factor.

3.3.9.9. set_contrast_param

Statement: set_contrast_param(param)

Significance: Set contrast adjustment parameters.

Parameter: [in] param Contrast adjustment parameters.

Exception handling:

- 1) If the input parameter is not an integer, throw the ParameterTypeError exception.
- 2) If the input parameter range is not between -50 and 100, an UnexpectedError exception is thrown.

3.3.9.10. get_contrast_param

Statement: get_contrast_param()

Significance: Query the currently used contrast adjustment parameters.

Return value: Return the contrast adjustment parameter.

3.3.9.11. set_gamma_param

Statement: set_gamma_param(param)

Significance: Set the Gamma coefficient.

Parameter: [in] param Gamma coefficient

Exception handling:

- 1) If the input parameter is not an integer or float type, a `ParameterTypeError` exception is thrown.
- 2) `UnexpectedError` exception is thrown if the input parameter range is not -0.1~10.0.

3.3.9.12. `get_gamma_param`**Statement:** `get_gamma_param()`**Significance:** Obtain Gamma coefficient.**Return value:** Return Gamma coefficient.3.3.9.13. `set_lightness_param`**Statement:** `set_lightness_param(param)`**Significance:** Set the brightness adjustment parameters.**Parameter:** [in] param Brightness Adjustment Parameters**Exception handling:**

- 1) If the input parameter is not an integer, a `ParameterTypeError` exception is thrown.
- 2) If the input parameter range is not -150~150, then throw `UnexpectedError` exception.

3.3.9.14. `get_lightness_param`**Statement:** `get_lightness_param()`**Significance:** Obtain brightness adjustment parameters.**Return value:** Return brightness adjustment parameters.3.3.9.15. `enable_denoise`**Statement:** `enable_denoise(enable)`**Significance:** Enable noise reduction.**Parameter:** [in] param Enable noise reduction switch (not supported by mono cameras).
true means enable noise reduction, false to disable noise reduction**Exception handling:** If the function call is unsuccessful, an exception is thrown, see [Error handling](#).3.3.9.16. `is_denoise()`**Statement:** `is_denoise()`**Significance:** Get noise reduction switch (not supported for mono cameras).**Return value:** Returns the noise reduction switch flag.

3.3.9.17. set_saturation_param

Statement: set_saturation_param(param)

Significance: Set saturation adjustment factor (not supported for mono cameras).

Parameter: [in] param Saturation adjustment factor.

Exception handling:

- 1) If the input parameter is not an integer or floating-point type, a `ParameterTypeError` exception is thrown.
- 2) `UnexpectedError` exception is thrown if the input parameter range is not 0~128.

3.3.9.18. get_saturation_param

Statement: get_saturation_param()

Significance: Obtain saturation adjustment coefficient (not supported by mono cameras).

Return value: Returns the saturation adjustment factor.

3.3.9.19. set_convert_type

Statement: set_convert_type(cv_type)

Significance: Set image interpolation processing algorithm (not supported for mono cameras).

Parameter: [in] cv_type Convert type, the default value is `DxBayerConvertType.NEIGHBOUR`, refer to [DxBayerConvertType](#)

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.20. get_convert_type

Statement: get_convert_type()

Significance: Obtain image interpolation processing algorithm (not supported for mono cameras).

Return value: Obtain image interpolation processing algorithm (not supported for mono cameras), see details in [DxBayerConvertType](#).

3.3.9.21. enable_convert_flip

Statement: enable_convert_flip(flip)

Significance: Enable interpolation flipping (not supported for mono cameras).

Parameter: [in] flip When flip is true enables flipping, false disables flipping.

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.22. is_convert_flip

Statement: is_convert_flip()

Significance: Get the interpolated flip flag (not supported by mono cameras), true means flip, false means no flip.

Return value: Returns the interpolated flip flop identifier.

3.3.9.23. enable_accelerate

Statement: enable_accelerate(accelerate)

Significance: Image processing acceleration enable. Set ture to enable acceleration, false to disable acceleration.

Parameter: [in] accelerate Set ture to enable acceleration, false to disable acceleration

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.24. is_accelerate

Statement: is_accelerate()

Significance: Queries whether it is currently working in acceleration enable state, true means acceleration, false means no acceleration.

Return value: Acceleration enable state.

3.3.9.25. enable_color_correction

Statement: enable_color_correction(enable)

Significance: Enable colour correction (not supported by mono cameras), true means enable colour correction, false means disable colour correction.

Parameter: [in] enable rue means enable colour correction, false means disable colour correction

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.26. is_color_correction

Statement: is_color_correction()

Significance: Queries whether to enable colour correction (not supported by mono cameras).

Return value: Enable the colour correction state.

3.3.9.27. enable_user_set_ccparam

Statement: enable_user_set_ccparam(enable)

Significance: Enable colour correction user setting mode (not supported by mono cameras), true means user mode, false means non-user mode.

Parameter: [in] enable enable is true to enable the colour correction user setting mode, false to disable the colour correction user setting mode function

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.9.28. is_user_set_ccparam

Statement: is_user_set_ccparam()

Significance: Queries whether the colour correction is user-set mode (not supported by mono cameras), true means user mode, false means non-user mode.

Return value: Calibrate whether or not the user sets mode.

3.3.9.29. set_user_ccparam

Statement: set_user_ccparam(color_transform_factor)

Significance: Set the colour conversion factor structure (the suggested range of parameters for the colour correction structure is -4~4, if the parameter setting is less than -4, the correction effect is consistent with -4, if the parameter setting is greater than 4, the correction effect is consistent with 4).

Parameter: [in] color_transform_factor Sets the colour conversion factor structure.

Exception handling: If the input parameter is not of type [ColorTransformFactor](#), a ParameterTypeError exception is thrown.

3.3.9.30. get_user_ccparam

Statement: get_user_ccparam()

Significance: Gets the colour conversion factor structure.

Return value: Conversion Factor Structures.

3.3.9.31. reset

Statement: reset()

Significance: Restore the optimal default parameter configuration.

3.3.10. Utility

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

Interface list:

get_gamma_lut()	Get gamma LUT by gamma value
get_contrast_lut()	Get contrast LUT by the contrast value
get_lut()	Calculate 8-bit LUT for image processing
calc_cc_param()	Calculate color correction parameter array for image processing

[calc_user_set_cc_param\(\)](#) Caculate color correction parameter array by user set for image processing

3.3.10.1. 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**.

3.3.10.2. 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**.

3.3.10.3. 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**.

3.3.10.4. 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**.

3.3.10.5. calc_user_set_cc_param (Static function)

Statement: Utility.calc_user_set_cc_param(color_transform_factor, saturation=64)

Significance: Caculate 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**.

3.3.11. ImageProcess

Enhance the current image and return the enhanced image data.

Note:

- 1) If [ImageProcessConfig.is_accelerate\(\)](#) returns true, it means that the image will be processed in an accelerated way, the accelerated way of processing the image requires that the height of the image must be an integer multiple of 4, otherwise, the interface reports an error.
- 2) The black and white camera void* returns 8bit image data, the image size is image width x image height.
- 3) The colour camera void* returns RGB image data with an image size of image width x image height x 3.

Interface List:

image_improvement()	Image quality enhancement processing
static_defect_correction()	Image static defect pixel correction
calcula_lut()	Calculate LUT
read_lut_file()	Read LUT file

3.3.11.1. image_improvement

Statement: image_improvement(image, output_address, image_process_config)

Significance: Do image quality enhancement for the input image, Note: The image quality enhancement of Bayer format image is RGB format after processing.

Parameter:

[in]	image	The source image (variable type), of type RawImage, available via DataStream.get_image() . see details in GxImageInfo
[out]	output_address	Returns the image Buffer after image quality enhancement
[in]	image_process_config	Image Quality Enhancement Auxiliary Configuration Object to control information such as image sharpening and brightness.

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.11.2. static_defect_correction

Statement: static_defect_correction(input_address, output_address,
defect_correction, defect_pos_buffer_address,
defect_pos_buffer_size)

Significance: Do static defect pixel correction for the input image.

Parameter:

[in]	input_address	Cache address of the original image to be corrected
[out]	output_address	Stores the output image cache address
[in]	defect_correction	Image static defect pixel correction parameters

[in]	defect_pos_buffer_address	Detecting static defect pixels file buffer
[in]	defect_pos_buffer_size	Detect static defect pixel file buffer location

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.11.3. calcula_lut

Statement: calcula_lut(contrast_param, gamma, light_ness, lut_address, lut_length_address)

Significance: Calculate LUT

Parameter:

[in]	contrast_param	contrast parameter
[in]	gamma	gamma parameter
[in]	light_ness	Brightness
[out]	lut_address	LUT address
[out]	lut_length_address	LUT length address

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.11.4. read_lut_file

Statement: read_lut_file(lut_file_path, lut_address, lut_length_address)

Significance: Reads LUT from file to program memory.

Parameter:

[in]	lut_file_path	Find LUT file path
[out]	lut_address	Address of the LUT to be stored
[out]	lut_length_address	LUT length

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12. ImageFormatConvert

Responsible for image format conversion, which can be used to convert image formats.

Interface List:

set_dest_format()	Set the target pixel format
get_dest_format()	Get target pixel format
set_interpolation_type()	Set conversion algorithm
get_interpolation_type()	Get conversion algorithm
set_alpha_value()	Set alpha channel
get_alpha_value()	Get alpha channel
set_valid_bits()	Set the significant bits to convert pixel grids
get_valid_bits()	Get the significant bits to convert pixel grids
get_buffer_size_for_conversion_ex()	Obtain the size of the buffer corresponding to converting pixel formats

get_buffer_size_for_conversion()	Obtain the size of the buffer corresponding to converting pixel formats
convert_ex()	Convert pixel format
convert()	Convert pixel format
set_3d_calib_param()	This function is used to load 3D calibration parameters and convert the height map into a point cloud map
get_3d_calib_param()	Obtain 3D calibration parameters
set_y_step(y_step)	Set the Y direction step size for converting height maps to point clouds
get_y_step()	Get the Y direction step size for converting height maps to point clouds

3.3.12.1. set_dest_format

Statement: set_dest_format(dest_pixel_format)

Significance: Set the expected conversion format, see details in [GxPixelFormatEntry](#).

Parameter: [in] dest_pixel_format The expected conversion format, see details in [GxPixelFormatEntry](#)

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.2. get_dest_format

Statement: get_dest_format()

Significance: Obtain the expected conversion format.

Return value: Return the expected conversion format, see details in [GxPixelFormatEntry](#).

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.3. set_interpolation_type

Statement: set_interpolation_type(cvt_type)

Significance: Set image format conversion algorithm.

Parameter: [in] cvt_type Conversion algorithm, see details in [DxBayerConvertType](#)
Default value: [DxBayerConvertType.NEIGHBOUR](#)

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.4. get_interpolation_type

Statement: get_interpolation_type()

Significance: Obtain image format conversion algorithm.

Return value: Return image format conversion algorithm, see details in [DxBayerConvertType](#).

3.3.12.5. set_alpha_value

Statement: set_alpha_value(alpha_value)

Significance: Set the alpha value of the image with an alpha channel.

Parameter: [in] alpha_value Alpha channel value, ranging from 0 to 255, with a default value of 255.

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.6. get_alpha_value

Statement: get_alpha_value()

Significance: Obtain image format conversion algorithm.

Return value: Get the Alpha channel value.

3.3.12.7. set_valid_bits

Statement: set_valid_bits(valid_bits)

Significance: Set the conversion significant bit.

Parameter: [in] valid_bits Valid bits, The default value is the top 8 bits of the current pixel format, refer to [DxValidBit](#)

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.8. get_valid_bits

Statement: get_valid_bits()

Significance: Obtain the conversion significant bit.

Return value: Valid bits, see details in [DxValidBit](#).

3.3.12.9. get_buffer_size_for_conversion_ex

Statement: get_buffer_size_for_conversion_ex(width, height, pixel_format)

Significance: Obtain the expected format of the image buffer length based on the input image data.

Parameter:

[in]	width	Source image width
[in]	height	Source image height
[in]	pixel_format	Source image format

Return value: The expected length of the image buffer in the format.

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.10. get_buffer_size_for_conversion

Statement: get_buffer_size_for_conversion(raw_image)

Significance: Obtain the expected format of the image buffer length based on the input image pointer.

Parameter: [in] raw_image Source image, obtain through DataStream.get_image()

Return value: The expected length of the image buffer in the format.

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.11. convert_ex

Statement: convert_ex(input_address, input_width,
input_height, src_fixel_format,
output_address, output_length, flip)

Significance: The expected length of the image buffer in the format. see details in [GxPixelFormatEntry](#).

Parameter:

[in]	input_address	Source image buffer pointer
[in]	input_width	Source image width
[in]	input_height	Source image height
[in]	src_fixel_format	Source image format
[out]	output_address	The destination image Buffer pointer, which is requested by the user, and the length is: output_length
[in]	output_length	Target image buffer length, which can be get through get_buffer_size_for_conversion_ex()
[in]	flip	Is the image flipped (true flipped, false not flipped)

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.12. convert

Statement: convert(raw_image, output_address, output_length, flip)

Significance: Convert the input source image to the desired image format, refer to [GxPixelFormatEntry](#).

Parameter:

[in]	raw_image	Source image, which can be obtained by DataStream.get_image()
[in]	output_address	The destination image Buffer pointer, which is requested by the user, and the length is: output_length
[in]	output_length	Target image buffer length, which can be obtained by get_buffer_size_for_conversion()
[in]	flip	Is the image flipped (true flipped, false not flipped)

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.13. set_3d_calib_param

Statement: set_3d_calib_param(self, buffer_address, buffer_size)

Significance: This function is used to load 3D calibration parameters and convert the height map into a point cloud map.

Parameter:

[in]	pCalibParamBuffer	3D calibration parameter memory
[in]	nBufferSize	3D calibration parameter memory length

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.14. get_3d_calib_param

Statement: get_3d_calib_param()

Significance: Obtain 3D calibration parameters.

Parameter: None.

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.15. set_y_step

Statement: set_y_step(y_step)

Significance: Set the Y direction step size for converting the height map to a point cloud.

Parameter: [in] dY Step size in Y direction

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.3.12.16. get_y_step

Statement: get_y_step()

Significance: Obtain the Y direction step size of the height transition point cloud.

Parameter: None

Exception handling: If the function call is unsuccessful, an exception is thrown, see [Error handling](#).

3.4. Feature Parameter (No longer maintaining)

Recommend using [Feature_s](#).

3.4.1. Device feature parameter

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

DeviceTemperatureSelector	Device temperature selection, see GxDeviceTemperatureSelectorEntry for details	EnumFeature
DeviceTemperature	Device temperature	FloatFeature
ImageFormat Section		
SensorWidth	Sensor width	IntFeature
SensorHeight	Sensor height	IntFeature
WidthMax	Maximum width	IntFeature
HeightMax	Maximum height	IntFeature
OffsetX	Horizontal offset	IntFeature
OffsetY	Vertical offset	IntFeature
Width	Image width	IntFeature
Height	Image height	IntFeature
BinningHorizontal	Horizontal Binning	IntFeature
BinningVertical	Vertical Binning	IntFeature
DecimationHorizontal	Horizontal decimation	IntFeature
DecimationVertical	Vertical decimation	IntFeature
PixelSize	Pixel size, see GxPixelSizeEntry for details	EnumFeature
PixelColorFilter	Bayer format, see GxPixelColorFilterEntry for details	EnumFeature
PixelFormat	Pixel format, see GxPixelFormatEntry for details	EnumFeature
ReverseX	Horizontal reverse	BoolFeature
ReverseY	Vertical reverse	BoolFeature
TestPattern	Test pattern, see GxTestPatternEntry for details	EnumFeature
TestPatternGeneratorSelector	Test pattern generator selector, see C Software Development Manual and GxTestPatternGeneratorSelectorEntry for details	EnumFeature

RegionSendMode	ROI send mode, see GxRegionSendModeEntry for details	EnumFeature
RegionMode	Region mode, see GxSwitchEntry for details	EnumFeature
RegionSelector	Region selector, see C Software Development Manual and GxRegionSelectorEntry for details	EnumFeature
CenterWidth	Center width	IntFeature
CenterHeight	Center height	IntFeature
BinningHorizontalMode	Horizontal Binning mode, see GxBinningHorizontalModeEntry for details	EnumFeature
BinningVerticalMode	Vertical Binning mode, see GxBinningVerticalModeEntry for details	EnumFeature
SensorShutterMode	Sensor shutter mode, see GxSensorShutterModeEntry for details	EnumFeature
TransportLayer Section		
PayloadSize	Payload size	IntFeature
CenterWidth	Center width	IntFeature
CenterHeight	Center height	IntFeature
GevCurrentIPConfigurationLLA	Configuring IP in LLA mode	BoolFeature
GevCurrentIPConfigurationDHCP	Configuring IP in DHCP mode	BoolFeature
GevCurrentIPConfigurationPersistentIP	Configuring IP in permanent IP mode	BoolFeature
EstimatedBandwidth	Estimated bandwidth	IntFeature
GevHeartbeatTimeout	Heartbeat timeout time	IntFeature
GevSCPSPacketSize	Stream channel packet size	IntFeature
GevSCPD	Stream channel packet delay	IntFeature
GevLinkSpeed	Link speed	IntFeature
DigitalIO Section		
UserOutputSelector	User-defined output selector, see C	EnumFeature

	Software Development Manual and GxUserOutputSelectorEntry for details	
UserOutputValue	User-defined output value	BoolFeature
UserOutputMode	User IO output mode, see GxUserOutputModeEntry for details	EnumFeature
StrobeSwitch	Strobe switch, see GxSwitchEntry for details	EnumFeature
LineSelector	Line selector, see C Software Development Manual and GxLineSelectorEntry for details	EnumFeature
LineMode	Line mode, see GxLineModeEntry for details	EnumFeature
LineSource	Line output source, see GxLineSourceEntry for details	EnumFeature
LineInverter	Line level inversion	BoolFeature
LineStatus	Line status	BoolFeature
LineStatusAll	Status of all lines	IntFeature
AnalogControls Section		
GainAuto	Automatic gain, see GxAutoEntry for details	EnumFeature
GainSelector	Gain channel selector, see C Software Development Manual and GxGainSelectorEntry for details	EnumFeature
BlackLevelAuto	Automatic black level, see GxAutoEntry for details	EnumFeature
BlackLevelSelector	Black level channel selector, see C Software Development Manual and GxBlackLevelSelectEntry for details	EnumFeature
BalanceWhiteAuto	Automatic white balance, see GxAutoEntry for details	EnumFeature
BalanceRatioSelector	White balance channel selector, see C Software Development Manual and GxBalanceRatioSelectorEntry for details	EnumFeature
BalanceRatio	White balance ratio	FloatFeature
DeadPixelCorrect	Defective pixel correction, see GxSwitchEntry for details	EnumFeature
Gain	Gain	FloatFeature

BlackLevel	Black level	FloatFeature
GammaEnable	Gamma enable	BoolFeature
GammaMode	Gamma mode, see GxGammaModeEntry for details	EnumFeature
Gamma	Gamma	FloatFeature
DigitalShift	Digital shift	IntFeature
LightSourcePreset	Light source preset, see GxLightSourcePresetEntry for details	EnumFeature
CustomFeature Section		
ADCLevel	AD conversion level	IntFeature
HBlanking	Horizontal blanking	IntFeature
VBlanking	Vertical blanking	IntFeature
UserPassword	User encryption zone password	StringFeature
VerifyPassword	User encryption zone verify password	StringFeature
UserData	User encryption zone content	BufferFeature
ExpectedGrayValue	Expected gray value	IntFeature
AALightEnvironment	Auto exposure, auto gain, lighting environment type, see GxAALightEnvironmentEntry for details	EnumFeature
ImageGrayRaiseSwitch	Image gray raise switch, see GxSwitchEntry for details	EnumFeature
AAROIOffsetX	Auto adjust the X offset of ROI	IntFeature
AAROIOffsetY	Auto adjust the Y offset of ROI	IntFeature
AAROIWidth	Auto adjust the width of ROI	IntFeature
AAROIHeight	Auto adjust the height of ROI	IntFeature
AutoGainMin	Minimum automatic gain	FloatFeature
AutoGainMax	Maximum automatic gain	FloatFeature

AutoExposureTimeMin	Minimum automatic exposure	FloatFeature
AutoExposureTimeMax	Maximum automatic exposure	FloatFeature
ContrastParam	Contrast parameter	IntFeature
ColorCorrectionParam	Color correction parameter	IntFeature
AWBROIOffsetX	The X coordinate of automatic white balance ROI	IntFeature
AWBROIOffsetY	The Y coordinate of automatic white balance ROI	IntFeature
AWBROIWidth	The width of automatic white balance ROI	IntFeature
AWBROIHeight	The height of automatic white balance ROI	IntFeature
GammaParam	Gamma parameter	FloatFeature
AWBLampHouse	Automatic white balance lamp house, see GxAWBLampHouseEntry for details	EnumFeature
SharpnessMode	Sharpening model, see GxSwitchEntry for details	EnumFeature
Sharpness	Sharpness	FloatFeature
FrameInformation	Image frame information	BufferFeature
DataFieldSelector	User selects the flash data field, see GxUserDataFieldSelectorEntry for details	EnumFeature
DataFieldValue	User data field content	BufferFeature
FlatFieldCorrection	Flat field correction switch, see GxSwitchEntry for details	EnumFeature
NoiseReductionMode	Noise reduction switch, see GxSwitchEntry for details	EnumFeature
NoiseReduction	Noise reduction value	FloatFeature
FFCLoad	Get flat field correction parameters	BufferFeature
FFCSave	Set flat field correction parameters	BufferFeature
StaticDefectCorrection	Static dead pixel correction switch, see GxSwitchEntry for details	EnumFeature

UserSetControl Section		
UserSetLoad	Load the user set	CommandFeature
UserSetSave	Save the user set	CommandFeature
UserSetSelector	User set selector, see C Software Development Manual and GxUserSetEntry for details	EnumFeature
UserSetDefault	Default the user set, see GxUserSetEntry for details	EnumFeature
Event Section		
EventSelector	Event source select, see GxEventSelectorEntry for details	EnumFeature
EventNotification	Switch of the notification to the host application of the occurrence of the selected Event, see GxSwitchEntry for details	EnumFeature
EventExposureEnd	Exposure end event	IntFeature
EventExposureEndTimestamp	The timestamp of Exposure end event	IntFeature
EventExposureEndFrameID	The frame id of Exposure end event	IntFeature
EventBlockDiscard	Block discard event	IntFeature
EventBlockDiscardTimestamp	The timestamp of Block discard event	IntFeature
EventOverrun	Event queue overflow event	IntFeature
EventOverrunTimestamp	The timestamp of event queue overflow event	IntFeature
EventFrameStartOvertrigger	Trigger signal shield event	IntFeature
EventFrameStartOvertriggerTimestamp	The timestamp of trigger signal shield event	IntFeature
EventBlockNotEmpty	Frame memory not empty event	IntFeature
EventBlockNotEmptyTimestamp	The timestamp of frame memory not empty event	IntFeature
EventInternalError	Internal erroneous event	IntFeature
EventInternalErrorTimestamp	The timestamp of internal erroneous event	IntFeature

EventFrameBurstStartOvertrigger	Frame burst start overtrigger event ID	IntFeature
EventFrameBurstStartOvertriggerFrame ID	Frame burst start overtrigger event frame ID	IntFeature
EventFrameBurstStartOvertriggerTimes tamp	Frame burst start overtrigger event timestamp	IntFeature
EventFrameStartWait	Frame start wait event ID	IntFeature
EventFrameStartWaitTimestamp	Frame start wait event timestamp	IntFeature
EventFrameBurstStartWait	Frame burst start wait event ID	IntFeature
EventFrameBurstStartWaitTimestamp	Frame burst start wait event timestamp	IntFeature
EventBlockDiscardFrameID	Data block discard event frame ID	IntFeature
EventFrameStartOvertriggerFrameID	Frame start wait overtrigger event frame ID	IntFeature
EventBlockNotEmptyFrameID	Data block not empty event frame ID	IntFeature
EventFrameStartWaitFrameID	Frame start wait event frame ID	IntFeature
EventFrameBurstStartWaitFrameID	Frame burst start wait event frame ID	IntFeature
LUT Section		
LUTValueAll	LUT content	BufferFeature
LUTSelector	LUT selector, see C Software Development Manual and GxLutSelectorEntry for details	EnumFeature
LUTEnable	LUT enable	BoolFeature
LUTIndex	LUT index	IntFeature
LUTValue	LUT value	IntFeature
Color Transformation Control		
ColorTransformationMode	Color transformation mode, see GxColorTransformationModeEntry for details	EnumFeature
ColorTransformationEnable	Color transformation enable	BoolFeature
ColorTransformationValueSelector	Color transformation matrix value selector, see GxColorTransformationValueSelectorE	EnumFeature

	ntry for details	
ColorTransformationValue	Color transformation matrix value	FloatFeature
SaturationMode	Saturation switch, see GxSwitchEntry for details	EnumFeature
Saturation	Saturation value	IntFeature
ChunkData Section		
ChunkModeActive	Chunk data enable	BoolFeature
ChunkEnable	Single chunk data enable	BoolFeature
ChunkSelector	Chunk data selector, see C Software Development Manual and GxChunkSelectorEntry for details	EnumFeature
Device Feature		
DeviceCommandTimeout	Device command timeout	IntFeature
DeviceCommandRetryCount	Device command retry count	IntFeature
AcquisitionTrigger Section		
FrameBufferOverwriteActive	Frame buffer overwrite enable	BoolFeature
AcquisitionStart	Start acquisition	CommandFeature
AcquisitionStop	Stop acquisition	CommandFeature
TriggerSoftware	Software trigger	CommandFeature
TransferStart	Start transfer	CommandFeature
AcquisitionMode	Acquisition mode, see GxAcquisitionModeEntry for details	EnumFeature
TriggerMode	Trigger mode, see GxSwitchEntry for details	EnumFeature
TriggerActivation	Trigger activation, see GxTriggerActivationEntry for details	EnumFeature
ExposureAuto	Auto exposure, see GxAutoEntry for details	EnumFeature
TriggerSource	Trigger source, see GxTriggerSourceEntry for details	EnumFeature

ExposureMode	Exposure mode, see GxExposureModeEntry for details	EnumFeature
TriggerSelector	Trigger type selector, see C Software Development Manual and GxTriggerSelectorEntry for details.	EnumFeature
TransferControlMode	Transfer control mode, see GxTransferControlModeEntry for details	EnumFeature
TransferOperationMode	Transfer operation mode, see GxTransferOperationModeEntry for details	EnumFeature
AcquisitionFrameRateMode	Acquisition frame rate adjustment mode, see GxSwitchEntry for details	EnumFeature
FixedPatternNoiseCorrectMode	Fixed pattern noise correction mode, see GxSwitchEntry for details	EnumFeature
ExposureTime	Exposure time	FloatFeature
TriggerFilterRaisingEdge	Raising edge trigger filter	FloatFeature
TriggerFilterFallingEdge	Falling edge trigger filter	FloatFeature
TriggerDelay	Trigger delay	FloatFeature
AcquisitionFrameRate	Acquisition frame rate	FloatFeature
CurrentAcquisitionFrameRate	Current acquisition frame rate	FloatFeature
TransferBlockCount	Transfer block count	IntFeature
TriggerSwitch	External trigger switch, see GxSwitchEntry for details	EnumFeature
AcquisitionSpeedLevel	Acquisition speed level	IntFeature
AcquisitionFrameCount	Acquisition frame count	IntFeature
AcquisitionBurstFrameCount	Acquisition burst frame count	IntFeature
AcquisitionStatusSelector	Acquisition status selector, see C Software Development Manual and GxAcquisitionStatusSelectorEntry for details	EnumFeature
AcquisitionStatus	Acquisition status	BoolFeature
ExposureDelay	Exposure delay	FloatFeature

ExposureOverlapTimeMax	Maximum exposure overlap time	FloatFeature
ExposureTimeMode	Exposure time mode, see GxExposureTimeModeEntry for details	EnumFeature
CounterAndTimerControl Section		
TimerSelector	Selects which Counter to configure, see GxTimerSelectorEntry for details	EnumFeature
TimerDuration	Sets the duration (in microseconds) of the Timer pulse	FloatFeature
TimerDelay	Sets the duration (in microseconds) of the delay to apply at the reception of a trigger before starting the Timer	FloatFeature
TimerTriggerSource	Selects the source of the trigger to start the Timer, see GxTimerTriggerSourceEntry for details	EnumFeature
CounterSelector	Selects which Counter to configure, see GxCounterSelectorEntry for details	EnumFeature
CounterEventSource	Selects the events that will be the source to increment the Counter, see GxCounterEventSourceEntry for details	EnumFeature
CounterResetSource	Selects the signals that will be the source to reset the Counter, see GxCounterResetSourceEntry for details	EnumFeature
CounterResetActivation	Selects the Activation mode of the Counter Reset Source signal, see GxCounterResetActivationEntry for details	EnumFeature
CounterReset	Does a software reset of the selected Counter and starts it	CommandFeature
CounterTriggerSource	Counter trigger source, see GxCounterTriggerSourceEntry for details	EnumFeature
CounterDuration	Counter duration value	IntFeature
TimerTriggerActivation	Timer Trigger Activation, see GxTimerTriggerActivationEntry for details	EnumFeature
RemoveParameterLimitControl Section		
RemoveParameterLimit	Remove parameter range restriction switch, see GxSwitchEntry for details	EnumFeature

3.4.2. Stream feature parameter

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

StopAcquisitionMode	Stop acquisition mode, see GxStopAcquisitionModeEntry for details	EnumFeature
StreamBufferHandlingMode	Buffer handling mode see GxDSSStreamBufferHandlingModeEntry for details	EnumFeature

4. FAQ

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

5. Revision History

No.	Version	API Version	Changes	Date
1	V1.0.0	1.0.1808.x or above	Initial release	2018-08-10
2	V1.0.1	1.0.1810.x or above	Add the new function description for the Mercury2 cameras	2018-10-29
3	V1.0.2	1.0.1904.x or above	Modify some subtitles and inaccurate descriptions	2019-04-16
4	V1.0.3	1.0.1905.x or above	Add some descriptions	2019-05-07
5	V1.0.4	1.0.2103.x or above	Add offline callback and capture callback register, unregister function, Synchronize new feature ID and corresponding Entries	2021-03-04
6	V1.0.5	1.0.2103.x or above	Modify some description issues	2021-03-08
7	V1.0.6	1.0.2106.x or above	Add reconnect device and brightness adjust, mirror and other image processing library interfaces	2021-06-03
8	V1.0.7	2.0.2304.x or above	<ol style="list-style-type: none">1. Modify 2.2.2 section, update enumeration interface2. Modify 2.2.5 section, update image format conversion interface and image quality improvement interface3. Modify 2.2.6 section, update attribute read/write queries to new interface4. Modify 2.2.7 section, update import and export camera configuration to a new interface5. Add 3.1.1 section, basic data type reading and writing instructions6. Modify 3.2.1 section, add CXP type7. Modify 3.2.5 section, add pixel format of MONO10P, MONO12P, R8, B8, G8, RGB8, BGR8. Expand GxPixelFormatEntry definitions8. Modify 3.2.100 section, add valid bits9. Add 3.2.103 section, TL type description10. Add 3.2.104 section, explanation of image processing auxiliary structure11. Add 3.2.105 section, explanation of IP configuration mode structure12. Modify 3.3.1 section, add get_interface_number, get_interface_info, get_interface, gige_force_ip, gige_ip_configuration, create_image_format_convert, create_image_process interfaces	2024-04-30

			13. Modify 3.3.2 section, add get_stream_channel_num, get_stream, get_local_device_feature_control, get_remote_device_feature_control, register_device_feature_callback, register_device_feature_callback_by_string, unregister_device_feature_callback, unregister_device_feature_callback_by_string, read_remote_device_port, write_remote_device_port, read_remote_device_port_stacked, write_remote_device_port_stacked, create_image_process_config Interfaces 14. Add 3.3.3 section, Interface related instructions 15. Modify 3.3.4 section, add get_featrue_contro, get_payload_size, set_payload_size interfaces 16. Add 3.3.5 section, query the relevant descriptions of reading and writing the properties of each node 17. Add 3.3.9, 3.3.11 section, image processing related instructions 18. Add 3.3.12 section, explanation of image format conversion	
9	V1.0.8	2.0.2406.x or above	1. Add GxLogTypeList data type definition 2. Add 3.3.1.1 and 3.3.1.2 section	2024-06-12
10	V1.0.9	2.0.2407.x or above	1. Add 3.3.4.3 section, dq_buf Interface 2. Add 3.3.4.4 section, q_buf Interface	2024-07-12
11	V1.0.10	2.0.2412.x or above	1. Add 3.2.106.GxActionCommandResult 2. Add 3.3.1.21.issue_action_command 3.3.1.22.issue_scheduled_action_command	2024-12-10
12	V1.0.11	2.0.2412.x or above	1. Proofreading and revising section 3.2 for missing interface and structure definitions 2. Add 3.2.107.GxRegisterStackEntry 3. Add 3.2.108.ColorTransformFactor 4. Add 3.3.4.10.register_buffer 5. Add 3.3.4.11.unregister_buffer 6. Add 3.3.7.21.get_user_param 7. Optimize section description information	2025-01-08