中国大恒（集团）有限公司北京图像视觉技术分公司

Python 接口开发说明书

版本：V1.0.3
发布日期：2019-05-07

DAHENG IMAGING | 大恒图像

本手册中所提及的其它软硬件产品的商标与名称，都属于相应公司所有。

本手册的内容若有任何修改，恕不另行通知。

网　　站：http://www.daheng-imaging.com

销售信箱：sales@daheng-imaging.com

销售热线：010-82828878 转 8068

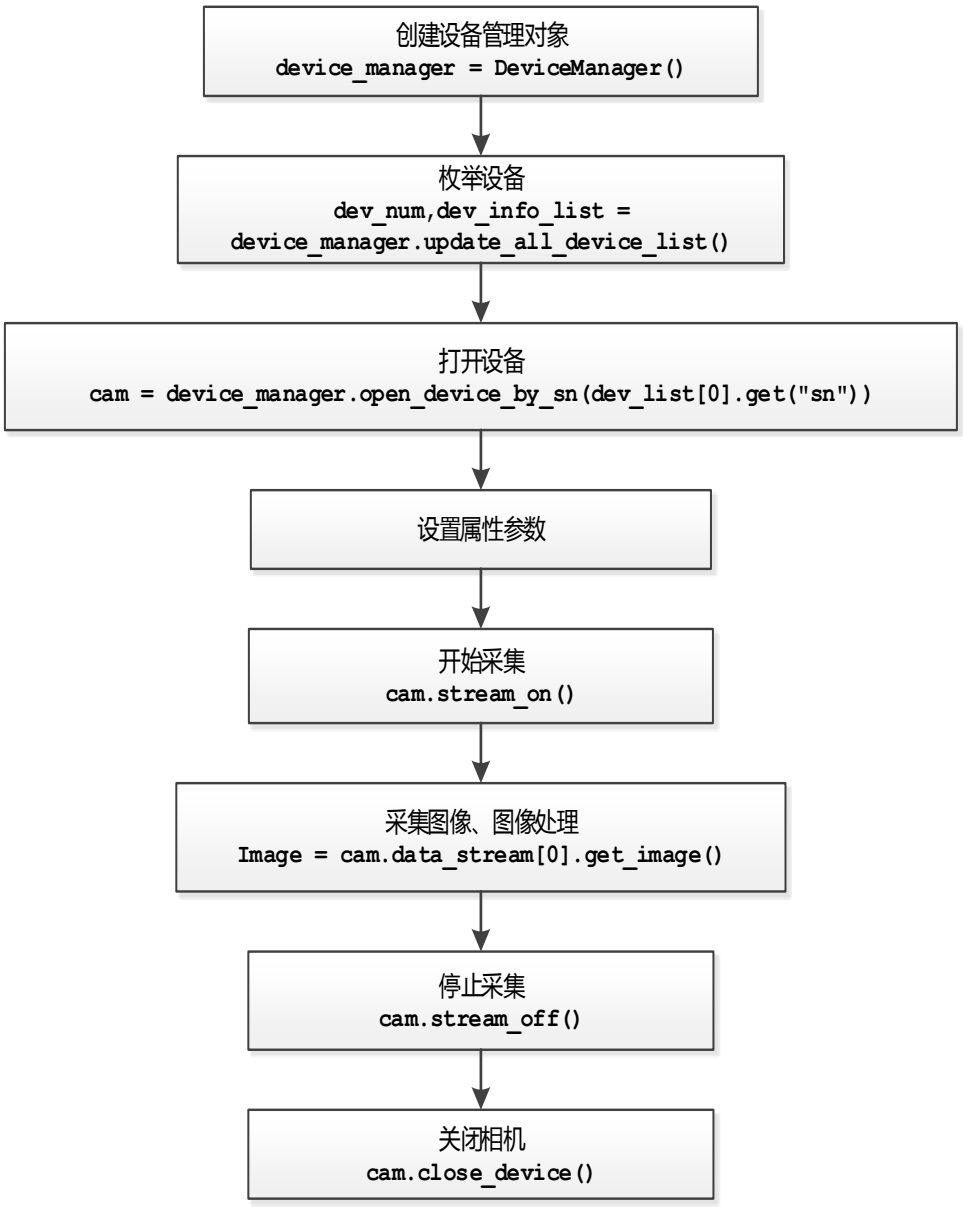
支持信箱：support@daheng-imaging.com

支持热线：400-999-7595

目录

1. 相机工作流程

1.1. 整体工作流程



1.2. 功能控制流程

1.3. 整体代码样例

```
# 用户可自定义调用前缀，样例中使用了 gx
import gxipy as gx

# 枚举设备。dev_info_list 是设备信息列表，列表的元素个数为枚举到的设备个数，列表元素是字典，其中包含设备索引（index）、ip 信息（ip）等设备信息
device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)

# 打开设备
# 获取设备基本信息列表
strSN = dev_info_list[0].get("sn")
# 通过序列号打开设备
cam = device_manager.open_device_by_sn(strSN)

# 开始采集
cam.stream_on()

# 获取流通道个数
# 如果 int_channel_num == 1，设备只有一个流通道，列表 data_stream 元素个数为 1
# 如果 int_channel_num > 1，设备有多个流通道，列表 data_stream 元素个数大于 1
# 目前千兆网相机、USB3.0、USB2.0 相机均不支持多流通道。
# int_channel_num = cam.get_stream_channel_num()

# 获取数据
# num 为采集图片次数
num = 1
for i in range(num):
    # 从第 0 个流通道获取一幅图像
    raw_image = cam.data_stream[0].get_image()
    # 从彩色原始图像获取 RGB 图像
    rgb_image = raw_image.convert("RGB")
    if rgb_image is None:
        continue
    # 从 RGB 图像数据创建 numpy 数组
    numpy_image = rgb_image.get_numpy_array()
    if numpy_image is None:
        continue
    # 显示并保存获得的 RGB 图片
    image = Image.fromarray(numpy_image, 'RGB')
    image.show()
    image.save("image.jpg")

# 停止采集，关闭设备
cam.stream_off()
cam.close_device()
```

2. 编程指引

2.1. 搭建编程环境

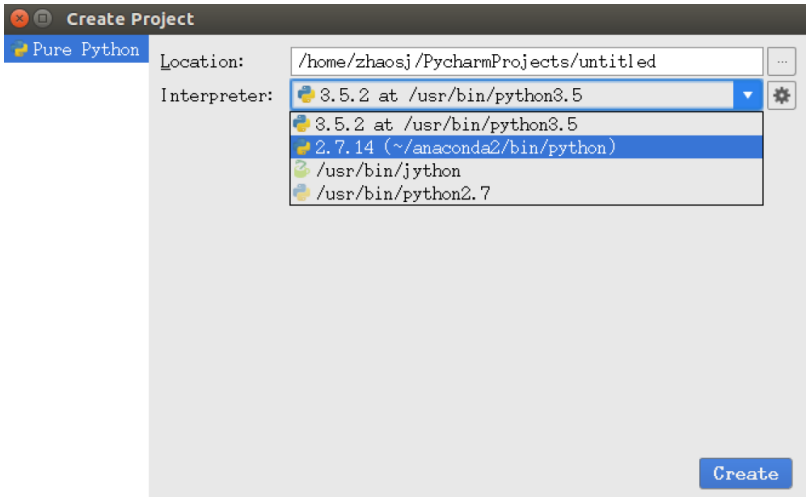
推荐用户优先使用 Python2.7、Python3.5 版本，本接口库已在上述两版本环境中测试通过。

2.1.1. Linux

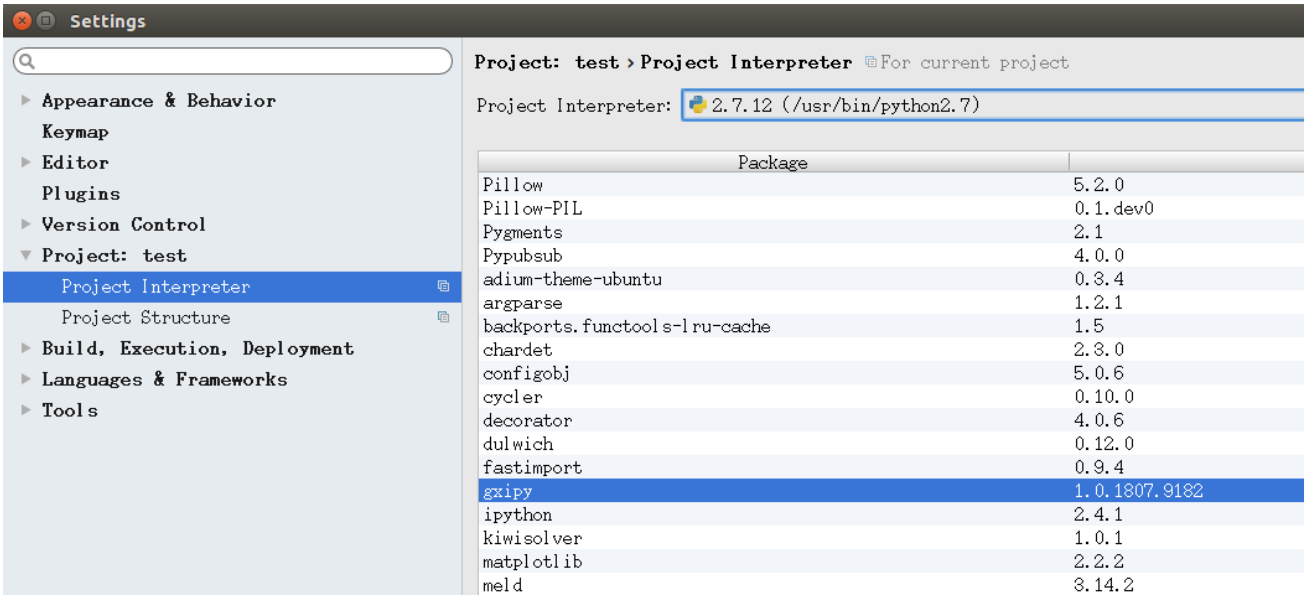
1) 安装 pycharm-community（社区免费版本）

sudo apt-get install pycharm-community

2) 新建 project，选择已安装 gxipy 库的 python 解释器，如果没有可选项，点击右侧的“设置”图标按钮，添加对应版本的 python 解释器。



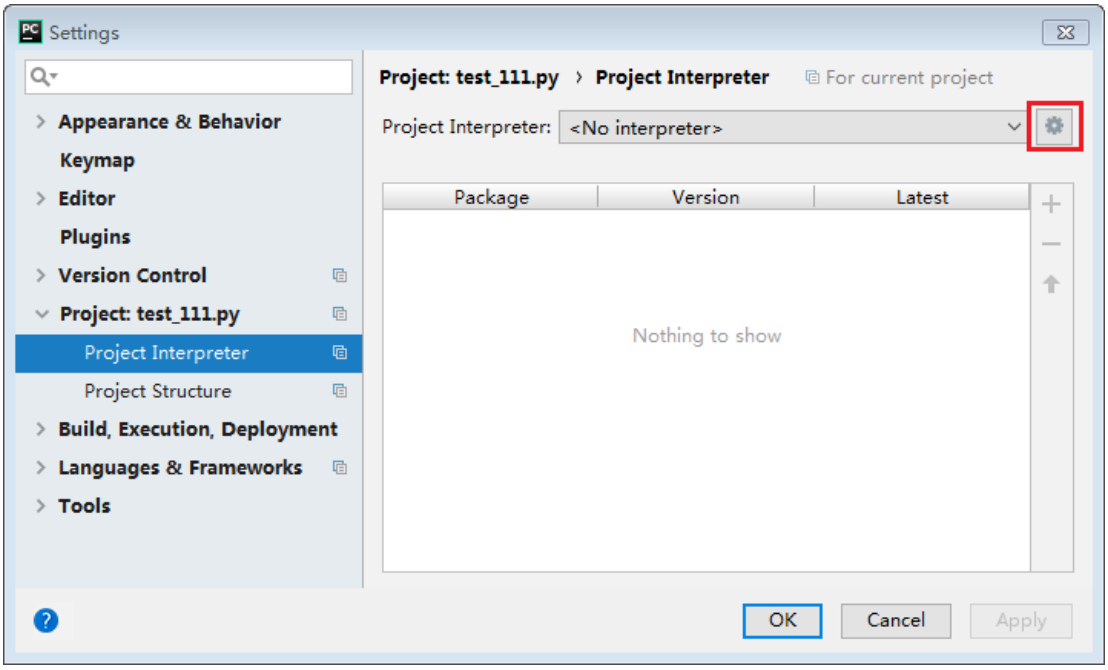
3) File->Settings->Project->Project Interpreter 可以查看 python 解释器已安装的 Package，如图表示 gxipy 库已正常安装，可以导入 gxipy 模块使用。



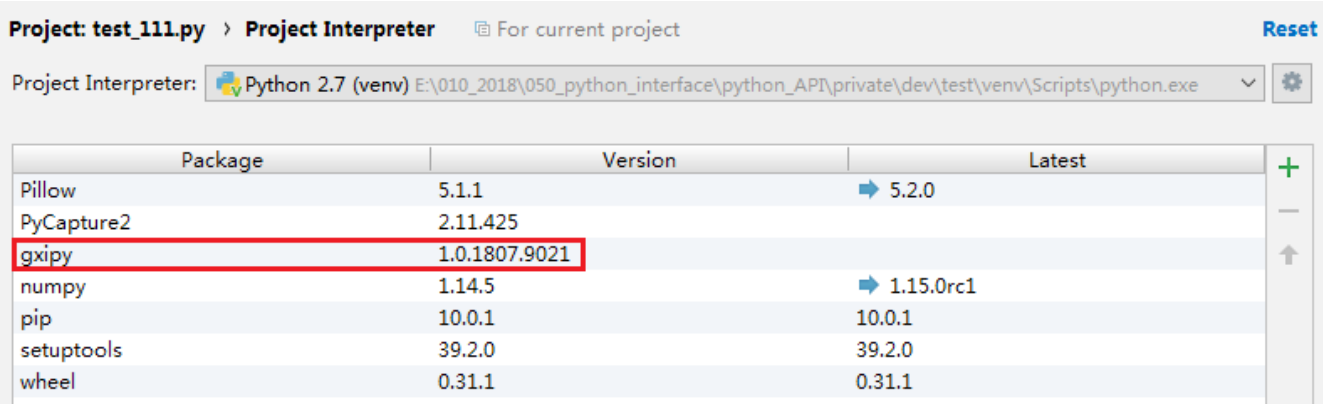
2.1.2. Windows

以 Python2.7、pycharm2018 平台为例，在 windows 7 64 位操作系统环境下演示安装 gxipy 库。初次安装 gxipy 库时：

1) 打开 pycharm2018 的工程配置 File->Settings->Project->Project Interpreter，点击下图红色方框出现的 Add。



2) 选择 New environment，选中 Inherit global site-packages 和 Make available to all projects，点击 OK。这时用户将在新建的 interpreter 中看到 gxipy 库已被成功加载到当前工程环境中。



2.2. 快速上手

2.2.1. 引入库

为使用已安装的库，在程序的开始引入库。在代码中任何使用库中的类、方法、数据类型时，请加自定义的前缀。

代码样例：

```
# 用户可自定义调用前缀，样例中使用了 gx
import gxipy as gx

device_manager = gx.DeviceManager()
```

2.2.2. 枚举设备

用户通过调用 DeviceManager.update_device_list()枚举当前所有可用设备，函数返回值为设备数量 dev_num 和设备信息列表 dev_info_list。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型为字典，字典的键详见下表：

注： MER-GEV：水星一代千兆网相机；MER-U3V：水星一代 USB3.0 相机；MER-U2：水星一代 USB2.0 相机；MER2-GEV：水星二代千兆网相机。

键名称	意义	类型	MER-GEV	MER-U3V	MER-U2	MER2-GEV
index	设备索引	整型	√	√	√	√
vendor_name	厂商名称	字符串	√	√	√	√
model_name	设备型号	字符串	√	√	√	√
sn	设备序列号	字符串	√	√	√	√
display_name	设备显示名称	字符串	√	√	√	√
device_id	设备标识	字符串	√	√	√	√
user_id	用户自定义名称	字符串	√	√	√	√
access_status	权限状态	GxAccessStatus	√	√	√	√
device_class	设备类	GxDeviceClassList	√	√	√	√
mac	mac 地址（GEV 相机特有）	字符串	√			√
ip	ip 地址（GEV 相机特有）	字符串	√			√
subnet_mask	子网掩码（GEV 相机特有）	字符串	√			√
gateway	网关（GEV 相机特有）	字符串	√			√
nic_mac	nic_mac 地址（GEV 相机特有）	字符串	√			√
nic_ip	nic_ip 地址（GEV 相机特有）	字符串	√			√
nic_subnet_mask	nic 子网掩码（GEV 相机特有）	字符串	√			√
nic_gateway	nic 网关（GEV 相机特有）	字符串	√			√
nic_description	nic 描述（GEV 相机特有）	字符串	√			√

枚举设备代码片段如下：

```
# 枚举设备
device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)
```

注意：

除上面的枚举接口，DeviceManager 还提供了另一个枚举接口 DeviceManager.update_all_device_list()：

1) 对于非千兆网相机来说，这两个枚举接口功能上是一样的；

2) 对千兆网相机来说，库内部使用的枚举机制不一样：

update_all_device_list：使用全网枚举，能够枚举到局域网内的所有千兆网相机

update_device_list：使用子网枚举，只能枚举到局域网内的同一网段的千兆网相机

2.2.3. 打开关闭设备

用户通过调用以下五种不同的方式打开设备：

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)

其中：

sn 为设备序列号

use_id 为用户自定义名称

index 为设备索引（1,2,3...）

mac 为设备 mac 地址（非千兆网相机不可用）

ip 为设备 ip 地址（非千兆网相机不可用）

注意：

最后两个函数只针对千兆网相机使用。

用户可以调用 Device 提供的 Device. close_device()接口来关闭设备，释放所有设备资源。

代码样例：

```
# 用户可自定义调用前缀，样例中使用了 gx
import gxipy as gx

# 枚举设备。dev_info_list 是设备信息列表，列表的元素个数为枚举到的设备个数，列表元素是字典，其中包含设备索引（index）、ip 信息（ip）等设备信息
device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)

# 打开设备
# 方法一
# 获取设备基本信息列表
str_sn = dev_info_list[0].get("sn")
# 通过序列号打开设备
```

```
cam = device_manager.open_device_by_sn(str_sn)

# 方法二
# 通过用户 ID 打开设备
# str_user_id = dev_info_list[0].get("user_id")
# cam = device_manager.open_device_by_user_id(str_user_id)

# 方法三
# 通过索引打开设备
# str_index = dev_info_list[0].get("index")
# cam = device_manager.open_device_by_index(str_index)

# 下面为只针对千兆网相机使用的打开方式

# 方法四
# 通过 ip 地址打开设备
# str_ip= dev_info_list[0].get("ip")
# cam = device_manager.open_device_by_ip(str_ip)

# 方法五
# 通过 mac 地址打开设备
# str_mac = dev_info_list[0].get("mac")
# cam = device_manager.open_device_by_mac(str_mac)

# 关闭设备
cam.close_device()
```

2.2.4. 采集控制

在设备正常开启并设置好相机采集参数后，用户可调用 Device.stream_on()和 Device.stream_off()执行开停采：

与采集相关的接口和控制都由 DataStream 提供，可通过设置循环次数控制采集图像次数。通过 Device.get_stream_channel_num()接口获得设备的流通道个数。获取的图像使用 RawImage.get_status()判断是否为残帧。返回的数据结构为代码如下：

```
# 开始采集
cam.stream_on()

# 获取流通道个数
# 如果 int_channel_num == 1，设备只有一个流通道，列表 data_stream 元素个数为 1
# 如果 int_channel_num > 1，设备有多个流通道，列表 data_stream 元素个数大于 1
# 目前千兆网相机、USB3.0、USB2.0 相机均不支持多流通道
# int_channel_num = cam.get_stream_channel_num()

# 获取数据
# num 为采集图片次数
num = 1
for i in range(num):
```

```
    # 打开第 0 通道数据流
    raw_image = cam.data_stream[0].get_image()
    if raw_image.get_status() == gx.GxFrameStatusList.INCOMPLETE:
        print("incomplete frame")

# 停止采集
cam.stream_off()
```

2.2.5. 图像处理

- **图像格式转换**

图像格式转换的对象是采集图像 DataStream.get_image()得到的 raw_image。

功能描述：

将 Bayer 格式图像转换成 RGB 格式图像。详见 RawImage 的接口 RawImage.convert()。

代码样例：

1) 彩色相机

```
raw_image = cam.data_stream[0].get_image()
# 保存 raw 图
raw_image.save_raw("raw_image.raw")
# 从彩色原始图像获取 RGB 图像
rgb_image = raw_image.convert("RGB")
if rgb_image is None:
    continue
# 从 RGB 图像数据创建 numpy 数组
numpy_image = rgb_image.get_numpy_array()
if numpy_image is None:
    continue
# 之后，用户可根据获取的 numpy_array 显示、保存图像
```

2) 黑白相机

```
raw_image = cam.data_stream[0].get_image()
# 从黑白原始图像获取 numpy 数组
numpy_image = raw_image.get_numpy()
if numpy_image is None:
    continue
# 之后，用户可根据获取的 numpy_array 显示、保存图像
```

- **图像质量提升**

本接口库还提供了软件端的图像质量提升接口，用户可以有选择的进行颜色校正、对比度、Gamma 等图像质量提升的操作。用户可调用 RGBImage 的接口 RGBImage.image_improvement()实现功能。一个简单的使用范例如下：

```
# 设置图像质量提升的参数
if cam.GammaParam.is_readable():
    gamma_value = cam.GammaParam.get()
    gamma_lut = gx.Utility.get_gamma_lut(gamma_value)
else:
    gamma_lut = None
```

```
if cam.ContrastParam.is_readable():
    contrast_value = cam.ContrastParam.get()
    contrast_lut = gx.Utility.get_contrast_lut(contrast_value)
else:
    contrast_lut = None
color_correction_param = cam.ColorCorrectionParam.get()

# 采集获取图像、格式转换
# .......
# 实现图像质量提升
rgb_image.image_improvement(color_correction_param, contrast_lut,
gamma_lut)
```

1) 颜色校正

设备属性参数名称：ColorCorrectionParam

名词解释：提高相机色彩还原度，使图像更加接近人眼视觉感受。

2) 对比度调节

Contrast 值：整型，范围[-50, 100]，缺省值为 0

设备属性参数名称：ContrastParam

名词解释：图像明亮部分与黑暗部分的亮度比称为对比度，又叫反差。对比度高或者反差大的图像，其中被摄景物的轮廓较清楚，图像也较清晰；反之，对比度低的图像轮廓不清，图像也不太清晰。

3) Gamma 调节

Gamma 值：整型或浮点型，范围[0.1, 10.0]，缺省值为 1

设备属性参数名称： GammaParam

名词解释：Gamma 调节是为了让显示器的输出尽量接近输入。

◆ **图像显示和保存**

调用 PIL(Python Imaging Library)的接口 Image.fromarray()，将 numpy 数组转换成 Image 图像，显示并保存。代码如下：

1) 黑白相机

```
# 显示并保存获得的黑白图片
image = Image.fromarray(numpy_image, 'L')
image.show()
image.save("acquisition_mono_image.jpg")
```

2) 彩色相机

```
# 显示并保存获得的彩色图片
image = Image.fromarray(numpy_image, 'RGB')
image.show()
image.save("acquisition_RGB_image.jpg")
```

2.2.6. 相机控制

◆ **属性参数访问类型**

属性参数的访问分三种类型：是否实现、是否可读、是否可写。接口设计如下：

Feature.is_implement	当前属性控制器是否支持此功能
Feature.is_readable	此功能是否可读
Feature.is_writable	此功能是否可写

建议用户在操作属性参数前，先查询属性的访问类型。

代码样例：

```
# 获取功能是否实现
is_implemented = cam.PixelFormat.is_implemented()
if is_implemented == True:
    # 获取是否可写
    is_writable = cam.PixelFormat.is_writable()
    if is_writable == True:
        # 设置像素格式
        cam.PixelFormat.set(gx.GxPixelFormatEntry.MONO8)
    # 获取是否可读
    is_readable = cam.PixelFormat.is_readable()
    if is_readable == True:
       # 打印像素格式
       print(cam.PixelFormat.get())
```

- **属性控制**

属性控制分为设备属性参数和流属性参数两类。

区别在于属性参数的控制功能有所不同：

设备属性参数：设备信息，比如宽高、曝光增益等；

流属性参数：关于采集控制和采集数据统计的属性访问控制器。

当获得相机实例化对象后，可通过访问其属性参数，访问到属性参数的控制接口，这些接口在控制相机功能中。

例如：

```
cam.UserSetSelector.set(gx.GxUserSetEntry.DEFAULT)
cam.UserSetLoad.send_command()
cam.UserSetSelector.set(gx.GxUserSetEntry.USER_SET0)
cam.UserSetLoad.send_command()
cam.UserSetDefault.set(gx.GxUserSetEntry.DEFAULT)
```

相机实例化对象　属性参数　控制属性参数方法

```
cam.data_stream[0].StreamLostFrameCount.get()
```

相机实例化对象　流对象　属性参数　控制属性参数方法

接口调用根据属性参数的类型的不同分类：

整型：

相关接口：

IntFeature.set(int_value) //设置

IntFeature.get() //读取

IntFeature.get_range() //获取最小值、最大值、步长

代码样例：

```
# 获取图像宽度可设置范围
int_range = cam.Width.get_range()
# 设置当前图像宽度为范围内任意值
cam.Width.set(800)
# 获取当前图像宽度
int_Width_value = cam.Width.get()
```

浮点型：

相关接口：

FloatFeature.set(float_value) //设置

FloatFeature.get() //读取

FloatFeature.get_range() //获取最小值、最大值、步长、单位、单位是否有效

代码样例：

```
# 获取曝光值可设置范围和最大值
float_range = cam.ExposureTime.get_range()
float_max = float_range["max"]
# 设置当前曝光值范围内任意值
cam.ExposureTime.set(10.0)
# 获取当前曝光值
float_exposure_value = cam.ExposureTime.get()
```

枚举型：

相关接口：

EnumFeature.set(enum_value) //设置

EnumFeature.get() //读取

EnumFeature.get_range() //获取字典

代码样例：

```
# 获取枚举值可设置范围。
enum_range = cam.PixelFormat.get_range()
# 设置当前枚举值
cam.PixelFormat.set(gx.GxPixelFormatEntry.MONO8)
# 打印当前枚举值
enum_PixelForamt_value , enum_PixelFormat_key = cam.PixelFormat.get()
```

布尔型：

相关接口：

BoolFeature.set(bool_value) //设置

BoolFeature.get() //读取

代码样例：

```
# 设置当前布尔值
cam.LineInverter.set(True)
# 读取布尔值
bool_LineInverter_value = cam.LineInverter.get()
```

字符串型：

相关接口：

StringFeature.set(string_value) //设置

StringFeature.get() //读取

StringFeature.get_string_max_length() //获取字符串型属性参数的最大长度值

代码样例：

```
# 读取最长字符串长度
string_max_length = cam.DeviceUserID.get_string_max_length()
# 读取当前字符串值
current_string = cam.DeviceUserID.get()
# 设置字符串值
cam.DeviceUserID.set("MyUserID")
```

Buffer 型：

相关接口：

BufferFeature.set_buffer(buf) //设置

BufferFeature.get_buffer() //读取

BufferFeature.get_buffer_length() //获取 Buffer 型属性参数的长度

代码样例：

```
import gxipy as gx
# 读取缓冲数据长度
buffer_length = cam.UserData.get_buffer_length()
# 设置缓冲数据
cam.UserData.set_buffer(gx.Buffer.from_string(b'BufferFeature Test!'))
# 读取缓冲数据
buffer_data = cam.UserData.get_buffer()
print("UserData: %s" % (buffer_data.get_data().decode())
```

Command 型：

相关接口：

CommandFeature.send_command //发送命令

代码样例：

```
# 发送命令：开停采
cam.AcquisitionStart.send_command()
cam.AcquisitionStop.send_command()
```

不同类型的设备具备的属性功能也略有差别。

在附录【属性参数】中可获取相机所有的属性参数。

2.2.7. 导入导出相机配置参数

在接口库中有供用户调用的导入导出设备配置文件的接口代码样例：

```
# 导入配置相机配置参数文件
cam.import_config_file("import_config_file.txt")
# 导出配置相机配置参数文件
cam.export_config_file("export_config_file.txt")
```

2.2.8. 错误处理

当调用接口函数内部出现异常时，错误处理机制会检测并抛出不同类型异常，异常类型均继承自Exception。

一个典型的错误处理代码样例：

```
try:
# 调用接口函数时函数内部抛异常
dev_num, dev_info_list = device_manager.updata_device_list()
except Exception as exception:
    print("打印错误信息：%s" % exception)
    exit(1)
```

用户也可通过判断捕获的具体错误类型，进行分类处理：

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

异常类型：

异常类型	意义
UnexpectedError	未预测
NotFoundTL	没找到 TL
NotFoundDevice	未找到设备
OffLine	掉线
InvalidParameter	参数无效
InvalidHandle	句柄无效
InvalidCall	无效回调
InvalidAccess	无效获取
NeedMoreBuffer	Buffer 不足

FeatureTypeError	功能码错误
OutOfRange	超过范围
NotInitApi	未初始化
Timeout	超时
ParameterTypeError	参数类型错误

3. 附录

3.1. 属性参数

注：MER-GEV：水星一代千兆网相机；MER-U3V：水星一代 USB3.0 相机；MER-U2：水星一代 USB2.0 相机；MER2-GEV：水星二代千兆网相机。

3.1.1. 设备属性参数

属性参数	解释	属性类	相机型号			
DeviceInformation Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
DeviceVendorName	厂商名称	StringFeature	√	√	√	√
DeviceModelName	设备型号	StringFeature	√	√	√	√
DeviceFirmwareVersion	设备固件版本	StringFeature	√	√	√	√
DeviceVersion	设备版本	StringFeature	√	√	√	√
DeviceSerialNumber	设备序列号	StringFeature	√	√	√	√
FactorySettingVersion	出厂参数版本	StringFeature	√	√	√	√
DeviceUserID	用户自定义名称	StringFeature	√	√	√	√
DeviceLinkSelector	设备链路选择，详见 C 软件开发说明书	IntFeature	√	√	√	√
DeviceLinkThroughputLimit	设备链路带宽限制	IntFeature	√	√	√	√
DeviceLinkThroughputLimitMode	设备带宽限制模式，详见 GxSwitchEntry	EnumFeature	√	√	√	√
DeviceLinkCurrentThroughput	当前设备采集带宽	IntFeature	√	√	√	√
DeviceReset	设备复位	CommandFeature				√
TimestampTickFrequency	时间戳时钟频率	IntFeature				√
TimestampLatch	时间戳锁存	CommandFeature				√
TimestampReset	重置时间戳	CommandFeature				√
TimestampLatchReset	重置时间戳锁存	CommandFeature				√
TimestampLatchValue	时间戳锁存值	IntFeature				√
ImageFormat Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
SensorWidth	传感器宽度	IntFeature	√	√	√	√
SensorHeight	传感器高度	IntFeature	√	√	√	√
WidthMax	最大宽度	IntFeature	√	√	√	√

HeightMax	最大高度	IntFeature	√	√	√	√
OffsetX	水平偏移	IntFeature	√	√	√	√
OffsetY	垂直偏移	IntFeature	√	√	√	√
Width	图像宽度	IntFeature	√	√	√	√
Height	图像高度	IntFeature	√	√	√	√
BinningHorizontal	水平像素 Binning	IntFeature	√	√	√	√
BinningVertical	垂直像素 Binning	IntFeature	√	√	√	√
DecimationHorizontal	水平像素抽样	IntFeature	√	√	√	√
DecimationVertical	垂直像素抽样	IntFeature	√	√	√	√
PixelSize	像素位深，详见 GxPixelSizeEntry	EnumFeature	√	√	√	√
PixelColorFilter	Bayer 格式，详见 GxPixelColorFilterEntry	EnumFeature	√	√	√	√
PixelFormat	像素格式，详见 GxPixelFormatEntry	EnumFeature	√	√	√	√
ReverseX	水平翻转	BoolFeature	√	√	√	√
ReverseY	垂直翻转	BoolFeature	√	√	√	√
TestPattern	测试图，详见 GxTestPatternEntry	EnumFeature	√	√	√	√
TestPatternGeneratorSelector	测试图源选择，详见 C 软件开发说明书和 GxTestPatternGeneratorSelectorEntry	EnumFeature	√	√	√	√
RegionSendMode	ROI 输出模式，详见 GxRegionSendModeEntry	EnumFeature	√	√	√	√
RegionMode	区域开关，详见 GxSwitchEntry	EnumFeature	√	√	√	√
RegionSelector	区域选择，详见 C 软件开发说明书和 GxRegionSelectorEntry	EnumFeature	√	√	√	√
CenterWidth	窗口宽度	IntFeature		√ *		
CenterHeight	窗口高度	IntFeature		√ *		
BinningHorizontalMode	水平像素 Binning 模式,详见 GxBinningHorizontalModeEntry	EnumFeature				√
BinningVerticalMode	垂直像素 Binning 模式,详见 GxBinningVerticalModeEntry	EnumFeature				√
TransportLayer Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
PayloadSize	数据大小	IntFeature	√	√	√	√
CenterWidth	窗口宽度	IntFeature	√	√	√	√
CenterHeight	窗口高度	IntFeature	√	√	√	√

GevCurrentIPConfigurationLLA	LLA 方式配置 IP	BoolFeature	√			√
GevCurrentIPConfigurationDHCP	DHCP 方式配置 IP	BoolFeature	√			√
GevCurrentIPConfigurationPersistentIP	永久 IP 方式配置 IP	BoolFeature	√			√
EstimatedBandwidth	预估带宽	IntFeature	√			√
GevHeartbeatTimeout	心跳超时时间	IntFeature	√			√
GevSCPSPacketSize	流通道包长	IntFeature	√			√
GevSCPD	流通道包间隔	IntFeature	√			√
GevLinkSpeed	连接速度	IntFeature	√			√
DigitalIO Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
UserOutputSelector	用户自定义输出选择，详见 C 软件开发说明书和 GxUserOutputSelectorEntry	EnumFeature	√	√	√	√
UserOutputValue	用户自定义输出值	BoolFeature	√	√	√	√
UserOutputMode	用户 IO 输出模式，详见 GxUserOutputModeEntry	EnumFeature			√	
StrobeSwitch	闪光灯开关，详见 GxSwitchEntry	EnumFeature			√	
LineSelector	引脚选择，详见 C 软件开发说明书和 GxLineSelectorEntry	EnumFeature	√	√	√	√
LineMode	引脚方向，详见 GxLineModeEntry	EnumFeature	√	√	√	√
LineSource	引脚输出源，详见 GxLineSourceEntry	EnumFeature	√	√	√	√
LineInverter	引脚电平反转	BoolFeature	√	√	√	√
LineStatus	引脚状态	BoolFeature	√	√	√	√
LineStatusAll	所有引脚的状态	IntFeature	√	√	√	√
AnalogControls Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
GainAuto	自动增益，详见 GxAutoEntry	EnumFeature	√	√	√	√
GainSelector	增益通道选择，详见 C 软件开发说明书和 GxGainSelectorEntry	EnumFeature	√	√	√	√
BlackLevelAuto	自动黑电平，详见 GxAutoEntry	EnumFeature	√	√	√	√
BlackLevelSelector	黑电平通道选择，详见 C 软件开发说明书和 GxBlackLevelSelectEntry	EnumFeature	√	√	√	√
BalanceWhiteAuto	自动白平衡，详见 GxAutoEntry	EnumFeature	√	√	√	√
BalanceRatioSelector	白平衡通道选择，详见 C 软件开发说明书和 GxBalanceRatioSelectorEntry	EnumFeature	√	√	√	√

BalanceRatio	白平衡系数	FloatFeature	√	√	√	√
DeadPixelCorrect	坏点校正	EnumFeature	√	√	√	√
Gain	增益	FloatFeature	√	√	√	√
BlackLevel	黑电平	FloatFeature	√	√	√	√
GammaEnable	Gamma 使能	BoolFeature				√
GammaMode	Gamma 模式,详见 GxGammaModeEntry	EnumFeature				√
Gamma	Gamma	FloatFeature				√
DigitalShift	数字移位	IntFeature				√
CustomFeature Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
ADCLevel	AD 转换级别	IntFeature			√	
HBlanking	水平消隐	IntFeature			√	
VBlanking	垂直消隐	IntFeature			√	
UserPassword	用户加密区密码	StringFeature			√	
VerifyPassword	用户加密区校验密码	StringFeature			√	
UserData	用户加密区内容	BufferFeature			√	
ExpectedGrayValue	期望灰度值	IntFeature	√	√	√	√
AALightEnvironment	自动曝光、自动增益，光照环境类型，详见 GxAALightEnvironmentEntry	EnumFeature			√	
ImageGrayRaiseSwitch	图像亮度拉伸开关，详见 GxSwitchEntry	EnumFeature			√	
AAROIOffsetX	自动调节感兴趣区域 X 坐标	IntFeature	√	√	√	√
AAROIOffsetY	自动调节感兴趣区域 Y 坐标	IntFeature	√	√	√	√
AAROIWidth	自动调节感兴趣区域宽度	IntFeature	√	√	√	√
AAROIHeight	自动调节感兴趣区域高度	IntFeature	√	√	√	√
AutoGainMin	自动增益最小值	FloatFeature	√	√	√	√
AutoGainMax	自动增益最大值	FloatFeature	√	√	√	√
AutoExposureTimeMin	自动曝光最小值	FloatFeature	√	√	√	√
AutoExposureTimeMax	自动曝光最大值	FloatFeature	√	√	√	√
ContrastParam	对比度参数	IntFeature	√	√	√	√
ColorCorrectionParam	颜色校正系数	IntFeature	√	√	√	√
AWBROIOffsetX	自动白平衡感兴趣区域 X 坐标	IntFeature	√	√	√	√

AWBROIOffsetY	自动白平衡感兴趣区域 Y 坐标	IntFeature	√	√	√	√
AWBROIWidth	自动白平衡感兴趣区域宽度	IntFeature	√	√	√	√
AWBROIHeight	自动白平衡感兴趣区域高度	IntFeature	√	√	√	√
GammaParam	伽马参数	FloatFeature	√	√	√	√
AWBLampHouse	自动白平衡光源，详见 GxAWBLampHouseEntry	EnumFeature	√	√	√	√
SharpnessMode	锐化模式，详见 GxSwitchEntry	EnumFeature	√	√	√	√
Sharpness	锐度	FloatFeature	√	√	√	√
FrameInformation	图像帧信息	BufferFeature			√	
UserSetControl Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
UserSetLoad	加载参数组	CommandFeature	√	√	√	√
UserSetSave	保存参数组	CommandFeature	√	√	√	√
UserSetSelector	参数组选择，详见 C 软件开发说明书和 GxUserSetEntry	EnumFeature	√	√	√	√
UserSetDefault	启动参数组，详见 GxUserSetEntry	EnumFeature	√	√	√	√
LUT Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
LUTValueAll	查找表内容	BufferFeature	√	√	√	√
LUTSelector	查找表选择，详见 C 软件开发说明书和 GxLutSelectorEntry	EnumFeature	√	√	√	√
LUTEnable	查找表使能	BoolFeature				√
LUTIndex	查找表索引	IntFeature				√
LUTValue	查找表值	IntFeature				√
Color Transformation Control			MER-GEV	MER-U3V	MER-U2	MER2-GEV
ColorTransformationMode	颜色转换模式，详见 GxColorTransformationModeEntry	EnumFeature				√
ColorTransformationEnable	颜色转换使能	BoolFeature				√
ColorTransformationValueSelector	颜色转换矩阵元素选择，详见 GxColorTransformationValueSelectorEntry	EnumFeature				√
ColorTransformationValue	颜色转换矩阵元素	FloatFeature				√
ChunkData Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV

ChunkModeActive	帧信息使能	BoolFeature	√	√	√	√
ChunkEnable	单项帧信息使能	BoolFeature	√	√	√	√
ChunkSelector	帧信息项选择，详见 C 软件开发说明书和 GxChunkSelectorEntry	EnumFeature	√	√	√	√
Device Feature			MER-GEV	MER-U3V	MER-U2	MER2-GEV
DeviceCommandTimeout	命令超时	IntFeature	√			√
DeviceCommandRetryCount	命令重试次数	IntFeature	√			√
AcquisitionTrigger Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
FrameBufferOverwriteActive	帧存覆盖使能	BoolFeature	√	√	√	√
AcquisitionStart	开始采集	CommandFeature	√	√	√	√
AcquisitionStop	停止采集	CommandFeature	√	√	√	√
TriggerSoftware	软触发	CommandFeature	√	√	√	√
TransferStart	开始传输	CommandFeature	√	√	√	√
AcquisitionMode	采集模式，详见 GxAcquisitionModeEntry	EnumFeature	√	√	√	√
TriggerMode	触发模式，详见 GxSwitchEntry	EnumFeature	√	√	√	√
TriggerActivation	触发极性，详见 GxTriggerActivationEntry	EnumFeature	√	√	√	√
ExposureAuto	自动曝光，详见 GxAutoEntry	EnumFeature	√	√	√	√
TriggerSource	触发源，详见 GxTriggerSourceEntry	EnumFeature	√	√	√	√
ExposureMode	曝光模式，详见 GxExposureModeEntry	EnumFeature	√	√	√	√
TriggerSelector	触发类型选择，详见 C 软件开发说明书和 GxTriggerSelectorEntry	EnumFeature	√	√	√	√
TransferControlMode	传输控制模式，详见 GxTransferControlModeEntry	EnumFeature	√	√	√	√
TransferOperationMode	传输操作模式，详见 GxTransferOperationModeEntry	EnumFeature	√	√	√	√
AcquisitionFrameRateMode	采集帧率调节模式，详见 GxSwitchEntry	EnumFeature	√	√	√	√
FixedPatternNoiseCorrectMode	模板噪声校正，详见 GxSwitchEntry	EnumFeature	√	√	√	√
ExposureTime	曝光时间	FloatFeature	√	√	√	√
TriggerFilterRaisingEdge	上升沿触发滤波	FloatFeature	√	√	√	√

TriggerFilterFallingEdge	下降沿触发滤波	FloatFeature	√	√	√	√
TriggerDelay	触发延迟	FloatFeature	√	√	√	√
AcquisitionFrameRate	采集帧率	FloatFeature	√	√	√	√
CurrentAcquisitionFrameRate	当前采集帧率	FloatFeature	√	√	√	√
TransferBlockCount	传输帧数	IntFeature	√	√	√	√
TriggerSwitch	外触发开关，详见 GxSwitchEntry	EnumFeature			√	
AcquisitionSpeedLevel	采集速度级别	IntFeature			√	
AcquisitionFrameCount	多帧采集帧数	IntFeature			√	
AcquisitionBurstFrameCount	高速连拍帧数	IntFeature				√
AcquisitionStatusSelector	采集状态选择,详见 C 软件开发说明书和 GxAcquisitionStatusSelectorEntry	EnumFeature				√
AcquisitionStatus	采集状态	BoolFeature				√
ExposureDelay	曝光延迟	FloatFeature				√
CounterAndTimerControl Section			MER-GEV	MER-U3V	MER-U2	MER2-GEV
TimerSelector	计时器选择，详见 GxTimerSelectorEntry	EnumFeature		√		
TimerDuration	计时器持续时间	FloatFeature		√		
TimerDelay	计时器延迟	FloatFeature		√		
TimerTriggerSource	计时器触发源，详见 GxTimerTriggerSourceEntry	EnumFeature		√		
CounterSelector	计数器选择，详见 GxCounterSelectorEntry	EnumFeature		√		
CounterEventSource	计数器事件触发源，详见 GxCounterEventSourceEntry	EnumFeature		√		
CounterResetSource	计数器复位源，详见 GxCounterResetSourceEntry	EnumFeature		√		
CounterResetActivation	计数器复位信号极性，详见 GxCounterResetActivationEntry	EnumFeature		√		
CounterReset	计数器复位	CommandFeature		√		

CenterWidth、CenterHeight（√*）仅支持基于 MER-U3V 的双目相机。

3.1.2. 流属性参数

属性参数	解释	属性类	MER-GEV	MER-U3V	MER-U2	MER2-GEV
StreamAnnouncedBufferCount	声明的 Buffer 个数	IntFeature	√	√	√	√
StreamDeliveredFrameCount	接收帧个数(包括残帧)	IntFeature	√	√	√	√
StreamLostFrameCount	buffer 不足导致的丢帧个数	IntFeature	√	√	√	√
StreamIncompleteFrameCount	接收的残帧个数	IntFeature	√	√	√	√
StreamDeliveredPacketCount	接收到的包数	IntFeature	√	√	√	√
StreamResendPacketCount	重传包个数	IntFeature	√			√
StreamRescuedPacketCount	重传成功包个数	IntFeature	√			√
StreamResendCommandCount	重传命令次数	IntFeature	√			√
StreamUnexpectedPacketCount	异常包个数	IntFeature	√			√
MaxPacketCountInOneBlock	数据块最大重传包数	IntFeature	√			√
MaxPacketCountInOneCommand	一次重传命令最大包含的包数	IntFeature	√			√
ResendTimeout	重传超时时间	IntFeature	√			√
MaxWaitPacketCount	最大等待包数	IntFeature	√			√
ResendMode	重传模式，详见 GxSwitchEntry	EnumFeature	√			√
StreamMissingBlockIDCount	BlockID 丢失个数	IntFeature	√			√
BlockTimeout	数据块超时时间	IntFeature	√			√
MaxNumQueueBuffer	采集队列最大 Buffer 个数	IntFeature	√			√
PacketTimeout	包超时时间	IntFeature	√			√
StreamTransferSize	传输数据块大小	IntFeature		√		
StreamTransferNumberUrb	传输数据块数量	IntFeature		√		

3.2. 功能类定义

3.2.1. Feature

负责查看各种数据类型功能的基础功能，判断其是否已实现、可读、可写。

Feature 类是 IntFeature/FloatFeature/ EnumFeature/BoolFeature/StringFeature/BufferFeature/ CommandFeature 属性类的父类。

接口列表：

is_implemented()	判断属性参数是否已实现
is_readable()	判断属性参数是否可读
is_writable ()	判断属性参数是否可写

◆ **接口说明**

➢ **is_implemented**

声明：

Feature.is_implemented()

意义：

判断属性参数是否已实现

返回值：

Ture：实现

False：未实现

异常处理：

1) 如果属性参数是无效参数，则返回 False。
2) 因为其他原因导致的获取属性参数是否实现失败，则抛出异常，异常类型详见错误处理。

➢ **is_readable**

声明：

Feature.is_readable()

意义：

判断属性参数是否可读

返回值：

Ture：可读

False：不可读

异常处理：

1) 如果功能未实现，则函数返回 False。
2) 如果获取属性参数是否可读失败，则抛出异常，异常类型详见错误处理。

➢ **is_writable**

声明：

Feature.is_writable()

意义：

判断属性参数是否可写。

返回值：

Ture：可写

False：不可写

异常处理：

1) 如果功能未实现，则返回 False。
2) 如果获取属性参数是否可写失败，则抛出异常，异常类型详见错误处理。

3.2.2. IntFeature

负责查看、控制相机的整型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断整型属性参数是否已实现
is_readable()	判断整型属性参数是否可读
is_writable ()	判断整型属性参数是否可写
get_range()	获取整型属性参数范围字典
get()	读取整型属性参数值
set(int_value)	设置整型属性参数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_range**

声明：

IntFeature.get_range()

意义：

获取整型属性参数范围字典

返回值：

记录整型属性参数范围字典。键包含：min 最小值，max 最大值，step 步长

异常处理：

1) 如果该整型属性参数功能未实现，则打印不支持该整型属性参数获取范围的信息，函数返回 None。
2) 如果获取该整型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

IntFeature.get()

意义：

读取整型属性参数值

返回值：

获取的整型值

异常处理：

1) 如果该整型属性参数功能未实现或不可读 ,则打印该整型属性参数不可读的信息 ,函数返回 None。
2) 如果获取该整型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：

IntFeature.set(self,int_value)

意义：

设置整型属性参数值

形参：

设置的整型数值

异常处理：

1) 如果输入参数不是整型值，则抛出”ParameterTypeError”异常。
2) 如果该整型属性参数功能未实现或不可写，则打印该整型属性参数不可写的信息，函数返回 None。
3) 如果输入参数不在该整型属性参数的范围内，则打印超过该整型属性参数范围的信息并打印范围，函数返回 None。
4) 如果设置该整型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.3. FloatFeature

负责查看、控制相机的浮点型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断浮点型属性参数是否已实现
is_readable()	判断浮点型属性参数是否可读
is_writable ()	判断浮点型属性参数是否可写
get_range()	获取浮点型属性参数范围字典
get()	读取浮点型属性参数值
set(float_value)	设置浮点型属性参数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

- **is_writable**

  详见 Feature::is_writable()。

- **get_range**

声明：

FloatFeature.get_range()

意义：

获取浮点型属性参数范围字典

返回值：

记录浮点型属性参数值范围的字典。键包含：min 最小值，max 最大值，inc 步长，unit 单位，inc_is_valid 单位是否有效

异常处理：

1) 如果该浮点型属性参数功能未实现 则打印不支持该浮点型属性参数获取范围的信息 ,函数返回 None。
2) 如果获取该浮点型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

- **get**

声明：

FloatFeature.get()

意义：

读取浮点型属性参数值

返回值：

获取的浮点型属性参数值

异常处理：

1) 如果该浮点型属性参数未实现或不可读 ,则打印该浮点型属性参数不可读的信息 ,函数返回 None。
2) 如果获取浮点型属性参数不成功，则抛出异常，异常类型详见错误处理。

- **set**

声明：

FloatFeature.set(float_value)

意义：

设置浮点型属性参数值

形参：

[in]float_value 设置的浮点型数值

异常处理：

1) 如果输入参数不是浮点型值，则抛出”ParameterTypeError”异常。
2) 如果该浮点型属性参数功能未实现或不可写，则打印该浮点型属性参数不可写的信息，函数返回 None。

3) 如果输入参数不在该浮点型属性参数的范围内，则打印超过该浮点型属性参数范围的信息并打印范围，函数返回 None。

4) 如果设置该浮点型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.4. EnumFeature

负责查看、控制相机的枚举型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断枚举型属性参数是否已实现
is_readable()	判断枚举型属性参数是否可读
is_writable ()	判断枚举型属性参数是否可写
get_range()	获取枚举型属性参数范围字典
get()	读取枚举型属性参数的值和字符串
set(enum_value)	设置枚举型属性参数数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_range**

声明：

EnumFeature.get_range()

意义：

获取枚举型属性参数范围字典

返回值：

记录枚举型属性参数范围的字典

异常处理：

1) 如果该枚举型属性参数功能未实现 则打印不支持该枚举型属性参数获取范围的信息 函数返回 None。

2) 如果获取该枚举型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

EnumFeature.get()

意义：

读取枚举型属性参数的值和字符串

返回值：

1) 枚举型属性参数的数值
2) 枚举型属性参数的字符串

异常处理：

1) 如果该枚举型属性参数功能未实现或不可读 ,则打印该枚举型属性参数不可读的信息 ,函数返回 None。
2) 如果获取枚举型属性参数值不成功，则抛出异常，异常类型详见错误处理。

- **set**

声明：

EnumFeature.set(enum_value)

意义：

设置枚举型属性参数值

形参：

[in]enum_value 设置的枚举型数值

异常处理：

1) 如果输入参数不是整型值，则抛出”ParameterTypeError”异常。
2) 如果该枚举型属性参数功能未实现或不可写，则打印该枚举型属性参数不可写的信息，函数返回 None。
3) 如果输入参数不在枚举型属性参数“值”的范围内，则打印超过该枚举型属性参数范围的信息并打印范围，函数返回 None。
4) 如果设置该枚举型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.5. BoolFeature

负责查看、控制相机的布尔型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断布尔型属性参数是否已实现
is_readable()	判断布尔型属性参数是否可读
is_writable ()	判断布尔型属性参数是否可写
get()	读取布尔型属性参数值
set(bool_value)	设置布尔型属性参数值

- **接口说明**

- **is_implemented**

详见 Feature::is_implemented()。

- **is_readable**

  详见 Feature::is_readable()。

- **is_writable**

  详见 Feature::is_writable()。

- **get**

声明：

BoolFeature.get()

意义：

读取布尔型属性参数值

返回值：

获取的布尔值

异常处理：

1) 如果该布尔型属性参数功能未实现或不可读 则打印该布尔型属性参数不可读的信息 函数返回 None。
2) 如果获取布尔型属性参数值不成功，则抛出异常，异常类型详见错误处理。

- **set**

声明：

BoolFeature.set(bool_value)

意义：

设置浮点型属性参数值

形参：

[in]bool_value 设置的布尔型数值

异常处理：

1) 如果输入参数不是布尔型值，则抛出”ParameterTypeError”异常。
2) 如果该布尔型属性参数功能未实现或不可写，则打印该布尔型属性参数不可写的信息，函数返回 None。
3) 如果设置该布尔型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.6. StringFeature

负责查看、控制相机的字符串型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断字符串型属性参数是否已实现
is_readable()	判断字符串型属性参数是否可读
is_writable ()	判断字符串型属性参数是否可写
get_string_max_length()	获取字符串型属性参数值可设置的最长长度
get()	读取字符串型属性参数值

set(input_string)	设置字符串型属性参数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_string_max_length**

声明：

StringFeature.get_string_max_length()

意义：

获取字符串型属性参数可设置的最大长度

返回值：

字符串型属性参数可设置的最大长度

异常处理：

1) 如果该字符串型属性参数功能未实现 则打印不支持获取该字符串型属性参数的信息 函数返回 None。

2) 如果获取字符串型属性参数值可设置最大长度不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

StringFeature.get()

意义：

读取字符串型属性参数值

返回值：

获取的字符串型属性参数值

异常处理：

1) 如果该字符串型属性参数功能未实现或不可读，则打印该字符串型属性参数不可读的信息，函数返回 None。

2) 如果获取该字符串型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：

StringFeature.set(input_string)

意义：

设置字符串型属性参数值

形参：

[in]input_string　　设置的字符串型数值

异常处理：

1) 如果输入参数不是字符串型值，则抛出”ParameterTypeError”异常。

2) 如果该字符串型属性参数功能未实现或不可写，则打印该字符串型属性参数不可写的信息，函数返回 None。

3) 如果输入参数长度大于可设置最大长度，则打印超过该字符串型属性参数长度最大值的信息并打印最大值，函数返回 None。

4) 如果设置该字符串型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.7. BufferFeature

负责查看、控制相机的缓冲功能，继承自 Feature 类。

接口列表：

is_implemented()	判断 Buffer 型属性参数是否已实现
is_readable()	判断 Buffer 型属性参数是否可读
is_writable ()	判断 Buffer 型属性参数是否可写
get_buffer_length()	获取 Buffer 型属性参数的长度
get_buffer()	读取 Buffer 型属性参数数据
set_buffer(buf)	设置 Buffer 型属性参数数据

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_buffer_length**

声明：

BufferFeature.get_buffer_length()

意义：

获取 Buffer 型属性参数的长度

返回值：

Buffer 型属性参数的长度

异常处理：

1) 如果该 Buffer 型属性参数功能未实现，则打印不支持该 Buffer 属性参数获取范围的信息，函数返回 None。

2) 如果获取 Buffer 型属性参数长度不成功，则抛出异常，异常类型详见错误处理。

- **get_buffer**

声明：

BufferFeature.get_buffer()

意义：

读取 Buffer 型属性参数数据

返回值：

Buffer 对象

异常处理：

1) 如果该 Buffer 型属性参数功能未实现或不可读，则打印该 Buffer 型属性参数不可读的信息，函数返回 None。

2) 如果获取该 Buffer 型属性参数数据不成功，则抛出异常，异常类型详见错误处理。

- **set_buffer**

声明：

BufferFeature.set_buffer(buf)

意义：

设置 Buffer 型属性参数数据

形参：

[in]buffer 设置的缓冲数据[Buffer 类型]

异常处理：

1) 如果输入参数不是 Buffer 类型，则抛出”ParameterTypeError”异常。

2) 如果该 Buffer 型属性参数功能未实现或不可写，则打印该 Buffer 型属性参数不可写的信息，函数返回 None。

3) 如果输入 Buffer 型属性参数数据长度大于最大长度，则打印超过该 Buffer 型属性参数最大长度的信息并打印最大值，函数返回 None。

4) 如果设置该 Buffer 型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.8. CommandFeature

负责制相机的命令型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断命令型属性参数是否已实现
is_readable()	判断命令型属性参数是否可读
is_writable ()	判断命令型属性参数是否可写
send_command()	发送命令

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **send_command**

声明：

CommandFeature.send_command()

意义：

发送命令

异常处理：

1) 如果该 Command 型属性参数功能未实现，则打印不支持该 Command 型属性参数的信息，函数返回 None。

2) 如果发送命令不成功，则抛出异常，异常类型详见错误处理。

3.3. 数据类型定义

3.3.1. GxDeviceClassList

定义	值	解释
UNKNOWN	0	未知设备种类
USB2	1	USB2.0 相机
GEV	2	千兆网相机（GigE Vision）
U3V	3	USB3.0 相机（USB3 Vision）

3.3.2. GxAccessStatus

定义	值	解释
UNKNOWN	0	设备当前状态未知

READWRITE	1	设备当前可读可写
READONLY	2	设备当前仅支持读
NOACCESS	3	设备当前既不支持读，又不支持写

3.3.3. GxAccessMode

定义	值	解释
READONLY	2	以只读方式打开设备
CONTROL	3	以控制方式打开设备
EXCLUSIVE	4	以独占方式打开设备

3.3.4. GxPixelFormatEntry

定义	值	解释
UNDEFINED	0x00000000	未定义
MONO8	0x01080001	Monochrome 8-bit
MONO8_SIGNED	0x01080002	Monochrome 8-bit signed
MONO10	0x01100003	Monochrome 10-bit unpacked
MONO12	0x01100005	Monochrome 12-bit unpacked
MONO14	0x01100025	Monochrome 14-bit unpacked
MONO16	0x01100007	Monochrome 16-bit
BAYER_GR8	0x01080008	Bayer Green-Red 8-bit
BAYER_RG8	0x01080009	Bayer Red-Green 8-bit
BAYER_GB8	0x0108000A	Bayer Green-Blue 8-bit
BAYER_BG8	0x0108000B	Bayer Blue-Green 8-bit
BAYER_GR10	0x0110000C	Bayer Green-Red 10-bit
BAYER_RG10	0x0110000D	Bayer Red-Green 10-bit
BAYER_GB10	0x0110000E	Bayer Green-Blue 10-bit
BAYER_BG10	0x0110000F	Bayer Blue-Green 10-bit
BAYER_GR12	0x01100010	Bayer Green-Red 12-bit
BAYER_RG12	0x01100011	Bayer Red-Green 12-bit
BAYER_GB12	0x01100012	Bayer Green-Blue 12-bit
BAYER_BG12	0x01100013	Bayer Blue-Green 12-bit
BAYER_GR16	0x0110002E	Bayer Green-Red 16-bit
BAYER_RG16	0x0110002F	Bayer Red-Green 16-bit
BAYER_GB16	0x01100030	Bayer Green-Blue 16-bit
BAYER_BG16	0x01100031	Bayer Blue-Green 16-bit
RGB8_PLANAR	0x02180021	Red-Green-Blue 8-bit planar
RGB10_PLANAR	0x02300022	Red-Green-Blue 10-bit planar
RGB12_PLANAR	0x02300023	Red-Green-Blue 12-bit planar
RGB16_PLANAR	0x02300024	Red-Green-Blue 16-bit planar

3.3.5. GxFrameStatusList

定义	值	解释
SUCCESS	0	正常帧
IMCOMPLETE	-1	残帧

3.3.6. GxPixelSizeEntry

定义	值	解释
BPP8	8	像素大小 BPP8
BPP10	10	像素大小 BPP10
BPP12	12	像素大小 BPP12

BPP16	16	像素大小 BPP16
BPP24	24	像素大小 BPP24
BPP30	30	像素大小 BPP30
BPP32	32	像素大小 BPP32
BPP36	36	像素大小 BPP36
BPP48	48	像素大小 BPP48
BPP64	64	像素大小 BPP64

3.3.7. GxPixelColorFilterEntry

定义	值	解释
NONE	0	无
BAYER_RG	1	RG 格式
BAYER_GB	2	GB 格式
BAYER_GR	3	GR 格式
BAYER_BG	4	BG 格式

3.3.8. GxAcquisitionModeEntry

定义	值	解释
SINGLE_FRAME	0	单帧模式
MULITI_FRAME	1	多帧模式
CONTINUOUS	2	连续模式

3.3.9. GxTriggerSourceEntry

定义	值	解释
SOFTWARE	0	软触发
LINE0	1	触发源 0
LINE1	2	触发源 1
LINE2	3	触发源 2
LINE3	4	触发源 3

3.3.10. GxTriggerActivationEntry

定义	值	解释
FALLING_EDGE	0	下降沿触发
RISING_EDGE	1	上升沿触发

3.3.11. GxExposureModeEntry

定义	值	解释
TIMED	1	曝光时间寄存器控制曝光时间
TRIGGER_WIDTH	2	触发信号宽度控制曝光时间

3.3.12. GxUserOutputSelectorEntry

定义	值	解释
OUTPUT0	1	输出 0

OUTPUT1	2	输出 1
OUTPUT2	4	输出 2

3.3.13. GxUserOutputModeEntry

定义	值	解释
STROBE	0	闪光灯
USER_DEFINED	1	用户自定义

3.3.14. GxGainSelectorEntry

定义	值	解释
ALL	0	所有增益通道
RED	1	红通道增益
GREEN	2	绿通道增益
BLUE	3	蓝通道增益

3.3.15. GxBlackLevelSelectEntry

定义	值	解释
ALL	0	所有黑电平通道
RED	1	红通道黑电平
GREEN	2	绿通道黑电平
BLUE	3	蓝通道黑电平

3.3.16. GxBalanceRatioSelectorEntry

定义	值	解释
RED	0	红通道
GREEN	1	绿通道
BLUE	2	蓝通道

3.3.17. GxAALightEnvironmentEntry

定义	值	解释
NATURE_LIGHT	0	自然光
AC50HZ	1	50 赫兹日光灯
AC60HZ	2	60 赫兹日光灯

3.3.18. GxUserSetEntry

定义	值	解释
DEFAULT	0	默认参数组
USER_SET0	1	用户参数组 0

3.3.19. GxAWBLampHouseEntry

定义	值	解释
ADAPTIVE	0	自适应光源
D65	1	指定色温 6500k

FLUORESCENCE	2	指定荧光灯
INCANDESCENT	3	指定白炽灯
D75	4	指定色温 7500k
D50	5	指定色温 5000k
U30	6	指定色温 3000k

3.3.20. GxTestPatternEntry

定义	值	解释
OFF	0	关闭
GRAY_FRAME_RAMP_MOVING	1	静止灰度递增
SLANT_LINE_MOVING	2	滚动斜条纹
VERTICAL_LINE_MOVING	3	滚动竖条纹

3.3.21. GxTriggerSelectorEntry

定义	值	解释
FRAME_START	1	采集一帧
FRAME_BURST_START	2	帧高速连拍开始

3.3.22. GxLineSelectorEntry

定义	值	解释
LINE0	0	引脚 0
LINE1	1	引脚 1
LINE2	2	引脚 2
LINE3	3	引脚 3

3.3.23. GxLineModeEntry

定义	值	解释
INPUT	0	输入
OUTPUT	1	输出

3.3.24. GxLineSourceEntry

定义	值	解释
OFF	0	关闭
STROBE	1	闪光灯
USER_OUTPUT0	2	用户自定义输出 0
USER_OUTPUT1	3	用户自定义输出 1
USER_OUTPUT2	4	用户自定义输出 2
EXPOSURE_ACTIVE	5	曝光有效
FRAME_TRIGGER_WAIT	6	单帧触发等待
ACQUISITION_TRIGGER_WAIT	7	多帧触发等待
TIMER1_ACTIVE	8	定时器 1 有效

3.3.25. GxLutSelectorEntry

定义	值	解释
LUMINANCE	0	亮度

3.3.26. GxTransferControlModeEntry

定义	值	解释
BASIC	0	基础模式
USER_CONTROLED	1	用户控制模式

3.3.27. GxTransferOperationModeEntry

定义	值	解释
MULTI_BLOCK	0	指定发送帧数

3.3.28. GxTestPatternGeneratorSelectorEntry

定义	值	解释
SENSOR	0	sensor 的测试图
REGION0	1	FPGA 的测试图

3.3.29. GxChunkSelectorEntry

定义	值	解释
FRAME_ID	1	帧号
TIME_STAMP	2	时间戳
COUNTER_VALUE	3	计数器值

3.3.30. GxBinningHorizontalModeEntry

定义	值	解释
SUM	0	BINNING 水平值和
AVERAGE	1	BINNING 水平值平均值

3.3.31. GxBinningVerticalModeEntry

定义	值	解释
SUM	0	BINNING 垂直值和
AVERAGE	1	BINNING 垂直值平均值

3.3.32. GxAcquisitionStatusSelectorEntry

定义	值	解释
ACQUISITION_TRIGGER_WAIT	0	采集触发等待
FRAME_TRIGGER_WAIT	1	帧触发等待

3.3.33. GxGammaModeEntry

定义	值	解释
SRGB	0	默认 Gamma 校正
USER	1	用户自定义 Gamma 校正

3.3.34. GxColorTransformationModeEntry

定义	值	解释
RGB_TO_RGB	0	默认颜色校正
USER	1	用户自定义颜色校正

3.3.35. GxColorTransformationValueSelectorEntry

定义	值	解释
GAIN00	0	颜色转换分量增益值 GAIN00
GAIN01	1	颜色转换分量增益值 GAIN01
GAIN02	2	颜色转换分量增益值 GAIN02
GAIN10	3	颜色转换分量增益值 GAIN10
GAIN11	4	颜色转换分量增益值 GAIN11
GAIN12	5	颜色转换分量增益值 GAIN12
GAIN20	6	颜色转换分量增益值 GAIN20
GAIN21	7	颜色转换分量增益值 GAIN21
GAIN22	8	颜色转换分量增益值 GAIN22

3.3.36. GxAutoEntry

定义	值	解释
OFF	0	关闭
CONTINUOUS	1	连续
ONCE	2	单次

3.3.37. GxSwitchEntry

定义	值	解释
OFF	0	关闭
ON	1	开启

3.3.38. GxRegionSendModeEntry

定义	值	解释
SINGLE_ROI	0	单 ROI
MULTI_ROI	1	多 ROI

3.3.39. GxRegionSelectorEntry

定义	值	解释
REGION0	0	区域 0
REGION1	1	区域 1
REGION2	2	区域 2
REGION3	3	区域 3
REGION4	4	区域 4
REGION5	5	区域 5
REGION6	6	区域 6

REGION7	7	区域 7

3.3.40. GxTimerSelectorEntry

定义	值	解释
TIMER1	1	定时器 1

3.3.41. GxTimerTriggerSourceEntry

定义	值	解释
EXPOSURE_START	1	曝光开始信号

3.3.42. GxCounterSelectorEntry

定义	值	解释
COUNTER1	1	计数器 1

3.3.43. GxCounterEventSourceEntry

定义	值	解释
FRAME_START	1	帧开始

3.3.44. GxCounterResetSourceEntry

定义	值	解释
OFF	0	无复位源
SOFTWARE	1	软触发
LINE0	2	引脚 0
LINE1	3	引脚 1
LINE2	4	引脚 2
LINE3	5	引脚 3

3.3.45. GxCounterResetActivationEntry

定义	值	解释
RISING_EDGE	1	上升沿触发

3.3.46. DxBayerConvertType

定义	值	解释
NEIGHBOUR	0	邻域平均插值算法
ADAPTIVE	1	边缘自适应插值算法
NEIGHBOUR3	2	更大区域的邻域平均插值算法

3.3.47. DxValidBit

定义	值	解释
BIT0_7	0	0-7 位
BIT1_8	1	1-8 位
BIT2_9	2	2-9 位
BIT3_10	3	3-10 位

3.3.48. DxImageMirrorMode

定义	值	解释
HORIZONTAL_MIRROR	0	水平镜像
VERTICAL_MIRROR	1	垂直镜像

3.4. 模块接口定义

3.4.1. DeviceManager

负责相机设备的管理，包括枚举设备、打开设备、获取设备数量信息等。

接口列表：

update_device_list (timeout=200)	枚举同一网段中的设备
update_all_device_list (timeout=200)	枚举不同网段中的设备
get_device_number ()	获取设备数量
get_device_info ()	获取设备信息
open_device_by_sn (sn, access_mode=GxAccessMode.CONTROL)	通过序列号打开设备
open_device_by_user_id (user_id, access_mode=GxAccessMode.CONTROL)	通过用户 ID 号打开设备
open_device_by_index (index, access_mode=GxAccessMode.CONTROL)	通过设备索引打开设备
open_device_by_ip (ip, access_mode=GxAccessMode.CONTROL)	通过 IP 地址打开设备
open_device_by_mac (mac, access_mode=GxAccessMode.CONTROL)	通过 mac 地址打开设备

◆ **接口说明**

➢ **update_device_list**

声明：

DeviceManager.update_device_list (timeout=200)

意义：

对于非千兆网相机，枚举所有设备；对于千兆网相机，枚举同一网段设备。

形参：

[in]timeout　枚举超时[0，0xffffffff]，缺省值为 200（ms）

返回值：

枚举得到设备数量和记录枚举设备信息的列表（list）。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型字典，字典中的键名称详见枚举设备

异常处理：

1) 如果输入参数不是整型值，则抛出”ParameterTypeError”异常。

2) 如果输入参数小于 0 或大于无符号整型的最大值，则打印”DeviceManager.update_device_list: Out of bounds, timeout:minimum=0, maximum= 0xffffffff”，函数返回 None。

3) 如果枚举同一网段中设备不成功，则抛出异常，异常类型详见错误处理。

4) 如果获取所有设备基本信息不成功，则抛出异常，异常类型详见错误处理。

- **update_all_device_list**

声明：

DeviceManager.update_all_device_list (timeout=200)

意义：

对于非千兆网相机，枚举所有设备；对于千兆网相机，枚举全网设备。

形参：

[in]timeout 枚举超时[0，0xffffffff]，缺省值为 200（ms）

返回值：

枚举得到设备数量和记录枚举设备信息的列表（list）。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型字典，字典中的键名称详见枚举设备

异常处理：

1) 如果输入参数不是整型值，则抛出”ParameterTypeError”异常。

2) 如果输入参数小于 0 或大于无符号整型的最大值，则打印”DeviceManager.update_all_device_list: Out of bounds, timeout:minimum=0, maximum= 0xffffffff”，函数返回 None。

3) 如果枚举不同网段中设备不成功，则抛出异常，异常类型详见错误处理。

4) 如果获取所有设备基本信息不成功，则抛出异常，异常类型详见错误处理。

- **get_device_number**

声明：

DeviceManager.get_device_number ()

意义：

获取设备数量

返回值：

设备数量

- **get_device_info**

声明：

DeviceManager.get_device_info()

意义：

获取设备信息

返回值：

设备信息列表。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型为字典，字典的键详见枚举设备

- **open_device_by_sn**

声明：

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

意义：

通过序列号打开设备

形参：

[in]sn 序列号[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值：

设备对象

异常处理：

1) 如果输入参数 1 不是字符串型，则抛出”ParameterTypeError”异常。
2) 如果输入参数 2 不是整型，则抛出”ParameterTypeError”异常。
3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值信息，函数返回 None。
4) 如果重复获取设备类不成功，则抛出 NotFoundDevice 异常。
5) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。
6) 如果打开获取的设备不是 U3V/USB2/GEV 类中的一种，则抛出 NotFoundDevice 异常。

- **open_device_by_user_id**

声明：

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

意义：

通过用户 ID 号打开设备

形参：

[in]user_id 用户 ID 号[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值：

设备对象

异常处理：

1) 如果输入参数 1 不是字符串型，则抛出”ParameterTypeError”异常。
2) 如果输入参数 2 不是整型，则抛出”ParameterTypeError”异常。
3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。
4) 如果重复获取设备类不成功，则抛出 NotFoundDevice 异常。
5) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。
6) 如果打开获取的设备不是 U3V/USB2/GEV 类中的一种，则抛出 NotFoundDevice 异常。

- **open_device_by_index**

声明：

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

意义：

通过设备索引打开设备

形参：

[in]index 设备索引[1,2,3...0xffffffff]

[in]access_mode 打开设备方式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值：

设备对象

异常处理：

1) 如果输入参数 1 或 2 不是整型值，则抛出”ParameterTypeError”异常。

2) 如果输入参数 1 小于 0 或大于无符号整型的最大值，则打印”DeviceManager.open_device_by_index: index out of bounds, index: minimum=1, maximum= 0xffffffff”，函数返回 None。

3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。

4) 如果设备数量小于输入参数 1 索引，则抛出 NotFoundDevice 异常。

5) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。

6) 如果打开获取的设备不是 U3V/USB2/GEV 类中的一种，则抛出 NotFoundDevice 异常。

- **open_device_by_ip**

声明：

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

意义：

通过设备 ip 地址打开千兆网相机设备

形参：

[in]ip 设备 ip 地址[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值：

设备对象

异常处理：

1) 如果输入参数 1 不是字符串型，则抛出”ParameterTypeError”异常。

2) 如果输入参数 2 不是整型，则抛出”ParameterTypeError”异常。

3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。

4) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。

➢ **open_device_by_mac**

声明：

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

意义：

通过设备 mac 地址打开千兆网相机设备

形参：

[in]mac 设备 mac 地址[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值：

设备对象

异常处理：

1) 如果输入参数 1 不是字符串型，则抛出”ParameterTypeError”异常。

2) 如果输入参数 2 不是整型，则抛出”ParameterTypeError”异常。

3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。

4) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。

3.4.2. Device

负责相机设备的采集控制、设备关闭、配置文件导入导出和获取设备句柄等。

接口列表：

get_stream_channel_num()	获取当前设备支持的流通道个数
stream_on ()	发送开始命令，相机开始传送图像数据
stream_off ()	发送结束命令，相机结束传送图像数据
export_config_file (file_path)	导出当前配置文件
import_config_file (file_path, verify=False)	导入配置文件
close_device ()	关闭设备，销毁设备句柄，将句柄置为空

◆ **接口说明**

➢ **get_stream_channel_num**

声明：

Device.get_stream_channel_num()

意义：

获取当前设备支持的流通道个数

返回值：

流通道个数

注：目前千兆网相机、USB3.0、USB2.0 相机均不支持多流通道

- **stream_on**

声明：

Device.stream_on()

意义：

发送开始命令，相机开始传送图像数据

返回值：

None

异常处理：

1) 如果发送开始命令不成功，则抛出异常，异常类型详见错误处理。

- **stream_off**

声明：

Device.stream_off()

意义：

发送停止命令，相机停止传送图像数据

返回值：

None

异常处理：

1) 如果发送停止命令不成功，则抛出异常，异常类型详见错误处理。

- **export_config_file**

声明：

Device.export_config_file(file_path)

意义：

导出当前配置文件

形参：

[in]file_path 文件路径

返回值：

None

异常处理：

1) 如果输入参数不是字符串型，则抛出”ParameterTypeError”异常。
2) 如果导出当前配置文件不成功，则抛出异常，异常类型详见错误处理。

- **import_config_file**

声明：

Device.import_config_file(file_path, verify=False)

意义：

导入配置文件

形参：

[in]file_path　文件路径

[in]verify　　是否所有导入值将被验证一致性，缺省值为 False

返回值：

None

异常处理：

1) 如果输入参数 1 不是字符串型，则抛出”ParameterTypeError”异常。
2) 如果输入参数 2 不是布尔型，则抛出”ParameterTypeError”异常。
3) 如果导入配置文件不成功，则抛出异常，异常类型详见错误处理。

➢ **close_device**

声明：

Device.close_device()

意义：

关闭设备，销毁设备句柄，将句柄置为空

返回值：

None

异常处理：

1) 如果关闭设备不成功，则抛出异常，异常类型详见错误处理。

注意：

当执行关闭设备后，如果还想使用此相机，请重新打开后再操作。

3.4.3. DataStream

负责相机设备的数据流设置、控制，获取图像等。

接口列表：

set_acquisition_buffer_number(buf_num)	设置采集缓冲的大小
get_image(timeout=1000)	获取图像，成功创建图像类对象
flush_queue()	清除相机采集缓冲队列

◆ **接口说明**

➢ **set_acquisition_buffer_number**

声明：

DataStream.set_acquisition_buffer_number(buf_num)

意义：

设置采集缓冲的大小

形参：

[in]buf_num　缓冲区地址的长度[1，0xffffffff]

返回值：

None

异常处理：

1) 如果输入参数不是整型，则抛出”ParameterTypeError”异常。

2) 如果输入参数小于 1 或大于无符号整型的最大值，则打印“DataStream.set_acquisition_buffer_number: buf_num out of bounds, minimum=1, maximum=0xffffffff”，函数返回 None。

3) 如果设置采集缓冲大小不成功，则抛出异常，异常类型详见错误处理。

- **get_image**

声明：

DataStream.get_image(timeout=1000)

意义：

获取图像，成功创建图像类对象

形参：

[in]timeout 获取超时[0，0xffffffff]，缺省值为 1000（ms）

返回值：

图像对象： 获取成功

None: 超时

抛出异常： 其他错误

异常处理：

1) 如果输入参数不是整型，则抛出”ParameterTypeError”异常。

2) 如果输入参数小于 0 或大于 0xffffffff，则打印”DataStream.get_image: timeout out of bounds, minimum=0, maximum=0xffffffff”，函数返回 None。

3) 如果获取数据大小不成功，则抛出异常，异常类型详见错误处理。

4) 如果超时导致未获取图像成功，则函数返回 None。

5) 如果未超时但获取图像失败，则打印“status，DataStream，get_image” ，函数返回 None。

- **flush_queue**

声明：

DataStream.flush_queue()

意义：

清除相机采集缓冲队列

异常处理：

1) 如果清除相机采集缓冲队列不成功，则抛出异常，异常类型详见错误处理。

3.4.4. RGBImage

负责 RGB 图像操作。

接口列表：

image_improvement(color_correction_param=0, contrast_lut=None, gamma_lut=None)	图像质量提高

saturation(factor)	图像进行饱和度调节
sharpen(factor)	图像进行锐化处理
get_numpy_array()	将 RGB 数据转换为 numpy 对象
get_image_size()	获取 RGB 数据大小

◆ **接口说明**

➢ **image_improvement**

声明：

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

意义：

图像质量提高

形参：

[in]contrast_lut 对比度 LUT
[in]gamma_lut gamma LUT
[in]color_collect 颜色校正

异常处理：

1) 如果参数 1、2 不是 Buffer 类型或 None，则抛异常 ParameterTypeError。
2) 如果参数 3 不是整型或 None，则抛异常 ParameterTypeError。
3) 如果提高图像质量不成功，则抛出异常 UnexpectedError。
4) 如果参数 1、2、3 都是默认缺省值，则不进行图像质量提升处理，函数退出。

➢ **saturation**

声明：

RGBImage.saturation(factor)

意义：

对 RGB 图像进行饱和度调节

形参：

[in]factor 饱和度调节参数，范围：0 ~ 128，
其中：64：饱和度没有变化；
大于 64：增加饱和度；
小于 64：减小饱和度；
128：饱和度为当前两倍；
0：黑白图像

返回值：

None

异常处理：

1) 如果图像饱和度调节失败，则抛出异常 UnexpectedError。

- **sharpen**

声明：

RGBImage.sharpen()

意义：

对 RGB 图像进行锐化处理

形参：

[in]factor　锐化调节参数，范围：0.1 ~ 5.0

返回值：

None

异常处理：

1) 如果图像锐化处理失败，则抛出异常 UnexpectedError。

- **get_numpy_array**

声明：

RGBImage.get_numpy_array()

意义：

将 RGB 数据转换为 numpy 对象

返回值：

numpy 对象

- **get_image_size**

声明：

RGBImage.get_image_size()

意义：

获取 RGB 数据大小

返回值：

RGB 图的大小

3.4.5. RawImage

负责 Raw 图像操作。

接口列表：

convert(mode, flip=False, valid_bits=DxValidBit.BIT4_11, convert_type=DxBayerConvertType.NEIGHBOUR)	图像格式转换
defective_pixel_correct()	图像坏点校正
get_numpy_array()	将 raw 数据转换为 numpy 对象
get_data()	获取 raw 数据
save_raw(file_path)	保存 raw 图数据
get_status()	获取 raw 图状态
get_width()	获取 raw 图宽度

get_height()	获取 raw 图高度
get_pixel_format()	获取图像像素格式
get_image_size()	获取 raw 图数据大小
get_frame_id()	获取帧 ID
get_timestamp()	获取时间戳

◆ **接口说明**

➢ **convert**

声明：

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

意义：

图像格式转换。

1) 当 mode = 'RAW8'模式时，将 16 位 raw 图转换为 8 位 raw 图，截取的有效位默认为当前像素格式的高 8 位。用户也可通过参数 valid_bits 手动设置有效位。仅支持 10/12bit 的 Raw 图。

2) 当 mode = 'RGB'模式时，将 raw 图转换为 RGB 图。如果输入为 10/12 位 raw 图，先转换为 8 位 raw 图，再转换为 RGB 图。

形参：

[in]mode	'RAW8'：将 16 位 raw 图转换为 8 位 raw 图 'RGB'：将 raw 图转换为 RGB24 图
[in]flip	输出的 RGB 图像是否上下翻转，缺省值为 False，该功能仅支持 mode = 'RGB'模式
[in]valid_bits	有效位数，缺省值为当前像素格式的高 8 位，参考 DxValidBit
[in]convert_type	转换类型 ,缺省值为 DxBayerConvertType.NEIGHBOUR ,参考 DxBayerConvertType，仅对 mode = 'RGB'模式有效

返回值：

RGB 图像对象

异常处理：

1) 如果帧信息状态不成功，则打印错误信息" RawImage.convert:This is a incomplete image" ，函数返回 None。

2) 如果参数 1 不是字符串型，则抛异常 ParameterTypeError。

3) 如果参数 2 不是布尔型，则抛异常 ParameterTypeError。

4) 如果参数 3、4 不是整型，则抛异常 ParameterTypeError。

5) 如果参数 4 不在 DxBayerConvertType，中，则打印：提示参数越界、当前参数所支持的枚举值，函数返回 None。

6) 如果参数 4 不在 DxValidBit 中，则打印：提示参数越界、当前参数所支持的枚举值，函数返回 None。

7) 如果像素不是 8/10/12bit，则打印错误信息" RawImage.convert:This pixel format is not support" ，函数返回 None。

8) 如果参数 1 为 ‘RAW8’ 且参数 2 为 True，则打印错误信息” RawImage.convert:mode = ‘RAW8’ don’t support flip = True“ ，函数返回 None。

9) mode = ‘RAW8’ ，位深不是 10/12bit，则打印错误信息” RawImage.convert: mode="RAW8" only support 10bit and 12bit“ ，函数返回 None。

10) 如果参数 1 不为 ‘RAW8’ 或’ RGB’ ，则打印接口名称和输入的 mode 不在范围内的信息，函数返回 None。

- **defective_pixel_correct**

声明：

RawImage.defective_pixel_correct()

意义：

对 raw 数据进行坏点校正

返回值：

None

异常处理：

1) 如果坏点校正不成功，则抛出异常 UnexpetedError。

- **get_numpy_array**

声明：

RawImage.get_numpy_array()

意义：

将 raw 数据转换为 numpy 对象

返回值：

numpy 对象： 成功

None： 失败

异常处理：

1) 如果帧信息状态不成功，则打印错误信息”RawImage.get_numpy_array:This is a incomplete image”，函数返回 None。

2) 如果像素格式不为 8 位或 16 位，则返回 None。

- **get_data**

声明：

RawImage.get_data()

意义：

获取 raw 数据

返回值：

raw 数据[字符串型]

- **save_raw**

声明：

RawImage.save_raw(file_path)

意义：

保存 raw 图数据

形参：

[in]file_path 文件路径。

例如：file_path = 'raw_image.raw'，则将 raw 图保存到当前工程路径下；file_path = 'E://python_gxiapi/raw_image.raw'，则将 raw 图保存到绝对路径'E://python_gxiapi/'下。

返回值：

None

异常处理：

1) 如果参数不是字符串格式，则抛出异常 ParameterTypeError。
2) 如果保存 raw 图数据未成功，则抛异常 UnexpectedError。

- **get_status**

声明：

RawImage.get_status()

意义：

获取 raw 图状态

返回值：

raw 图状态，数据类型参考 GxFrameStatusList

- **get_width**

声明：

RawImage.get_width()

意义：

获取 raw 图宽度

返回值：

raw 图宽度

- **get_height**

声明：

RawImage.get_height()

意义：

获取 raw 图高度

返回值：

raw 图高度

- **get_pixel_format**

声明：

RawImage.get_pixel_format()

意义：

获取图像像素格式

返回值：

像素格式

- **get_image_size**

声明：

RawImage.get_image_size()

意义：

获取 raw 图数据大小

返回值：

Raw 图的大小

- **get_frame_id**

声明：

RawImage.get_frame_id()

意义：

获取帧 ID

返回值：

帧 ID

- **get_timestamp**

声明：

RawImage.get_timestamp()

意义：

获取时间戳

返回值：

时间戳

3.4.6. Buffer

负责 Buffer 类的操作。Buffer 类将在图像质量提升的部分使用，Utility.get_gamma_lut(gamma)和 Utility.get_contrast_lut(contrast)接口返回的 Buffer 类型对象将作为参数传给 RGBImage.image_improvement(color_correction_param=0, contrast_lut=None, gamma_lut=None)接口。

接口列表：

from_file(file_name)	从文件获取 Buffer 对象

from_string(string_data)	从字符串获取 Buffer 对象
get_data()	返回 Buffer 对象的字符串数据
get_ctype_array()	返回 Buffer 对象的数据数组
get_numpy_array()	返回 Buffer 对象的 numpy 数组
get_length()	返回 Buffer 对象的数据数组长度

◆ **接口说明**

➢ **from_file（静态函数）**

声明：

Buffer.from_file(file_name)

意义：

从文件获取 Buffer 对象

形参：

[in]file_name　文件路径

返回值：

Buffer 对象

➢ **from_string（静态函数）**

声明：

Buffer.from_string(string_data)

意义：

从字符串获取 Buffer 对象

形参：

[in]string_data　　字符串

返回值：

Buffer 对象

➢ **get_data**

声明：

Buffer.get_data()

意义：

返回 Buffer 对象的字符串数据

返回值：

string_data 字符串数据

注：python2.7：返回字符串类型；python3.5：返回 bytes 类型

➢ **get_ctype_array**

声明：

Buffer.get_ctype_array()

意义：

返回 Buffer 对象的数据数组

返回值：

Buffer 对象的数据数组[ctype 类型]

➢ **get_numpy_array**

声明：

Buffer.get_numpy_array()

意义：

返回 Buffer 对象的 numpy 数组

返回值：

Buffer 对象的数据数组[numpy 类型]

➢ **get_length**

声明：

Buffer.get_length()

意义：

返回 Buffer 对象的数据数组长度

返回值：

数据数组长度

3.4.7. Utility

负责参数 gamma 和 contrast 的操作。

接口列表：

get_gamma_lut(gamma=1)	通过 gamma 值获取 gamma 查找表的 Buffer 类型对象
get_contrast_lut(contrast=0)	通过对比度值获取对比度查找表的 Buffer 类型对象

◆ **接口说明**

➢ **get_gamma_lut（静态函数）**

声明：

Utility.get_gamma_lut(gamma=1)（静态函数）

意义：

通过 gamma 值获取 gamma 查找表的 Buffer 类型对象

形参：

[in]gamma　　整型或浮点型，范围[0.1, 10.0]，缺省值为 1

返回值：

gamma 查找表的 Buffer 类型对象

异常处理：

1) 如果输入参数不是整型或浮点型，则抛出”ParameterTypeError”异常。

2) 如果输入参数不在 0.1~10.0 范围内，则打印错误信息”Utility.get_gamma_lut:gamma out of bounds, range:[0.1, 10.0]”，函数返回 None。

3) 如果获取 gamma 查找表失败，则打印接口名称、获取 gamma lut 失败和错误码的信息，函数返回 None。

- **get_contrast_lut（静态函数）**

声明：

Utility.get_contrast_lut(contrast=0)（静态函数）

意义：

通过对比度值获取对比度查找表的 Buffer 类型对象

形参：

[in]contrast 整型，范围[-50, 100]，缺省值为 0

返回值：

对比度查找表的 Buffer 类型对象

异常处理：

1) 如果输入参数不是整型，则抛出”ParameterTypeError”异常。

2) 如果输入参数不在-50~100 范围内，则打印错误信息”Utility.get_contrast_lut:contrast out of bounds, range:[-50, 100]”，函数返回 None。

3) 如果获取对比度查找表失败，则打印接口名称、获取 constrast lut 失败和错误码的信息，函数返回 None。

4. 常见问题解答

序号	常见问题	解决办法
1	程序运行中出现如下错误”NotInitApi: DeviceManager.update_device_list:{-13}{Not init API}”	1) 请检查并删除程序中调用 DeviceManager 类对象的__del__()函数的语句。因为 Python 的垃圾回收机制会自动调用__del__()函数销毁对象，所以不需要、不允许用户显示调用__del__()函数，如：” device_manager.__del__()” 。

5. 版本说明

序号	修订版本号	所做改动	发布日期
1	V1.0.0	初始发布	2018-08-10
2	V1.0.1	添加水星二代相机新增功能说明	2018-10-31
3	V1.0.2	修改部分标题，更正了部分不准确的描述	2019-04-12
4	V1.0.3	补充了部分描述	2019-05-07