

# C++ 软件开发说明书

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。

本手册的版权属于中国大恒（集团）有限公司北京图像视觉技术分公司所有。未得到本公司的正式许可，任何组织或个人均不得以任何手段和形式对本手册内容进行复制或传播。

本手册的内容若有任何修改，恕不另行通知。

© 2024 中国大恒（集团）有限公司北京图像视觉技术分公司版权所有

网 站: [www.daheng-imaging.com](http://www.daheng-imaging.com)

公 司 总 机: 010-82828878

客户服务热线: 400-999-7595

销 售 信 箱: [sales@daheng-imaging.com](mailto:sales@daheng-imaging.com)

支 持 信 箱: [support@daheng-imaging.com](mailto:support@daheng-imaging.com)

# 目录

- 1. 简介..... 1
- 2. 编程向导 ..... 2
  - 2.1. 搭建编程环境 .....2
    - 2.1.1. 库的选择..... 2
    - 2.1.2. 配置引用头文件..... 2
    - 2.1.3. 配置 lib 文件 ..... 2
  - 2.2. 调试千兆网相机注意事项 .....4
  - 2.3. 初始化和反初始化.....5
  - 2.4. 错误处理 .....5
  - 2.5. 枚举设备 .....5
  - 2.6. 获取 Interface 信息.....6
  - 2.7. 获取 Interface 对象.....6
  - 2.8. 配置相机 IP 地址.....7
  - 2.9. 打开关闭设备 .....7
  - 2.10. 属性控制 .....9
    - 2.10.1. 属性控制器种类..... 9
    - 2.10.2. 属性数据类型 ..... 10
    - 2.10.3. 属性访问类型 ..... 10
    - 2.10.4. 从哪里获取相机属性参数 .....11
    - 2.10.5. 从哪里获取属性读写的示例代码 .....11
    - 2.10.6. 不同类型属性示例代码..... 12
    - 2.10.7. 注意事项..... 14
  - 2.11. 采集控制与图像处理.....15
    - 2.11.1. 零拷贝采单帧 (DQBuf) ..... 15
    - 2.11.2. 采单帧 (GetImage) ..... 16
    - 2.11.3. 回调采集..... 17
    - 2.11.4. 设置采集 buffer 个数..... 19
    - 2.11.5. 图像处理..... 20
    - 2.11.6. 流对象属性控制 ..... 25
    - 2.11.7. 注意事项..... 26

2.12. 获取设备事件 .....	27
2.12.1. 选择事件.....	27
2.12.2. 使能事件.....	27
2.12.3. 注册事件通知回调函数.....	27
2.12.4. 获取事件数据信息 .....	27
2.12.5. 代码样例.....	28
2.13. 获取掉线通知 .....	29
2.14. 导入导出相机配置参数.....	30
<b>3. 智能指针对象类型定义 .....</b>	<b>32</b>
<b>4. 数据类型定义 .....</b>	<b>33</b>
4.1. GX_STATUS_LIST .....	33
4.2. GX_DEVICE_CLASS_LIST.....	33
4.3. GX_ACCESS_STATUS.....	33
4.4. GX_ACCESS_MODE .....	34
4.5. GX_PIXEL_FORMAT_ENTRY .....	34
4.6. GX_FRAME_STATUS_LIST.....	37
4.7. GX_FEATURE_TYPE .....	37
4.8. GX_BAYER_CONVERT_TYPE_LIST .....	37
4.9. GX_VALID_BIT_LIST .....	37
4.10. GX_IP_CONFIGURE_MODE_LIST .....	38
4.11. GX_RESET_DEVICE_MODE.....	38
4.12. COLOR_TRANSFORM_FACTOR.....	38
4.13. GX_TL_TYPE_LIST .....	38
4.14. GX_ACTUAL_BITS .....	38
4.15. GX_PIXEL_COLOR_FILTER .....	39
4.16. GX_LOG_TYPE_LIST .....	39
4.17. GX_STATIC_DEFECT_CORRECTION.....	39
4.18. GX_IMAGE_INFO .....	39
<b>5. 句柄类型定义 .....</b>	<b>40</b>
<b>6. 回调事件虚基类.....</b>	<b>41</b>
<b>7. 模块接口定义 .....</b>	<b>42</b>
7.1. IGXFactory .....	42
7.1.1. GetInstance .....	42
7.1.2. Init .....	42
7.1.3. Uninit.....	43
7.1.4. UpdateDeviceList.....	43
7.1.5. UpdateAllDeviceList .....	43

7.1.6. UpdateAllDeviceListEx .....	43
7.1.7. OpenDeviceByIP/MAC/SN/UserID .....	43
7.1.8. GigEIpConfiguration .....	43
7.1.9. GigEForcelp .....	44
7.1.10. GigEResetDevice .....	44
7.1.11. CreateImageFormatConvert .....	45
7.1.12. CreateImageSpecialFunction .....	45
7.1.13. GetAllInterfaceInfo .....	45
7.1.14. GetInterface .....	45
7.1.15. CreateImageProcess .....	45
7.1.16. SetLogType .....	45
7.1.17. GetLogType .....	45
7.2. IGXInterfaceInfo .....	45
7.2.1. GetType .....	46
7.2.2. GetModelName .....	46
7.2.3. GetDisplayName .....	46
7.2.4. GetSerialNumber .....	46
7.2.5. GetInterfaceID .....	46
7.2.6. GetInitFlag .....	46
7.2.7. GetVendorName .....	46
7.3. IGXInterface .....	46
7.3.1. GetInterfaceInfo .....	47
7.3.2. GetAllDeviceInfo .....	47
7.3.3. GetFeatureControl .....	47
7.4. IGXDeviceInfo .....	47
7.4.1. GetVendorName .....	48
7.4.2. GetModelName .....	48
7.4.3. GetSN .....	48
7.4.4. GetDisplayName .....	48
7.4.5. GetDeviceID .....	48
7.4.6. GetUserID .....	48
7.4.7. GetAccessStatus .....	48
7.4.8. GetDeviceClass .....	48
7.4.9. GetMAC .....	48
7.4.10. GetIP .....	48
7.4.11. GetSubnetMask .....	49
7.4.12. GetGateway .....	49
7.4.13. GetNICMAC .....	49
7.4.14. GetNICIP .....	49
7.4.15. GetNICSubnetMask .....	49
7.4.16. GetNICGateway .....	49
7.4.17. GetNICDescription .....	49
7.5. IGXDevice .....	49
7.5.1. GetDeviceInfo .....	50
7.5.2. GetStreamCount .....	50
7.5.3. OpenStream .....	50

7.5.4. GetFeatureControl .....	50
7.5.5. GetRemoteFeatureControl .....	50
7.5.6. GetEventNumInQueue .....	50
7.5.7. FlushEvent.....	50
7.5.8. RegisterDeviceOfflineCallback.....	50
7.5.9. UnregisterDeviceOfflineCallback.....	51
7.5.10. CreateImageProcessConfig .....	51
7.5.11. ExportConfigFile .....	51
7.5.12. ImportConfigFile .....	51
7.5.13. Close.....	51
7.5.14. GetParentInterface .....	51
7.6. IGXFeatureControl .....	51
7.6.1. GetFeatureNameList .....	52
7.6.2. GetFeatureType.....	52
7.6.3. IsImplemented .....	52
7.6.4. IsReadable.....	52
7.6.5. IsWritable .....	52
7.6.6. GetIntFeature.....	53
7.6.7. GetFloatFeature.....	53
7.6.8. GetEnumFeature .....	53
7.6.9. GetBoolFeature .....	53
7.6.10. GetStringFeature .....	53
7.6.11. GetCommandFeature .....	53
7.6.12. GetRegisterFeature .....	53
7.6.13. RegisterFeatureCallback.....	53
7.6.14. UnregisterFeatureCallback.....	53
7.6.15. ReadPort.....	53
7.6.16. WritePort.....	54
7.6.17. ReadPortStacked .....	54
7.6.18. WritePortStacked.....	54
7.6.19. FeatureSave .....	54
7.6.20. FeatureSaveW.....	54
7.6.21. FeatureLoad .....	54
7.6.22. FeatureLoadW.....	54
7.7. IIntFeature .....	55
7.7.1. GetMin .....	55
7.7.2. GetMax .....	55
7.7.3. GetInc .....	55
7.7.4. GetValue .....	55
7.7.5. SetValue.....	55
7.8. IFloatFeature .....	55
7.8.1. GetMin .....	56
7.8.2. GetMax .....	56
7.8.3. HasInc.....	56
7.8.4. GetInc .....	56
7.8.5. GetUnit.....	56

7.8.6. GetValue .....	56
7.8.7. SetValue.....	56
7.9. IEnumFeature.....	56
7.9.1. GetEnumEntryList.....	56
7.9.2. GetValue .....	57
7.9.3. SetValue.....	57
7.10. IBoolFeature .....	57
7.10.1. GetValue .....	57
7.10.2. SetValue.....	57
7.11. IStringFeature .....	57
7.11.1. GetValue .....	57
7.11.2. SetValue.....	57
7.11.3. GetStringMaxLength .....	58
7.12. ICommandFeature.....	58
7.12.1. Execute.....	58
7.13. IRegisterFeature.....	58
7.13.1. GetLength.....	58
7.13.2. GetBuffer.....	58
7.13.3. SetBuffer .....	58
7.14. IGXStream.....	58
7.14.1. StartGrab .....	59
7.14.2. StopGrab .....	59
7.14.3. RegisterCaptureCallback.....	59
7.14.4. UnregisterCaptureCallback .....	59
7.14.5. GetImage .....	59
7.14.6. DQBuf .....	60
7.14.7. QBuf.....	60
7.14.8. GetFeatureControl.....	60
7.14.9. FlushQueue .....	60
7.14.10. SetAcquisitionBufferNumber .....	60
7.14.11. Close.....	60
7.14.12. GetOptimalPacketSize .....	61
7.14.13. GetPayloadSize.....	61
7.15. IImageData.....	61
7.15.1. GetStatus.....	61
7.15.2. GetPayloadSize.....	61
7.15.3. GetWidth.....	62
7.15.4. GetHeight.....	62
7.15.5. GetPixelFormat.....	62
7.15.6. GetFrameID .....	62
7.15.7. GetTimeStamp.....	62
7.15.8. GetBuffer.....	62
7.15.9. ConvertToRaw8 .....	62
7.15.10. ConvertToRGB24 .....	62
7.15.11. ImageProcess .....	62
7.16. IImageProcessConfig .....	63
7.16.1. SetValidBit.....	64

7.16.2. GetValidBit .....	64
7.16.3. EnableDefectivePixelCorrect.....	64
7.16.4. IsDefectivePixelCorrect .....	64
7.16.5. EnableSharpen .....	64
7.16.6. IsSharpen .....	64
7.16.7. EnableAccelerate .....	64
7.16.8. IsAccelerate .....	64
7.16.9. SetSharpenParam .....	65
7.16.10. GetSharpenParam.....	65
7.16.11. SetContrastParam .....	65
7.16.12. GetContrastParam.....	65
7.16.13. SetGammaParam .....	65
7.16.14. GetGammaParam .....	65
7.16.15. SetLightnessParam .....	65
7.16.16. GetLightnessParam.....	65
7.16.17. EnableDenoise .....	65
7.16.18. IsDenoise.....	65
7.16.19. SetSaturationParam .....	65
7.16.20. GetSaturationParam.....	66
7.16.21. SetConvertType .....	66
7.16.22. GetConvertType .....	66
7.16.23. EnableConvertFlip .....	66
7.16.24. IsConvertFlip.....	66
7.16.25. EnableColorCorrection .....	66
7.16.26. IsColorCorrection.....	66
7.16.27. Reset.....	66
7.16.28. IsUserSetCCParam .....	66
7.16.29. EnableUserSetCCParam.....	66
7.16.30. SetUserCCParam .....	67
7.16.31. GetUserCCParam .....	67
7.17. ICaptureEventHandler .....	67
7.17.1. DoOnImageCaptured .....	67
7.18. IDeviceOfflineEventHandler .....	67
7.18.1. DoOnDeviceOfflineEvent.....	67
7.19. IFeatureEventHandler .....	67
7.19.1. DoOnFeatureEvent.....	67
7.20. IGXImageFormatConvert .....	68
7.20.1. 支持转换类型 .....	68
7.20.2. SetDstFormat.....	69
7.20.3. GetDstFormat .....	69
7.20.4. SetInterpolationType.....	69
7.20.5. GetInterpolationType .....	70
7.20.6. SetAlphaValue .....	70
7.20.7. GetAlphaValue .....	70
7.20.8. GetBufferSizeForConversion.....	70
7.20.9. GetBufferSizeForConversion.....	70



7.20.10. Convert .....	70
7.20.11. Convert.....	71
7.20.12. SetValidBits.....	71
7.20.13. GetValidBits .....	71
7.20.14. Set3DCalibParam .....	71
7.20.15. Get3DCalibParam .....	71
7.20.16. SetYStep.....	72
7.20.17. GetYStep .....	72
7.21. IGXImageProcess .....	72
7.21.1. StaticDefectCorrection.....	72
7.21.2. ImageImprovment.....	72
7.21.3. ImageImprovment.....	73
7.21.4. CalculaLut.....	73
7.21.5. ReadLutFile .....	74
7.22. IGXImageSpecialFunction (不再维护) .....	74
7.22.1. CalculaLut.....	74
7.22.2. ReadLutFile .....	75
<b>8. 示例工程 .....</b>	<b>76</b>
<b>9. 版本说明 .....</b>	<b>80</b>

## 1. 简介

GxIAPICPPEx.dll 库是基于面向对象思想封装的通用并且统一的编程接口，适用于大恒图像 MERCURY、MARS 等系列相机。

下面介绍主要工作模块以及主要功能职责，见表 1-1：

主要工作模块	主要功能职责
IGXFactory	初始化接口库，枚举设备、打开设备
IGXDevice	设备对象，以此对象为入口进行属性控制、图像采集、获取相机事件等
IGXStream	流对象，从 IGXDevice 获得，专门负责图像采集相关职能
IGXFeatureControl	属性控制对象，可以分别从 IGXDevice 和 IGXStream 获取的属性控制对象，负责各种的属性控制
IIImageData	回调采集和采单帧的图像结构体，包含采集输出结果：图像 buffer 和图像信息等，还自带图像格式转换、图像效果增强等功能
IGXInterface	Interface 功能指针，可以枚举后通过 GetInterface()获取，包含获取 Interface 层信息指针、获取 Interface 层属性控制器指针、获取 Interface 层下所有设备列表等功能。

表 1-1 主要工作模块及主要功能职责介绍表

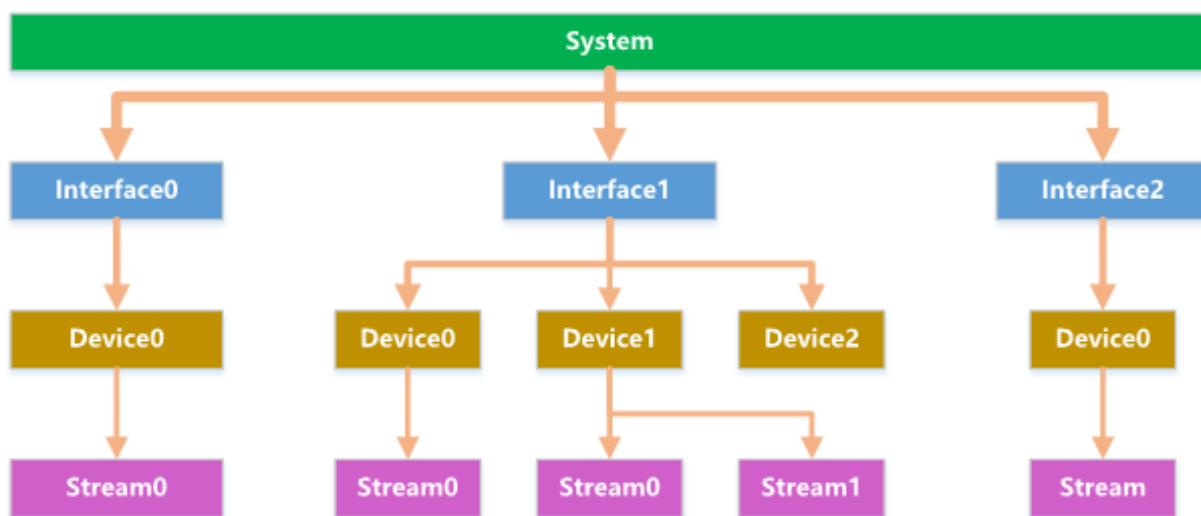


图 1-1 模块分层结构图

- 对编程环境的支持情况

Microsoft Visual Studio 2005  
 Microsoft Visual Studio 2008  
 Microsoft Visual Studio 2010  
 Microsoft Visual Studio 2012  
 Microsoft Visual Studio 2013

Microsoft Visual Studio 2015  
 Microsoft Visual Studio 2017  
 Microsoft Visual Studio 2019  
 Microsoft Visual Studio 2022

## 2. 编程向导

### 2.1. 搭建编程环境

以 VS2008 平台为例,编译开发需要头文件和 lib 文件,请参见安装目录下面的 Sample 程序工程配置。

#### 2.1.1. 库的选择

由于 GxIAPICPP 库存在编译器的兼容性问题,在 2019 年以后新发布的 GxIAPICPP 库升级为 GxIAPICPPEX,旧的 GxIAPICPP 库依然保留(但不再发布其 Lib 文件),不会影响您已经编译好的程序正常运行,但是如果您需要重新编译或者新开发程序,请使用 GxIAPICPPEX 库进行开发,配置过程如下。

#### 2.1.2. 配置引用头文件

在解决方案资源管理窗口中选中用户创建的工程,然后点击菜单中的 project->properties 弹出 Property page 窗口。选择 Configuration Properties->C/C++->General 在 Additional Include Directories 中填写 GalaxyIncludes.h 所在目录路径地址(依用户安装目录为准),如图 2-1 所示:

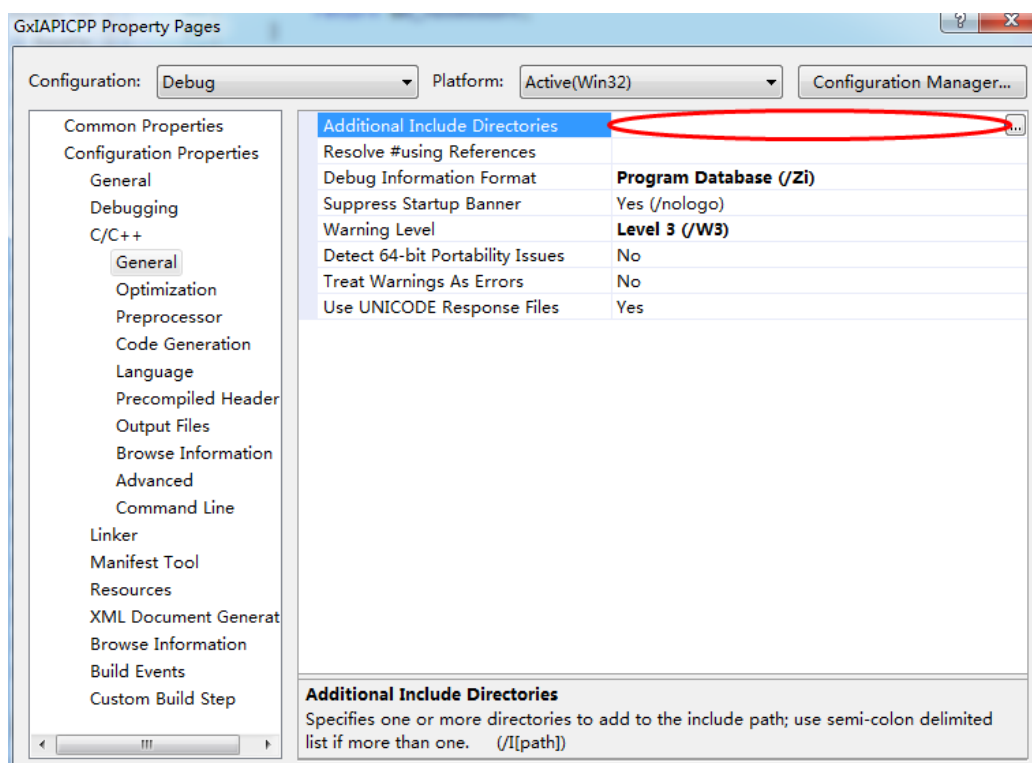


图 2-1 配置引用头文件

#### 2.1.3. 配置 lib 文件

然后选择 Configuration Properties->Linker->General 在 Additional Library Directories 中填写 GxIAPICPPEX.lib 所在目录路径地址(依用户安装目录为准),如图 2-2 所示:

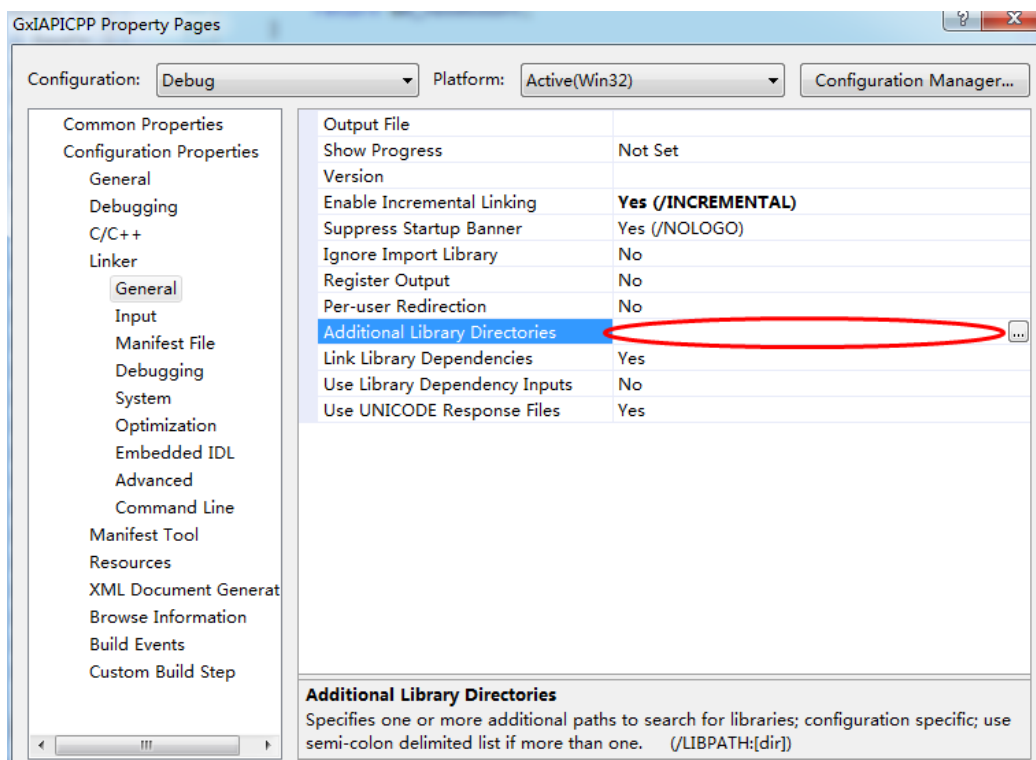


图 2-2 配置 lib 文件路径

然后选择 Configuration Properties->Linker->Input 在 Additional Dependencies 中填写 GxIAPICPPEx.lib, 如图 2-3 所示:

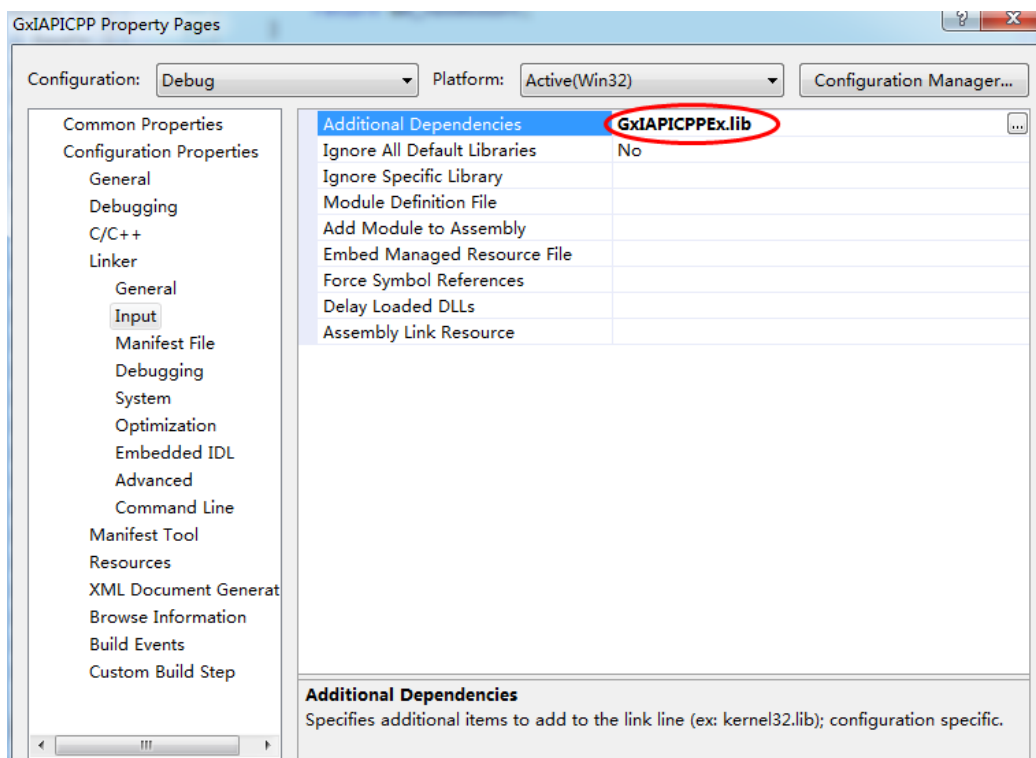


图 2-3 配置引用 lib 的名称

## 2.2. 调试千兆网相机注意事项

Windows 的用户在使用 Visual Studio 开发平台在 Debug 模式下调试千兆网相机的时候可能会遇到因为心跳超时而导致设备掉线的情况。应用程序必须以固定的时间间隔发送心跳包到设备，如果设备没有接收到心跳包则会认为当前连接已经断开，从而不会再接收从应用程序发送的任何命令。

当用户正常运行应用程序的时候，底层库会正常发送心跳包，保持和设备的连接状态，但是当用户在应用程序中设置断点调试的时候，程序运行到这些断点时，调试器会暂停所有线程，包括发送心跳包的线程，所以当用户在 Debug 模式下单步调试代码的时候，就不会发送心跳包到设备。

用户可以通过增加心跳超时时间的方式来解决这个问题，用户修改心跳超时时间有以下两种方式。

1) 第一种方式，在程序中打开设备后添加如下代码：

**代码使用宏编译条件判断包含，只有在 Debug 调试模式才编译此段代码，切忌不要将此段代码带到正常发布产品中。**

```
//打开设备，此处以序列号为例打开设备，序列号以实际设备为准。用户可以选择其他方式打开设备。  
#ifdef _DEBUG  
CGXDevicePointer objDevicePtr = IGXFactory::GetInstance().OpenDeviceBySN(  
    "RN0001007012", GX_ACCESS_EXCLUSIVE);  
  
//获取远端设备属性控制器  
CGXFeatureControlPointer objFeatureControl =  
    objDevicePtr->GetRemoteFeatureControl();  
  
//设置心跳超时时间 5 分钟  
objFeatureControl->GetIntFeature("GevHeartbeatTimeout")->SetValue(300000);  
#endif
```

2) 第二种方式，在系统中添加环境变量"MER\_HEARTBEAT\_TIMEOUT"，并赋给一个大于零的值，添加完成后使用应用程序打开设备，设备的心跳超时时间就会自动变成此环境变量的值。注意只需要在开发的系统上添加此环境变量，因为无论在 Debug 还是 Release 的应用程序中都起作用。

**注意：**如果用户将心跳超时时间设置的非常长，当结束程序的时候没有调用关闭设备接口来关闭设备，这就会导致设备在心跳时间内无法复位，从而导致用户再次尝试打开设备的时候失败。可通过复位或重连设备解决此问题，有两种方式实现设备的复位或重连：

- 1) 在 IP 配置工具中直接选择复位设备或重连设备；
- 2) 通过接口 GXGigEResetDevice 实现设备的复位/重连。

复位设备：等同于给设备掉电上电一次，相机内程序全部重新加载。

重连设备：等同于软件接口关闭设备，执行此操作后，允许用户重新打开设备。

## 2.3. 初始化和反初始化

1) GxIAPICPPEx 库在使用之前必须执行初始化。在调用其他接口之前必须调用 IGXFactory::GetInstance().Init()接口执行初始化。

2) 代码样例:

```
//使用其他接口之前，必须执行初始化
IGXFactory::GetInstance().Init();
```

3) 在进程退出之前必须调用 IGXFactory::GetInstance().Uninit()接口释放 GxIAPICPPEx 申请的所有资源。

4) 代码样例:

```
//关闭设备之后，不能再调用其他任何库接口
IGXFactory::GetInstance().Uninit();
```

## 2.4. 错误处理

当 GxIAPICPPEx 库检测到内部错误的时候会以异常的形式抛出错误描述信息，异常类型 CGalaxyException，其继承自 runtime\_error。用户可以调用 CGalaxyException::GetErrorCode()获取错误码，错误码种类参见 [GX STATUS LIST](#)；用户还可以访问异常类的属性 CGalaxyException::what()获取详细的错误描述信息字符串。

错误处理代码样例:

```
try
{
    IGXFactory::GetInstance().Init();
}
catch (CGalaxyException &e)
{
    cout <<"错误码: "<< e.GetErrorCode() <<endl;
    cout <<"错误描述信息: "<< e.what() <<endl;
}
```

## 2.5. 枚举设备

用户通过调用 IGXFactory::GetInstance().UpdateDeviceList、IGXFactory::GetInstance().UpdateAllDeviceListEx 枚举当前所有可用设备，获取一个设备信息列表，列表类型为 GxIAPICPP::gxdeviceinfo\_vector。

下面为两个函数如何调用的伪代码:

```
GxIAPICPP::gxdeviceinfo_vector vecDeviceInfo;
IGXFactory::GetInstance().UpdateDeviceList(1000, vecDeviceInfo);
for (uint32_t i = 0; i < vecDeviceInfo.size(); i++)
{
    cout << vecDeviceInfo[i].GetVendorName() <<endl;
    cout << vecDeviceInfo[i].GetModelName() <<endl;
}
```

```
//更多设备信息详见 IGXDeviceInfo 接口
}
GxIAPICPP::gxdeviceinfo_vector vecDeviceInfo;
IGXFactory::GetInstance().UpdateAllDeviceListEx(GX_TL_TYPE_USB |
GX_TL_TYPE_GEV | GX_TL_TYPE_U3V | GX_TL_TYPE_CXP, 1000, vecDeviceInfo);
for (uint32_t i = 0; i < vecDeviceInfo.size(); i++)
{
    cout << vecDeviceInfo[i].GetVendorName() <<endl;
    cout << vecDeviceInfo[i].GetModelName() <<endl;
    //更多设备信息详见 IGXDeviceInfo 接口
}
```

**注意：**除上面的枚举接口，GxIAPICPPEx 还提供了另外一个枚举接口 IGXFactory::GetInstance().UpdateAllDeviceList。这三个枚举接口对于非千兆网相机来说功能上是一样的，对千兆网相机来说，库内部使用的枚举机制不一样：

UpdateAllDeviceList：使用全网枚举

UpdateDeviceList：使用子网枚举

UpdateAllDeviceListEx：使用全网枚举

## 2.6. 获取 Interface 信息

用户通过调用 IGXFactory::GetInstance().GetAllInterfaceInfo()接口获取 Interface 的信息，参数类型为 GxIAPICPP::gxinterfaceinfo\_vector。

样例代码：

```
//获取 Interface 信息
GxIAPICPP::gxinterfaceinfo_vector vectorInterfaceInfo;
IGXFactory::GetInstance().GetAllInterfaceInfo(vectorInterfaceInfo);
for (uint32_t i = 0; i < vectorInterfaceInfo.size(); i++)
{
    cout << vectorInterfaceInfo[i].GetVendorName() <<endl;
    cout << vectorInterfaceInfo[i].GetModelName() <<endl;
    //更多设备信息详见 IGXInterfaceInfo 接口
}
```

## 2.7. 获取 Interface 对象

用户通过调用 IGXFactory::GetInstance().GetInterface() 接口获取 Interface 实例化对象，参数类型为整型，下标从 1 开始计数。

样例代码：

```
//获取 Interface 指针
CGXInterfacePointer objInterfacePointer = NULL;
objInterfacePointer = IGXFactory::GetInstance().GetInterface(1);
```

## 2.8. 配置相机 IP 地址

用户通过调用：`IGXFactory::GetInstance().GigEIpConfiguration`、`IGXFactory::GetInstance().GigEForceIp` 这两种不同的方式设置相机的 IP 地址。

其中使用 `GigEIpConfiguration` 函数可以设置相机静态（永久）IP 地址，并且提供四种设置方式：直接写入静态 IP 地址、使用 DHCP 服务器分配 IP 地址、以 LLA(Link-Local Address)方式配置相机 IP 和以默认方式设置 IP。当选用默认方式时，相机内部会将其他三种配置方式全部使能，但优先选用静态 IP 方式配置相机 IP 地址。

使用 `GigEForceIp` 函数可以对相机执行 `ForceIp` 操作。`ForceIp` 是指设置的 IP 地址只对本次使用有效，当相机掉电重启后会恢复原来的 IP 地址。

代码样例：

```
//示例 MAC 地址，实际相机 MAC 地址可以通过枚举操作获得
GxIAPICPP::gxstring strMAC= "00-21-49-00-00-00";
GxIAPICPP::gxstring strIPAddress= "192.168.10.10";
GxIAPICPP::gxstring strSubnetMask= "255.255.255.0";
GxIAPICPP::gxstring strDefaultGateway= "192.168.10.2";
GxIAPICPP::gxstring strUserID= "Daheng Imaging";
GX_IP_CONFIGURE_MODE emIpConfigureMode = IP_CONFIGURE_STATIC_IP;

IGXFactory::GetInstance().GigEIpConfiguration(strMAC, emIpConfigureMode,
                                                strIPAddress, strSubnetMask,
                                                strDefaultGateway, strUserID);
```

```
//示例 MAC 地址，实际相机 MAC 地址可以通过枚举操作获得
GxIAPICPP::gxstring strMAC= "00-21-49-00-00-00";
GxIAPICPP::gxstring strIPAddress= "192.168.10.10";
GxIAPICPP::gxstring strSubnetMask= "255.255.255.0";
GxIAPICPP::gxstring strDefaultGateway= "192.168.10.2";

//ForceIp
IGXFactory::GetInstance().GigEForceIp(strMAC, strIPAddress,
                                       strSubnetMask, strDefaultGateway);
```

**注意事项：**

- 1) 调用这两个接口之前，必须先进行枚举操作，并且执行此操作时相机不能处于打开状态；
- 2) 用户自定义名称（UserID）允许的最大长度是 16 个字符。

## 2.9. 打开关闭设备

用户通过调用：

```
IGXFactory::GetInstance().OpenDeviceBySN
IGXFactory::GetInstance().OpenDeviceByUserID
IGXFactory::GetInstance().OpenDeviceByMAC
IGXFactory::GetInstance().OpenDeviceByIP
```



这四种不同的方式打开设备，其中：

- 1) SN 为设备序列号。
- 2) UserID 为用户自定义名称（不支持 UserID 的设备此项为空字符串）。
- 3) MAC 为设备 MAC 地址（非千兆网相机此项为空字符串）。
- 4) IP 为设备 IP 地址（非千兆网相机此项为空字符串）。

强烈建议用户在打开设备之前先调用枚举设备接口，更新库内部的设备列表，否则可能打开设备失败。

代码样例：

```
GxIAPICPP::gxdeviceinfo_vector vectorDeviceInfo;
IGXFactory::GetInstance().UpdateDeviceList(1000, vectorDeviceInfo);

if (vectorDeviceInfo.size() > 0)
{
    //打开链表中的第一个设备
    CGXDevicePointer objDevicePtr;
    GxIAPICPP::gxstring strSN = vectorDeviceInfo[0].GetSN();
    GxIAPICPP::gxstring strUserID = vectorDeviceInfo[0].GetUserID();
    GxIAPICPP::gxstring strMAC = vectorDeviceInfo[0].GetMAC();
    GxIAPICPP::gxstring strIP = vectorDeviceInfo[0].GetIP();

    //用户也可以直接指定打开的设备信息，下面代码中使用的信息为伪造信息，用户以实际设备为准
    //GxIAPICPP::gxstring strSN = "GA0140100002";
    //GxIAPICPP::gxstring strUserID = "MyUserName";
    //GxIAPICPP::gxstring strMAC = "A1-0B-32-7C-6F-81";
    //GxIAPICPP::gxstring strIP = "192.168.0.100";
    objDevicePtr = IGXFactory::GetInstance().OpenDeviceBySN(strSN, GX_ACCESS_EXCLUSIVE);
    //objDevicePtr = IGXFactory::GetInstance().OpenDeviceByUserID(strUserID,
    //GX_ACCESS_EXCLUSIVE);
    //objDevicePtr = IGXFactory::GetInstance().OpenDeviceByMAC(strMAC,
    //GX_ACCESS_EXCLUSIVE);
    //objDevicePtr = IGXFactory::GetInstance().OpenDeviceByIP(strIP,
    //GX_ACCESS_EXCLUSIVE);
}
```

**注意：**对于千兆网设备，由于网络机制允许可以通过 IP 地址直接建立连接，所以可以不枚举直接打开千兆网设备。

用户可以调用 IGXDevice::Close 接口来关闭设备，释放所有设备资源。

代码样例：

```
//关闭设备之后不允许再调用 IDevice 以及设备的 IFeatureControl、IStream 的所有接口
objDevicePtr->Close();
```

**注意：**设备关闭后，与设备相关的所有资源都会被释放，包括从设备对象获取的 CGXFeatureControlPointer 对象和 CGXStreamPointer 对象，所以关闭设备之后不允许再调用这些对象的接口。

## 2.10. 属性控制

### 2.10.1. 属性控制器种类

分四种属性控制器，如下：

- 1) IGXFeatureControl IGXDevice::GetRemoteFeatureControl  
//包含主要设备信息，比如宽高、曝光增益等，一般用户主要使用此属性控制器即可。
- 2) IGXFeatureControl IGXDevice::GetFeatureControl  
//包含一些本地属性，不同类型的设备具备的功能也不一样。
- 3) IGXFeatureControl IGXStream::GetFeatureControl  
//流对象属性控制器，关于采集控制和采集数据统计的属性访问控制器。
- 4) IGXFeatureControl IGXInterface::GetFeatureControl  
//包含主要 Interface 信息，不同类型的 Interface 具备的功能也不一样。

通过 GalaxyView 演示程序打开设备，见右侧属性控制树，在设备属性页中分三个区间，上面区间对应功能是 IGXDevice::GetRemoteFeatureControl 返回的属性集合（图 2-4），中间区间对应功能是 IGXDevice::GetFeatureControl 返回属性集合（图 2-4），下面区间对应功能是 IGXStream::GetFeatureControl 返回的属性集合（图 2-4）。

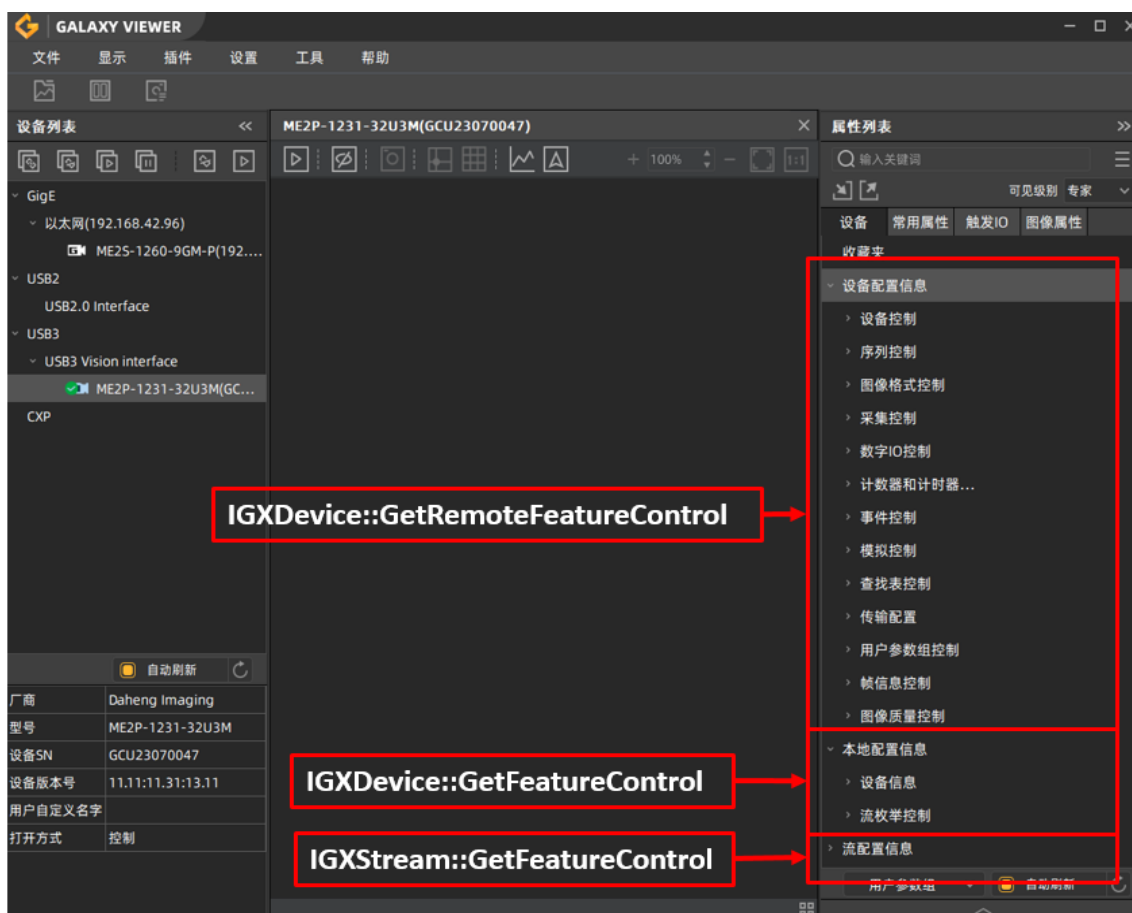


图 2-4 远端、本地和流属性集合

通过 GalaxyView 演示程序点击 Interface 节点，见右侧属性控制树，在属性页对应功能是 IGXInterface::GetFeatureControl 返回的属性集合（图 2-5）。

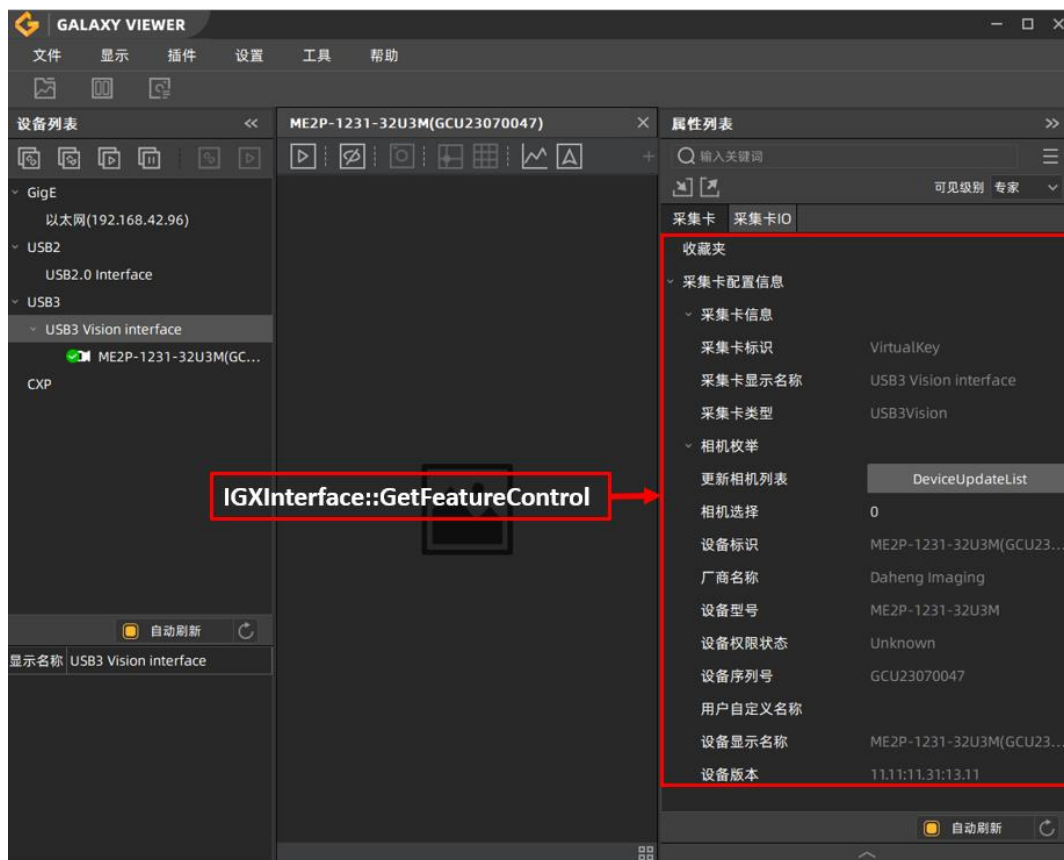


图 2-5 Interface 属性集合

### 2.10.2. 属性数据类型

分为七种数据类型，如下：

- |                            |                                       |         |
|----------------------------|---------------------------------------|---------|
| 1) CIntFeaturePointer      | IGXFeatureControl::GetIntFeature      | //整型    |
| 2) CFloatFeaturePointer    | IGXFeatureControl::GetFloatFeature    | //浮点型   |
| 3) CBoolFeaturePointer     | IGXFeatureControl::GetBoolFeature     | //布尔型   |
| 4) CEnumFeaturePointer     | IGXFeatureControl::GetEnumFeature     | //枚举型   |
| 5) CStringFeaturePointer   | IGXFeatureControl::GetStringFeature   | //字符串   |
| 6) CCommandFeaturePointer  | IGXFeatureControl::GetCommandFeature  | //命令类型  |
| 7) CRegisterFeaturePointer | IGXFeatureControl::GetRegisterFeature | //寄存器类型 |

### 2.10.3. 属性访问类型

分为三种访问类型，如下：

- |         |                                  |                  |
|---------|----------------------------------|------------------|
| 1) bool | IGXFeatureControl::IsImplemented | //当前属性控制器是否支持此功能 |
| 2) bool | IGXFeatureControl::IsReadable    | //此功能是否可读        |
| 3) bool | IGXFeatureControl::IsWritable    | //此功能是否可写        |

#### 2.10.4. 从哪里获取相机属性参数

从接口获取：IGXFeatureControl::GetFeatureNameList，接口返回当前属性控制支持的功能名称字符串。

从演示程序 GalaxyView 获取，使用 GalaxyView 打开设备，查看右侧属性控制树，控制树中的每个控制节点的英文名称。如图 2-6 所示。

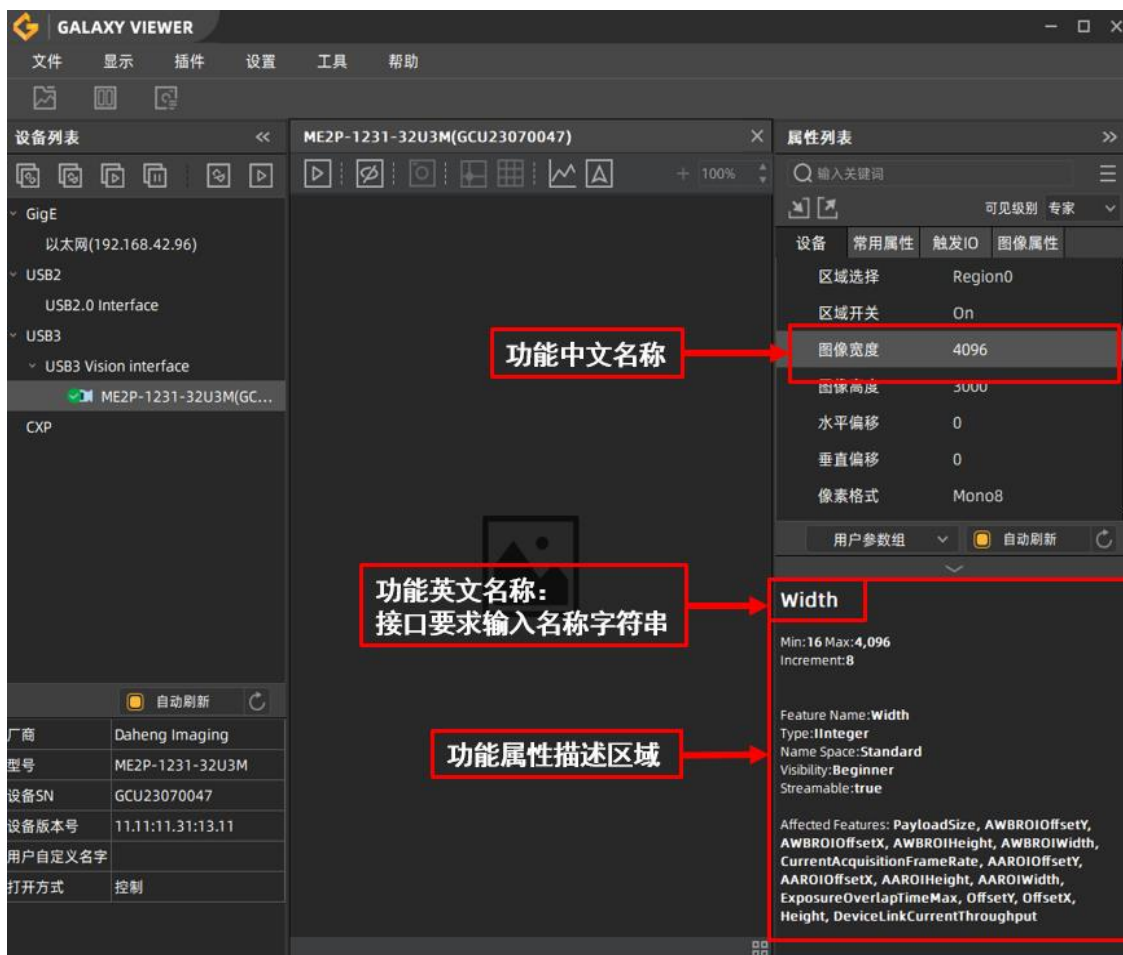


图 2-6 获取相机属性参数

如图 2-6，当鼠标选中“图像宽度”功能的时候，在“功能属性描述区域”内部会显示其功能名称、最大值、最小值、步长等信息。

**注意：**功能名称字符串大小写敏感。

#### 2.10.5. 从哪里获取属性读写的示例代码

为了方便用户开发，演示程序提供了属性读写的示例代码。在演示程序菜单栏：视图 -> 属性文档，即可打开该窗口，如下图所示：

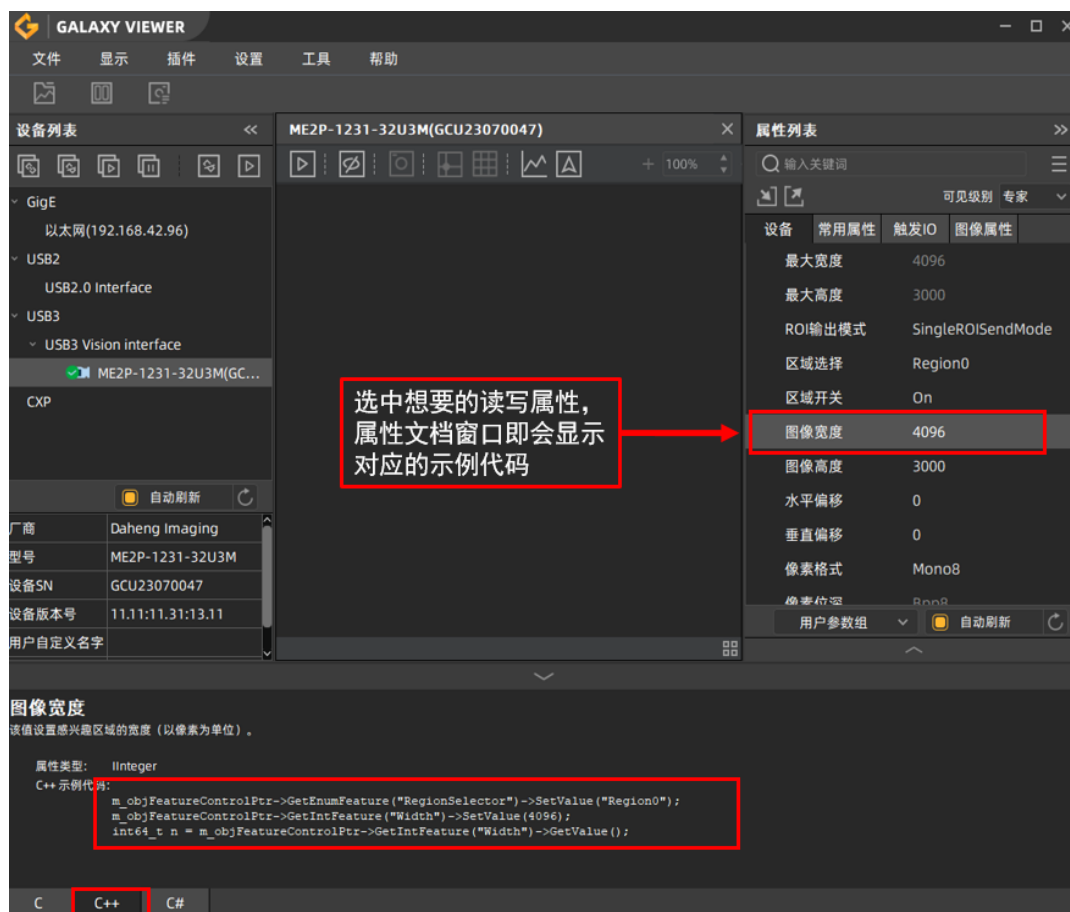


图 2-7

目前，属性文档功能提供了三种开发语言的示例程序，切换到 C++ Tab 即可获得 C++语言读写用户指定属性的代码，该代码可以直接拷贝到用户开发工程中使用。

## 2.10.6. 不同类型属性示例代码

### 2.10.6.1. 读写访问控制

所有功能都是通过功能字符串来控制，建议使用之前先查询当前属性控制器是否支持此功能，然后再进行读写操作。

代码样例：

```
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();

//获取Width节点
bool bIsImplemented = objFeatureControlPtr->IsImplemented("Width");
if (bIsImplemented)
{
    bool bIsReadable = objFeatureControlPtr->IsReadable("Width");//查询是否可读
    bool bIsWritable = objFeatureControlPtr->IsWritable("Width");//查询是否可写
}
```

### 2.10.6.2. 整数控制

按照数据类型，相机功能被分成七大类，整型是其一，包含宽、高等整数控制参数，用户可以通过整数控制器 CIntFeaturePointer 来读写当前值，查询最大值、最小值和步长。

代码样例：

```
CGXFeatureControlPointer objFeatureControlPtr=
    objDevicePtr->GetRemoteFeatureControl();
CIntFeaturePointer objIntPtr =
    objFeatureControlPtr->GetIntFeature("Width");
int64_t i64Max = objIntPtr->GetMax(); //获取最大值
int64_t i64Min = objIntPtr->GetMin(); //获取最小值
int64_t i64Inc = objIntPtr->GetInc(); //获取步长
int64_t i64Value = objIntPtr->GetValue(); //获取当前值
objIntPtr->SetValue(i64Value); //设置当前值
```

### 2.10.6.3. 浮点数控制

通过 CFloatFeaturePointer 来访问浮点类型数，可以读写当前值，查询最大值、最小值、是否支持步长、步长值、单位。

代码样例：

```
CGXFeatureControlPointer objFeatureControlPtr=
    objDevicePtr->GetRemoteFeatureControl();
CFloatFeaturePointer objFloatPtr =
    objFeatureControlPtr->GetFloatFeature("Gain");
double dMax = objFloatPtr->GetMax(); //获取最大值
double dMin = objFloatPtr->GetMin(); //获取最小值
bool bHasInc = objFloatPtr->HasInc();
if (bHasInc)
{
    double dInc = objFloatPtr->GetInc(); //获取步长
}
GxIAPICPP::gxstring strUnit = objFloatPtr->GetUnit(); //获取单位
double dValue = objFloatPtr->GetValue(); //获取当前值
objFloatPtr->SetValue(dValue); //设置当前值
```

### 2.10.6.4. 布尔类型控制

布尔类型相对比较简单，只有设置 true 和 false 两个接口。

代码样例：

```
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
CBoolFeaturePointer objBoolPtr =
    objFeatureControlPtr->GetBoolFeature("ChunkModeActive");
bool bValue = objBoolPtr->GetValue(); //获取当前值
objBoolPtr->SetValue(bValue); //设置当前值
```



### 2.10.6.5. 枚举类型控制

枚举类型可以查询枚举功能支持的可选项以及读写接口。

代码样例：

```
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
CEnumFeaturePointer objEnumPtr =
    objFeatureControlPtr->GetEnumFeature("AcquisitionMode");
GxIAPICPP::gxstring_vector vecEnumEntry;
vecEnumEntry = objEnumPtr->GetEnumEntryList(); //获取枚举项列表
for(int i = 0; i < vecEnumEntry.size(); i++)
{
    cout << vecEnumEntry[i] << endl; //打印所有枚举项
}
GxIAPICPP::gxstring strValue = objEnumPtr->GetValue(); //获取当前项
objEnumPtr->SetValue(strValue); //设置当前项
```

### 2.10.6.6. 字符串类型控制

字符串类型提供了读写当前值的功能，写入值的时候可以先查询最大支持写入多长字符串。

代码样例：

```
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
CStringFeaturePointer objStringPtr =
    objFeatureControlPtr->GetStringFeature("DeviceUserID");
GxIAPICPP::gxstring strValue = objStringPtr->GetValue(); //获取当前值
int64_t i64MaxLength = objStringPtr->GetStringMaxLength(); //获取字符串可写入最大长度
objStringPtr->SetValue(strValue); //写入当前值
```

### 2.10.6.7. 命令类型控制

命令类型最简单，只有一个执行接口。

代码样例：

```
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
CCommandFeaturePointer objCommandPtr =
    objFeatureControlPtr->GetCommandFeature("AcquisitionStart");
objCommand->Execute(); //执行命令
```

### 2.10.7. 注意事项

属性控制器 CGXFeatureControlPointer 从设备对象获取之后，在关闭设备之前都有效，一旦执行了关闭设备，属性控制器也会失效，就不能调用任何属性控制器的接口了。

## 2.11. 采集控制与图像处理

与采集相关的接口和控制都在 CGXStreamPointer 对象上面，获取和打开流对象的代码如下：

```
uint32_t ui32StreamNum = objDevicePtr->GetStreamCount();
if (nStreamNum > 0)
{
    CGXStreamPointer objStreamPtr = objDevicePtr->OpenStream(0);

    //流对象控制或者采集

    //当用户不使用流对象的时候，需要将其关闭
    objStreamPtr->Close();
}
```

采集方式分为两种：回调采集和采单帧。

### 2.11.1. 零拷贝采单帧 (DQBuf)

用户开启流对象采集并且给设备发送开采命令之后，就可以调用 DQBuf 接口采单帧。

代码样例：

```
CGXStreamPointer objStreamPtr = objDevicePtr->OpenStream(0);

//提高网络相机的采集性能, 设置方法参考以下代码（目前只有千兆网系列相机支持设置最优包长）
GX_DEVICE_CLASS_LIST objDeviceClass = m_objDevicePtr->GetDeviceInfo().GetDeviceClass();
if (GX_DEVICE_CLASS_GEV == objDeviceClass)
{
    //判断设备是否支持流通道数据包功能
    if (true == m_objFeatureControlPtr->IsImplemented("GevSCPSPacketSize"))
    {
        //获取当前网络环境的最优包长值
        int nPacketSize = m_objStreamPtr->GetOptimalPacketSize();

        //将最优包长值设置为当前设备的流通道包长值
        m_objFeatureControlPtr->GetIntFeature("GevSCPSPacketSize")->SetValue(
                                                                    nPacketSize);
    }
}

//开启流通道采集
objStreamPtr->StartGrab();

//给设备发送开采命令
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
objFeatureControlPtr->GetCommandFeature("AcquisitionStart")->Execute();

//零拷贝采单帧
CImageDataPointer objImageDataPtr;
```



```
objImageDataPtr = objStreamPtr->DQBUF(500); //超时时间使用 500ms, 用户可以自行设定
if (GX_FRAME_STATUS_SUCCESS == objImageDataPtr->GetStatus())
{
    //采图成功而且是完整帧, 可以进行图像处理...
}

//图像使用完毕后需要调用 QBuf, 否则可能导致无法正常循环获取图像
objStreamPtr->QBuf(objImageDataPtr);

//停采
objFeatureControlPtr->GetCommandFeature("AcquisitionStop")->Execute();
objStreamPtr->StopGrab();

//关闭流通道
objStreamPtr->Close();
```

**注意：**必须先调用 StartGrab 开启流通道的采集，然后再给设备发送开采命令，否则开采命令无效。

### 2.11.2. 采单帧 (GetImage)

用户开启流对象采集并且给设备发送开采命令之后，就可以调用 GetImage 接口采单帧。

代码样例：

```
CGXStreamPointer objStreamPtr = objDevicePtr->OpenStream(0);

//提高网络相机的采集性能, 设置方法参考以下代码 (目前只有千兆网系列相机支持设置最优包长)
GX_DEVICE_CLASS_LIST objDeviceClass =
    m_objDevicePtr->GetDeviceInfo().GetDeviceClass();
if (GX_DEVICE_CLASS_GEV == objDeviceClass)
{
    //判断设备是否支持流通道数据包功能
    if (true == m_objFeatureControlPtr->IsImplemented("GevSCPSPacketSize"))
    {
        //获取当前网络环境的最优包长值
        int nPacketSize = m_objStreamPtr->GetOptimalPacketSize();

        //将最优包长值设置为当前设备的流通道包长值
        m_objFeatureControlPtr->GetIntFeature("GevSCPSPacketSize")->SetValue(
            nPacketSize);
    }
}

//开启流通道采集
objStreamPtr->StartGrab();

//给设备发送开采命令
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
objFeatureControlPtr->GetCommandFeature("AcquisitionStart")->Execute();
```

```
//采单帧
CImageDataPointer objImageDataPtr;

objImageDataPtr = objStreamPtr->GetImage(500); //超时时间使用500ms, 用户可以自行设定
if (GX_FRAME_STATUS_SUCCESS == objImageDataPtr->GetStatus())
{
    //采图成功而且是完整帧, 可以进行图像处理...
}

//停采
objFeatureControlPtr->GetCommandFeature("AcquisitionStop")->Execute();
objStreamPtr->StopGrab();

//关闭流通道
objStreamPtr->Close();
```

**注意：**必须先调用 StartGrab 开启流通道的采集，然后再给设备发送开采命令，否则开采命令无效。当使用高分辨率相机进行高速采集的时候，因为 GetImage 接口内部有 buffer 拷贝会影响传输性能，建议用户在此种情况下使用 DQBuf 方式或者回调采集方式。

### 2.11.3. 回调采集

用户需要继承 ICaptureEventHandler 虚基类实现自己的回调处理类。

代码样例：

```
class CSampleCaptureEventHandler : public ICaptureEventHandler
{
public:
    void DoOnImageCaptured(CImageDataPointer& objImageDataPointer, void* pUserParam)
    {
        if (GX_FRAME_STATUS_SUCCESS == objImageDataPointer->GetStatus())
        {
            //图像获取为完整帧, 可以读取图像宽、高、数据格式等
            uint64_t ui64Width = objImageDataPointer->GetWidth();
            uint64_t ui64Height = objImageDataPointer->GetHeight();
            GX_PIXEL_FORMAT_ENTRY emPixelFormat =
                objImageDataPointer->GetPixelFormat();

            //其他图像信息的获取参见 IImageData 接口定义
        }
    }
};
```

用户可以注册回调函数采集。

代码样例：

```
CGXStreamPointer objStreamPtr = objDevicePtr->OpenStream(0);
//提高网络相机的采集性能, 设置方法参考以下代码 (目前只有千兆网系列相机支持设置最优包长)
```

```

GX_DEVICE_CLASS_LIST objDeviceClass =
    m_objDevicePtr->GetDeviceInfo().GetDeviceClass();
if (GX_DEVICE_CLASS_GEV == objDeviceClass)
{
    //判断设备是否支持流通道数据包功能
    if (true == m_objFeatureControlPtr->IsImplemented("GevSCPSPacketSize"))
    {
        //获取当前网络环境的最优包长值
        int nPacketSize = m_objStreamPtr->GetOptimalPacketSize();

        //将最优包长值设置为当前设备的流通道包长值
        m_objFeatureControlPtr->GetIntFeature(
            "GevSCPSPacketSize")->SetValue(nPacketSize);
    }
}

//注册采集回调函数, 注意第一个参数是用户私有参数, 用户可以传入任何 object 对象, 也可以是 null
//用户私有参数在回调函数内部还原使用, 如果不使用私有参数, 可以传入 null
ICaptureEventHandler* pCaptureEventHandler =
    new CSampleCaptureEventHandler();
objStreamPtr->RegisterCaptureCallback(pCaptureEventHandler, NULL);

//开启流通道采集
objStreamPtr->StartGrab();

//给设备发送开采命令
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
objFeatureControlPtr->GetCommandFeature("AcquisitionStart")->Execute();

//回调采集过程, 参见回调函数

//停采、注销采集回调函数
objFeatureControlPtr->GetCommandFeature("AcquisitionStop")->Execute();
objStreamPtr->StopGrab();
objStreamPtr->UnregisterCaptureCallback();
delete pCaptureEventHandler;
pCaptureEventHandler = NULL;

//关闭流通道
objStreamPtr->Close();

```

**注意：**必须先调用 StartGrab 开启流通道的采集，然后再给设备发送开采命令，否则开采命令无效。

#### 2.11.4. 设置采集 buffer 个数

用户开启流对象之后并且开启流层采集之前，可以调用 SetAcquisitionBufferNumber 设置采集 buffer 个数。

代码样例：

```
CGXStreamPointer objStreamPtr = objDevicePtr->OpenStream(0);

//提高网络相机的采集性能,设置方法参考以下代码（目前只有千兆网系列相机支持设置最优包长）
GX_DEVICE_CLASS_LIST objDeviceClass =
    m_objDevicePtr->GetDeviceInfo().GetDeviceClass();
if (GX_DEVICE_CLASS_GEV == objDeviceClass)
{
    //判断设备是否支持流通道数据包功能
    if (true == m_objFeatureControlPtr->IsImplemented("GevSCPSPacketSize"))
    {
        //获取当前网络环境的最优包长值
        int nPacketSize = m_objStreamPtr->GetOptimalPacketSize();

        //将最优包长值设置为当前设备的流通道包长值
        m_objFeatureControlPtr->GetIntFeature(
            "GevSCPSPacketSize")->SetValue(nPacketSize);
    }
}

//设置采集 buffer 个数
objStreamPtr->SetAcquisitionBufferNumber(10);

//开启流通道采集
objStreamPtr->StartGrab();

//给设备发送开采命令
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();
objFeatureControlPtr->GetCommandFeature("AcquisitionStart")->Execute();

// .....

//停采
objFeatureControlPtr->GetCommandFeature("AcquisitionStop")->Execute();
objStreamPtr->StopGrab();

//关闭流通道
objStreamPtr->Close();
```

**注意:**

- 1) 该接口是可选的。在多机同时采集的场景下,如果出现个别相机采集帧率为 0 的情况时,可以调用该接口调整所有相机的采集 buffer 个数,从而保证所有的相机有用于采集的 buffer。
- 2) 必须在调用 StartGrab 开启流通道的采集之前,设置采集 buffer 个数,否则设置无效。

### 2.11.5. 图像处理

#### 2.11.5.1. 图像格式转换

功能描述:指定获取 8 位有效数据。假设原始数据为非 8 位数据,无论黑白或者彩色,都可以调用 ConvertToRaw8 来指定获取 8 位有效数据,返回类型 void\*。所指内存大小是 Width\*Height。

代码样例:

```
void* pRaw8Buffer = NULL;
//假设原始数据是 Mono8 图像
pRaw8Buffer = objImageDataPtr->ConvertToRaw8(GX_BIT_0_7);
//假设原始数据是 Mono12 图像
pRaw8Buffer = objImageDataPtr->ConvertToRaw8(GX_BIT_4_11);
//假设原始数据是 BayerRG8 图像
pRaw8Buffer = objImageDataPtr->ConvertToRaw8(GX_BIT_0_7);
//假设原始数据是 BayerRG12 图像
pRaw8Buffer = objImageDataPtr->ConvertToRaw8(GX_BIT_4_11);
```

功能描述: Bayer 转 RGB24,无论当前图像输出为黑白或者彩色,8 位或者非 8 位,都能够调用 ConvertToRGB24 完成数据的插值处理,最终返回 RGB24 格式的 void\*。所指内存大小是 Width\*Height\*3。

代码样例:

```
void* pRGB24Buffer = NULL;
//假设原始数据是 Mono8 图像
pRGB24Buffer = objImageDataPtr->ConvertToRGB24(GX_BIT_0_7,
                                                GX_RAW2RGB_NEIGHBOUR, true);
//假设原始数据是 Mono12 图像
pRGB24Buffer = objImageDataPtr->ConvertToRGB24(GX_BIT_4_11,
                                                GX_RAW2RGB_NEIGHBOUR, true);
//假设原始数据是 BayerRG8 图像
pRGB24Buffer = objImageDataPtr->ConvertToRGB24(GX_BIT_0_7,
                                                GX_RAW2RGB_NEIGHBOUR, true);
//假设原始数据是 BayerRG12 图像
pRGB24Buffer = objImageDataPtr->ConvertToRGB24(GX_BIT_4_11,
                                                GX_RAW2RGB_NEIGHBOUR, true);
```

#### 2.11.5.2. 图像效果增强

本接口库还提供了软件端的图像效果增强接口,用户可以选择的进行坏点校正、锐化、对比度、亮度等图像效果增强的操作。图像效果增强涉及到 CImageProcessConfigPointer 和 CImageDataPointer。

代码样例:

```
//通过设备对象构建图像处理配置对象
CImageProcessConfigPointer objImageProcessConfigPtr =
    objDevicePtr->CreateImageProcessConfig();
//objImageDataPtr 可以是采集回调函数传入的还可以是 GetImage/DQBuf 获取的
void* pRGB24Processed = NULL;
//返回结果就是经过图像效果增强之后的 RGB24 格式的数据
pRGB24Processed = objIBaseData->ImageProcess(objImageProcessConfigPtr);
```

对不同数据类型的相机，接口库提供的效果增强的支持情况也不同，见表 2-1。

功能	对应接口	原始数据			
		Mono8	Mono 非 8	Bayer8	Bayer 非 8
选取有效位	SetValidBit	支持	支持	支持	支持
坏点校正开关	EnableDefectivePixelCorrect	支持	支持	支持	支持
锐化开关	EnableSharpen	支持	支持	支持	支持
锐化因子	SetSharpenParam	支持	支持	支持	支持
对比度	SetContrastParam	支持	支持	支持	支持
Gamma 调节	SetGammaParam	支持	支持	支持	支持
亮度	SetLightnessParam	支持	支持	支持	支持
降噪开关	EnableDenoise	不支持	不支持	支持	支持
饱和度	SetSaturationParam	不支持	不支持	支持	支持
RGB24 插值类型	SetConvertType	不支持	不支持	支持	支持
RGB24 插值翻转开关	EnableConvertFlip	不支持	不支持	支持	支持
颜色校正开关	EnableColorCorrection	不支持	不支持	支持	支持

表 2-1 图像效果增强对照表

图像效果增强参数的范围及详细解释见表 2-2。

功能	默认值	范围	备注
选取有效位	GX_BIT_0_7	参见 GX_VALID_BIT_LIST	<ul style="list-style-type: none"><li>对于 8 位数据只能选择 GX_BIT_0_7</li><li>对于 10 位数据只能选择 GX_BIT_0_7、GX_BIT_1_8、GX_BIT_2_9，建议选择 GX_BIT_2_9</li><li>对于 12 位数据可以选择 GX_BIT_0_7、GX_BIT_1_8 、 GX_BIT_2_9 、 GX_BIT_3_10、GX_BIT_4_11，建议选</li></ul>

			择 GX_BIT_4_11
坏点校正开关	false	true, false	true: 开启坏点校正; false: 关闭坏点校正
锐化开关	false	true, false	true: 开启锐化; false: 关闭锐化
锐化因子	0.1	[0.1, 5]	从小到大逐渐增强锐化效果
对比度	0	[-50, 100]	0: 对比度没有变化 大于 0: 增加对比度 小于 0: 减小对比度
Gamma 调节	1.0	[0.1, 10]	-
亮度	0	[-150, 150]	0: 亮度没有变化 大于 0: 增加亮度 小于 0: 减小亮度
降噪开关	false	true, false	true: 开启降噪; false: 关闭降噪
饱和度	64	[0, 128]	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
RGB24 插值类型	GX_RAW2RGB_NEIGHBOUR	参见 GX_BAYER_CONVE RT_TYPE_LIST	-
RGB24 插值 翻转开关	false	true, false	true: 翻转; false: 不翻转

表 2-2 图像效果增强参数的范围和说明

高级的用户可以在获取 CImageProcessConfigPointer 对象之后进行参数微调，如下：

```
//通过设备对象构建图像处理配置对象
CImageProcessConfigPointer objImageProcessConfigPtr =
    objDevicePtr->CreateImageProcessConfig();

//objImageProcessConfigPtr 对象在构建的时候会初始化默认配置参数，用户可以选择对配置
//参数进行微调，如下：
objImageProcessConfigPtr->SetValidBit(GX_BIT_0_7); //选择有效数据位 0~7
objImageProcessConfigPtr->EnableDefectivePixelCorrect(true); //使能坏点校正功能
objImageProcessConfigPtr->EnableSharpen(true); //使能锐化
objImageProcessConfigPtr->SetSharpenParam(1); //设置锐化强度因子 1
objImageProcessConfigPtr->SetContrastParam(0); //设置对比度调节参数
objImageProcessConfigPtr->SetGammaParam(1); //设置 Gamma 系数
```



```
objImageProcessConfigPtr->SetLightnessParam(0);    //设置亮度调节参数
objImageProcessConfigPtr->EnableDenoise(true);    //使能降噪开关（黑白相机不支持）
//设置饱和度调节系数（黑白相机不支持）
objImageProcessConfigPtr->SetSaturationParam(0);
//设置插值算法（黑白相机不支持）
objImageProcessConfigPtr->SetConvertType(GX_RAW2RGB_NEIGHBOUR);
//使能插值翻转（黑白相机不支持）
objImageProcessConfigPtr->EnableConvertFlip(true);

//用户还可以选择恢复最佳默认参数配置
objImageProcessConfigPtr->Reset();

//objImageDataPtr 可以是采集回调函数传入的还可以是 GetImage/DQBuf 获取的
void* pRGB24Processed = NULL;
//返回结果就是经过图像效果增强之后的 RGB24 格式的数据
pRGB24Processed = objImageDataPtr->ImageProcess(objImageProcessConfigPtr);
```

### 2.11.5.3. 效果图样

根据相机类型不同，使用场景不同，图像效果增强配置参数也不同，但是还是有几个推荐用户使用的功能可以解决大部分需求，下面以水星千兆网相机 MER-200-20GC 为例，相机输出的图像经过图像格式转换成 RGB24 位图像之后，不做任何图像效果增强处理如图 2-8 所示。



图 2-8 邻域插值得到的 RGB 图（颜色校正：未开启；饱和度：64；Gamma 值：1；对比度：0）



在图 2-8 的基础上开启颜色校正如图 2-9 所示。



图 2-9 开启颜色校正 (颜色校正: 开启; 饱和度: 64; Gamma 值: 1; 对比度: 0)

在图 2-9 的基础上增强饱和度到 80 的效果如图 2-10 所示。



图 2-10 设置饱和度 80 (颜色校正: 开启; 饱和度: 80; Gamma 值: 1; 对比度: 0)

在图 2-10 的基础上调节 Gamma 值为 1.98, 使图像更接近于现实场景中肉眼所见图像 (图 2-11), 1.98 只是示例值, 不同相机的 Gamma 值不同, USB2.0 和 USB3.0 相机可以通过通用读写通道读取 Gamma 值; GigE 千兆网相机没有直接的接口读取此值, 需要用户自己看实际情况适当调节。



图 2-11 设置 Gamma1.98 (颜色校正: 开启; 饱和度: 80; Gamma 值: 1.98; 对比度: 0)

在图 2-11 的基础上将对比度设置为 40 的效果如图 2-12 所示。



图 2-12 设置对比度 40 (颜色校正: 开启; 饱和度: 80; Gamma 值: 1.98; 对比度: 40)

### 2.11.6. 流对象属性控制

在“属性控制”章节已经提到流层也有自己的属性控制，通过 `IGXStream::GetFeatureControl` 接口获取。流层属性控制里面包含的控制属性可能是采集相关的控制和统计信息等，下面以千兆网流层统计为例，比如在采集过程中查看当前采集统计信息：

```
//objGXStream 为通过 IGXDevice::OpenStream 获取到的 CGXStreamPointer 对象  
CGXFeatureControlPointer objStreamFeatureControlPtr =  
    objGXStream->GetFeatureControl();
```

```
//查看采集统计信息
//buffer 不足导致丢帧数
objStreamFeatureControlPtr->GetIntFeature (
    "StreamLostFrameCount")->GetValue();

//接收的残帧个数
objStreamFeatureControlPtr->GetIntFeature (
    "StreamIncompleteFrameCount")->GetValue();

//接收到的包数
objStreamFeatureControlPtr->GetIntFeature (
    "StreamDeliveredPacketCount")->GetValue();

//重传包个数
objStreamFeatureControlPtr->GetIntFeature (
    "StreamResendPacketCount")->GetValue();

//设置采集配置参数
//设置块超时时间 200ms
objStreamFeatureControlPtr->GetIntFeature (
    "BlockTimeout")->SetValue(200);
```

流对象属性中“StreamBufferHandlingMode”可以设置 Buffer 的处理模式，Buffer 处理模式目前支持三种，分别为：

- 1) OldestFirst：默认值。图像缓冲区遵守先进先出的原则，所有的缓冲区全部填满后，新的图像数据会被丢弃，直到用户完成已经填满图像数据的缓冲区处理。典型应用场景是，要求接收到相机采集的每帧图像，不丢帧。该模式实现不丢帧，还需要图像数据的传输与处理的速度尽量快（至少小于帧周期）。
- 2) OldestFirstOverwrite：同样遵守先进先出的原则。与 OldestFirst 模式的区别是，当所有的缓冲区全部填满后，SDK 将主动丢弃缓冲区中时间戳最旧的一帧图像缓冲区，用于接收新的图像数据。典型的应用成绩是，不要求接收相机采集的每帧图像，应用环境下图像传输与处理速度较慢。
- 3) NewestOnly：该模式下用户拿到的始终是 SDK 接收到的最新图。SDK 每接收到一帧新的图像数据，就会主动丢弃旧时间戳的图像，因此当用户图像处理不及时或者速度较慢时，就会出现丢帧。该模式主要应用场合是，对图像采集与显示实时性要求比较高，且不要求接收到相机采集的每帧图像。但是受相机的采集帧率和内部缓存，以及传输速度、用户使用场景的限制，SDK 接收的最新图与相机最新曝光的图像可能有延迟。

**注意：**示例代码中展示的流程控制器的功能是以千兆网为例，其他类型设备请使用 GalaxyView 演示程序打开设备查看，或者调用 IGXFeatureControl::GetFeatureNameList 来获取功能名称列表。

#### 2.11.7. 注意事项

流对象从设备对象获取之后，在关闭设备之前都有效，一旦执行了关闭设备，流对象也会失效，就不能调用任何流对象的接口了。

## 2.12. 获取设备事件

大恒 GigE Vision 相机能够发送事件信息,比如当 Sensor 曝光完成之后会发送一个曝光结束事件给 PC,此事件先于图像数据到达。本节主要讲解如何获取事件以及事件数据。

### 2.12.1. 选择事件

功能码字符串	功能类型	中文描述	可选项 (以实际查询当前设备为准)
EventSelector	枚举	事件源选择	可能包含以下可选项: ExposureEnd [曝光结束] BlockDiscard [图像帧丢弃] EventOverrun [事件队列溢出] FrameStartOvertrigger [触发信号溢出] BlockNotEmpty [图像帧存不为空] InternalError [内部错误]

### 2.12.2. 使能事件

功能码字符串	功能类型	中文描述	可选项 (以实际查询当前设备为准)
EventNotification	枚举	事件使能	可能包含以下可选项: Off [关闭]    On [开启]

### 2.12.3. 注册事件通知回调函数

注册事件使用的接口: IGXFeatureControl.RegisterFeatureCallback, 此接口第一个参数就是写明想要注册的事件功能码,注册事件时可供选择的事件功能码如下:

功能码字符串	功能类型	中文描述
EventExposureEnd	整型	曝光结束事件 ID
EventBlockDiscard	整型	数据块丢失事件 ID
EventOverrun	整型	事件队列溢出事件 ID
EventFrameStartOvertrigger	整型	触发信号被屏蔽事件 ID
EventBlockNotEmpty	整型	帧存不为空事件 ID
EventInternalError	整型	内部错误事件 ID

### 2.12.4. 获取事件数据信息

在事件回调函数内部可以通过 CGXFeatureControlPointer 读写通道读取当前的事件信息,2.11.4 节说明了 5 种事件类型,这 5 种事件类型携带的事件信息可以通过以下控制码获取:

注册事件用字符串	相关事件数据字符串	功能类型	中文描述
EventExposureEnd	EventExposureEndTimestamp	整型	曝光结束事件时间戳
	EventExposureEndFrameID	整型	曝光结束事件帧 ID

EventBlockDiscard	EventBlockDiscardTimestamp	整型	数据块丢失事件时间戳
EventOverrun	EventOverrunTimestamp	整型	事件队列溢出事件时间戳
EventFrameStartOvertrigger	EventFrameStartOvertriggerTimestamp	整型	触发信号被屏蔽事件时间戳
EventBlockNotEmpty	EventBlockNotEmptyTimestamp	整型	帧存不为空事件时间戳
EventInternalError	EventInternalErrorTimestamp	整型	内部错误事件时间戳

### 2.12.5. 代码样例

下面以获取曝光结束事件为例，用户必须继承 IFeatureEventHandler 虚基类实现自己的回调类。

代码片段：

```
class CSampleFeatureEventHandler : public IFeatureEventHandler
{
public:
    void DoOnFeatureEvent (
        const GxIAPICPP::gxstring& strFeatureName, void* pUserParam)
    {
        cout << "发生曝光结束事件！" << endl;

        //pUserParam 是用户注册回调函数的时候传入的，此处将其还原用来获取事件数据
        CGXFeatureControlPointer* pObjFeatureControlPtr =
            (CGXFeatureControlPointer*)pUserParam;

        //获取曝光结束事件时间戳
        (*pObjFeatureControlPtr)->GetIntFeature (
            "EventExposureEndTimestamp")->GetValue();

        //获取曝光结束事件帧 ID
        (*pObjFeatureControlPtr)->GetIntFeature (
            "EventExposureEndFrameID")->GetValue();
    }
};
```

```
//objDevicePtr 为 CGXDevicePointer 设备对象，设备已经打开
//设备事件属性在远端设备属性控制器上，首先应该获取远端设备属性控制器
CGXFeatureControlPointer objFeatureControlPtr =
    objDevicePtr->GetRemoteFeatureControl();

//选择事件源
objFeatureControlPtr->GetEnumFeature ("EventSelector")->SetValue (
    "ExposureEnd");

//使能事件
objFeatureControlPtr->GetEnumFeature (
    "EventNotification")->SetValue ("On");
```

```
//注册事件回调函数,注意参数三是用户私有参数,用户可以传入任何指针
//此处我们演示传入属性控制器指针,因为稍后会在回调函数内部使用此对象获取曝光结束事
//件数据信息
//此私有参数在回调函数内部可以被还原供用户使用,如果用户不使用私有参数,可以简单的将
//此参数设置为 NULL
GX_FEATURE_CALLBACK_HANDLE hEventHandle = NULL;
IFeatureEventHandler* pFeatureEventHandler =
    new CSampleFeatureEventHandler();
hEventHandle = objFeatureControlPtr->RegisterFeatureCallback(
    "EventExposureEnd", pFeatureEventHandler, &objFeatureControlPtr);

//开启流通道采集
objStreamPtr->StartGrab();

//给设备发送开采命令
objFeatureControlPtr->GetCommandFeature("AcquisitionStart")->Execute();

//发送开采命令,相机开始曝光输出图像,当曝光结束的时候会产生曝光结束事件,此时就会
//激活回调 OnFeatureCallback 接口
//接收曝光结束事件,见 OnFeatureCallback

//发送停采命令
objFeatureControlPtr->GetCommandFeature("AcquisitionStop")->Execute();
objStreamPtr->StopGrab();

//注销事件
objFeatureControlPtr->UnregisterFeatureCallback(objEventHandle);
delete pFeatureEventHandler;
pFeatureEventHandler = NULL;
```

**注意：**必须先发送开采命令，相机才能输出曝光结束事件，而且发送开采命令，必须严格按照：先调用 IGXStream::StartGrab 接口，然后再通过属性控制器获取“AcquisitionStart”命令节点，执行命令 Execute。

## 2.13. 获取掉线通知

目前只有 Gige Vision 相机提供了设备掉线通知事件机制，用户可以通过回调通知的方式获取设备掉线事件，打开设备之后就可以注册此事件。

获取掉线事件分成以下几步：

- 1) 注册掉线事件;
- 2) 设备掉线之后激发回调函数;
- 3) 注销掉线事件。



用户必须继承 IDeviceOfflineEventHandler 虚基类回调函数，代码片段如下：

```
class CSampleDeviceOfflineEventHandler : public IDeviceOfflineEventHandler
{
public:
    void DoOnDeviceOfflineEvent(void* pUserParam)
    {
        cout << " 掉线!"<<endl;
    }
};
```

注册掉线事件示例代码片段如下：

```
GX_DEVICE_OFFLINE_CALLBACK_HANDLE hDeviceOffline = NULL;
//objDevicePtr 为 CGXDevicePointer 设备对象，设备已经打开
//第二个参数为用户私有参数，用户可以在回调函数内部将其还原然使用，如果不需要则可传入 NULL 即可
IDeviceOfflineEventHandler* pDeviceOfflineEventHandler =
    new CSampleDeviceOfflineEventHandler();
hDeviceOffline = objDevicePtr->RegisterDeviceOfflineCallback(
    pDeviceOfflineEventHandler, NULL);
```

在关闭设备之前一定要注销事件：

```
objDevicePtr->UnregisterDeviceOfflineCallback(hDeviceOffline);
delete pDeviceOfflineEventHandler;
pDeviceOfflineEventHandler = NULL;
```

**注意：**在掉线回调函数内部不允许执行关闭设备操作。

## 2.14. 导入导出相机配置参数

通过演示程序 GalaxyView 打开设备之后，展开菜单栏的“文件”会看到“导入设备配置”和“导出设备配置”，如图 2-13 所示。



图 2-13 导入导出设备配置

选择“导出设备配置”会在本地磁盘新建一个文本文件，保存设备此时的运行参数。

选择“导入设备配置”选择本地已经存在的文本文件，然后导入。

在接口库中也有与之对应的用户接口供用户调用。

导入导出配置文件示例代码：

```
//打开设备，此处以序列号为例打开设备，序列号以实际设备为准。用户可以选择其他方式打开设备
CGXDevicePointer objDevicePtr = objIGXFactory->OpenDeviceBySN(
    "RN0001007012", GX_ACCESS_EXCLUSIVE);

//打开设备下面第一个流对象
CGXStreamPointer objStreamPtr = objDevicePtr->OpenStream(0);

//导出配置文件，文件名用户可以任意。导出目录必须已经存在
objDevicePtr->ExportConfigFile("D:\\Config.txt");

//导入配置文件
objDevicePtr->ImportConfigFile("D:\\Config.txt");
```

**注意：**执行导入导出配置文件之前，必须先打开设备，并且打开设备下面第一个流对象。



### 3. 智能指针对象类型定义

C++接口库开放给用户的控制对象大多是以智能指针的方式封装，详见 GXSmartPtr.h 头文件对智能指针类型的定义。

#### 类型重定义

```
typedef GXSmartPtr<IGXDevice>CGXDevicePointer;
typedef GXSmartPtr<IGXStream>CGXStreamPointer;
typedef GXSmartPtr<IImageProcessConfig> CImageProcessConfigPointer;
typedef GXSmartPtr<IImageData>CImageDataPointer;
typedef GXSmartPtr<IGXFeatureControl> CGXFeatureControlPointer;
typedef GXSmartPtr<IIntFeature>CIntFeaturePointer;
typedef GXSmartPtr<IFloatFeature> CFloatFeaturePointer;
typedef GXSmartPtr<IEnumFeature> CEnumFeaturePointer;
typedef GXSmartPtr<IBoolFeature>CBoolFeaturePointer;
typedef GXSmartPtr<IStringFeature> CStringFeaturePointer;
typedef GXSmartPtr<ICommandFeature>CCommandFeaturePointer;
typedef GXSmartPtr<IRegisterFeature>CRegisterFeaturePointer;
typedef GXSmartPtr<IGXImageSpecialFunction> CGXImageSpecialFunctionPointer;
typedef GXSmartPtr<IGXImageFormatConvert> CGXImageFormatConvertPointer;
```

为了避免直接让用户获取库内部的指针，采用智能指针封装成智能指针对象的形式开放给用户，用户可以像使用指针一样使用这些重定义的智能指针对象，这些智能指针对象拥有的功能同其封装对象，比如用户可以直接使用一个 CGXDevicePointer 对象 ObjDevicePointer 如下：

ObjDevicePointer->OpenStream(0); 这些接口都可以在 IGXDevice 头文件声明中找到定义。

智能指针对象不需要刻意执行销毁动作，其采用内部引用计数的机制，当引用计数为 0 的时候才真正的销毁资源。

## 4. 数据类型定义

### 4.1. GX\_STATUS\_LIST

定义	值	解释
GX_STATUS_SUCCESS	0	成功
GX_STATUS_ERROR	-1	不期望发生的错误
GX_STATUS_NOT_FOUND_TL	-2	找不到 TL 库
GX_STATUS_NOT_FOUND_DEVICE	-3	找不到设备
GX_STATUS_OFFLINE	-4	当前设备为掉线状态
GX_STATUS_INVALID_PARAMETER	-5	无效参数
GX_STATUS_INVALID_HANDLE	-6	无效句柄
GX_STATUS_INVALID_CALL	-7	无效的接口调用
GX_STATUS_INVALID_ACCESS	-8	功能当前不可访问或设备访问模式错误
GX_STATUS_NEED_MORE_BUFFER	-9	用户 buffer 不足
GX_STATUS_ERROR_TYPE	-10	Feature 类型错误，比如用 IntFeature 操作浮点功能
GX_STATUS_OUT_OF_RANGE	-11	写入值越界
GX_STATUS_NOT_IMPLEMENTED	-12	不支持的功能
GX_STATUS_NOT_INIT_API	-13	没有调用初始化接口
GX_STATUS_TIMEOUT	-14	超时错误

### 4.2. GX\_DEVICE\_CLASS\_LIST

定义	值	解释
GX_DEVICE_CLASS_UNKNOWN	0	未知设备类型
GX_DEVICE_CLASS_USB2	1	USB2.0 设备
GX_DEVICE_CLASS_GEV	2	千兆网设备
GX_DEVICE_CLASS_U3V	3	USB3.0 设备
GX_DEVICE_CLASS_SMART	4	Smart 设备
GX_DEVICE_CLASS_CXP	5	CXP 设备

### 4.3. GX\_ACCESS\_STATUS

定义	值	解释
GX_ACCESS_STATUS_UNKNOWN	0	设备当前状态未知
GX_ACCESS_STATUS_READWRITE	1	设备当前状态可读可写
GX_ACCESS_STATUS_READONLY	2	设备当前只支持读
GX_ACCESS_STATUS_NOACCESS	3	设备当前既不支持读，又不支持写

## 4.4. GX\_ACCESS\_MODE

定义	值	解释
GX_ACCESS_READONLY	2	只读方式
GX_ACCESS_CONTROL	3	控制方式
GX_ACCESS_EXCLUSIVE	4	独占方式

## 4.5. GX\_PIXEL\_FORMAT\_ENTRY

定义	值	解释
GX_PIXEL_FORMAT_UNDEFINED	0x00000000	-
GX_PIXEL_FORMAT_MONO8	0x01080001	Monochrome 8-bit
GX_PIXEL_FORMAT_MONO8_SIGNED	0x01080002	Monochrome 8-bit signed
GX_PIXEL_FORMAT_MONO10	0x01100003	Monochrome 10-bit unpacked
GX_PIXEL_FORMAT_MONO10_P	0x010A0046	Monochrome 10-bit packed
GX_PIXEL_FORMAT_MONO12	0x01100005	Monochrome 12-bit unpacked
GX_PIXEL_FORMAT_MONO12_P	0x010C0047	Monochrome 12-bit packed
GX_PIXEL_FORMAT_MONO14	0x01100025	Monochrome 14-bit unpacked
GX_PIXEL_FORMAT_MONO14_P	0x010E0104	Monochrome 14-bit packed
GX_PIXEL_FORMAT_MONO16	0x01100007	Monochrome 16-bit
GX_PIXEL_FORMAT_BAYER_GR8	0x01080008	Bayer Green-Red 8-bit
GX_PIXEL_FORMAT_BAYER_RG8	0x01080009	Bayer Red-Green 8-bit
GX_PIXEL_FORMAT_BAYER_GB8	0x0108000A	Bayer Green-Blue 8-bit
GX_PIXEL_FORMAT_BAYER_BG8	0x0108000B	Bayer Blue-Green 8-bit
GX_PIXEL_FORMAT_BAYER_GR10	0x0110000C	Bayer Green-Red 10-bit
GX_PIXEL_FORMAT_BAYER_GR10_P	0x010A0056	Bayer Green-Red 10-bit packed
GX_PIXEL_FORMAT_BAYER_RG10	0x0110000D	Bayer Red-Green 10-bit
GX_PIXEL_FORMAT_BAYER_RG10_P	0x010A0058	Bayer Red-Green 10-bit packed
GX_PIXEL_FORMAT_BAYER_GB10	0x0110000E	Bayer Green-Blue 10-bit
GX_PIXEL_FORMAT_BAYER_GB10_P	0x010A0054	Bayer Green-Blue 10-bit packed
GX_PIXEL_FORMAT_BAYER_BG10	0x0110000F	Bayer Blue-Green 10-bit
GX_PIXEL_FORMAT_BAYER_BG10_P	0x010A0052	Bayer Blue-Green 10-bit packed
GX_PIXEL_FORMAT_BAYER_GR12	0x01100010	Bayer Green-Red 12-bit
GX_PIXEL_FORMAT_BAYER_GR12_P	0x010C0057	Bayer Green-Red 12-bit packed
GX_PIXEL_FORMAT_BAYER_RG12	0x01100011	Bayer Red-Green 12-bit
GX_PIXEL_FORMAT_BAYER_RG12_P	0x010C0059	Bayer Red-Green 12-bit packed

GX_PIXEL_FORMAT_BAYER_GB12	0x01100012	Bayer Green-Blue 12-bit
GX_PIXEL_FORMAT_BAYER_GB12_P	0x010C0055	Bayer Green-Blue 12-bit packed
GX_PIXEL_FORMAT_BAYER_BG12	0x01100013	Bayer Blue-Green 12-bit
GX_PIXEL_FORMAT_BAYER_BG12_P	0x010C0053	Bayer Blue-Green 12-bit packed
GX_PIXEL_FORMAT_BAYER_GR14	0x01100109	Bayer Green-Red 14-bit
GX_PIXEL_FORMAT_BAYER_GR14_P	0x010E0105	Bayer Green-Red 14-bit packed
GX_PIXEL_FORMAT_BAYER_RG14	0x0110010A	Bayer Red-Green 14-bit
GX_PIXEL_FORMAT_BAYER_RG14_P	0x010E0106	Bayer Red-Green 14-bit packed
GX_PIXEL_FORMAT_BAYER_GB14	0x0110010B	Bayer Green-Blue 14-bit
GX_PIXEL_FORMAT_BAYER_GB14_P	0x010E0107	Bayer Green-Blue 14-bit packed
GX_PIXEL_FORMAT_BAYER_BG14	0x0110010C	Bayer Blue-Green 14-bit
GX_PIXEL_FORMAT_BAYER_BG14_P	0x010E0108	Bayer Blue-Green 14-bit packed
GX_PIXEL_FORMAT_BAYER_GR16	0x0110002E	Bayer Green-Red 16-bit
GX_PIXEL_FORMAT_BAYER_RG16	0x0110002F	Bayer Red-Green 16-bit
GX_PIXEL_FORMAT_BAYER_GB16	0x01100030	Bayer Green-Blue 16-bit
GX_PIXEL_FORMAT_BAYER_BG16	0x01100031	Bayer Blue-Green 16-bit
GX_PIXEL_FORMAT_RGB8_PLANAR	0x02180021	Red-Green-Blue 8-bit planar
GX_PIXEL_FORMAT_RGB10_PLANAR	0x02300022	Red-Green-Blue 10-bit unpacked
GX_PIXEL_FORMAT_RGB12_PLANAR	0x02300023	Red-Green-Blue 12-bit unpacked
GX_PIXEL_FORMAT_RGB16_PLANAR	0x02300024	Red-Green-Blue 16-bit planar
GX_PIXEL_FORMAT_RGB8	0x2180014	Red-Green-Blue 8-bit
GX_PIXEL_FORMAT_RGB10	0x2300018	Red-Green-Blue 10-bit
GX_PIXEL_FORMAT_RGB12	0x230001A	Red-Green-Blue 12-bit
GX_PIXEL_FORMAT_RGB14	0x230005E	Red-Green-Blue 14-bit
GX_PIXEL_FORMAT_RGB16	0x2300033	Red-Green-Blue 16-bit
GX_PIXEL_FORMAT_BGR8	0x2180015	Blue-Green-Red 8-bit
GX_PIXEL_FORMAT_BGR10	0x2300019	Blue-Green-Red 10-bit
GX_PIXEL_FORMAT_BGR12	0x230001B	Blue-Green-Red 12-bit
GX_PIXEL_FORMAT_BGR14	0x230004A	Blue-Green-Red 14-bit
GX_PIXEL_FORMAT_BGR16	0x230004B	Blue-Green-Red 16-bit
GX_PIXEL_FORMAT_RGBA8	0x2200016	Red-Green-Blue-Alpha 8-bit
GX_PIXEL_FORMAT_BGRA8	0x2200017	Blue-Green-Red-Alpha 8-bit
GX_PIXEL_FORMAT_ARGB8	0x2200018	Alpha-Red-Green-Blue 8-bit
GX_PIXEL_FORMAT_ABGR8	0x2200019	Alpha-Blue-Green-Red 8-bit

GX_PIXEL_FORMAT_R8	0x010800C9	Red 8-bit
GX_PIXEL_FORMAT_G8	0x010800CD	Green 8-bit
GX_PIXEL_FORMAT_B8	0x010800D1	Blue 8-bit
GX_PIXEL_FORMAT_COORD3D_ABC32F	0x026000C0	3D coordinate A-B-C 32-bit floating point
GX_PIXEL_FORMAT_COORD3D_ABC32F_PLANAR	0x026000C1	3D coordinate A-B-C 32-bit floating point planar
GX_PIXEL_FORMAT_COORD3D_C16	0x011000B8	3D coordinate C 16-bit
GX_PIXEL_FORMAT_COORD3D_C16_I16	0x0110FF02	custom pixel format
GX_PIXEL_FORMAT_COORD3D_C16_S16	0x0110FF03	custom pixel format
GX_PIXEL_FORMAT_COORD3D_C16_I16_S16	0x0110FF04	custom pixel format
GX_PIXEL_FORMAT_YUV444_8	0x2180020	YUV444 8-bit
GX_PIXEL_FORMAT_YUV422_8	0x2100032	YUV422 8-bit
GX_PIXEL_FORMAT_YUV411_8	0x20C001E	YUV411 8-bit
GX_PIXEL_FORMAT_YUV420_8_PLANAR	0x20C0040	YUV420 8-bit planar
GX_PIXEL_FORMAT_YCBCR444_8	0x218005B	YCBCR 444 8-bit
GX_PIXEL_FORMAT_YCBCR422_8	0x210003B	YCBCR 422 8-bit
GX_PIXEL_FORMAT_YCBCR411_8	0x20C005A	YCBCR 411 8-bit
GX_PIXEL_FORMAT_YCBCR601_444_8	0x218003D	YCBCR601 444 8-bit
GX_PIXEL_FORMAT_YCBCR601_422_8	0x210003E	YCBCR601 422 8-bit
GX_PIXEL_FORMAT_YCBCR601_411_8	0x20C003F	YCBCR601 411 8-bit
GX_PIXEL_FORMAT_YCBCR709_444_8	0x2180040	YCBCR709 444 8-bit
GX_PIXEL_FORMAT_YCBCR709_422_8	0x2100041	YCBCR709 422 8-bit
GX_PIXEL_FORMAT_YCBCR709_411_8	0x20C0042	YCBCR709 411 8-bit
GX_PIXEL_FORMAT_MONO10_PACKED	0x010C0004	GigE Vision specific format, Monochrome 10-bit packed
GX_PIXEL_FORMAT_MONO12_PACKED	0x010C0006	GigE Vision specific format, Monochrome 12-bit packed
GX_PIXEL_FORMAT_BAYER_BG10_PACKED	0x010C0029	GigE Vision specific format, Bayer Blue-Green 10-bit packed
GX_PIXEL_FORMAT_BAYER_BG12_PACKED	0x010C002D	GigE Vision specific format, Bayer Blue-Green 12-bit packed
GX_PIXEL_FORMAT_BAYER_GB10_PACKED	0x010C0028	GigE Vision specific format, Bayer Green-Blue 10-bit packed
GX_PIXEL_FORMAT_BAYER_GB12_PACKED	0x010C002C	GigE Vision specific format, Bayer Green-Blue 12-bit packed
GX_PIXEL_FORMAT_BAYER_GR10_PACKED	0x010C0026	GigE Vision specific format, Bayer Green-Red 10-bit packed
GX_PIXEL_FORMAT_BAYER_GR12_PACKED	0x010C002A	GigE Vision specific format, Bayer Green-Red 12-bit packed
GX_PIXEL_FORMAT_BAYER_RG10_PACKED	0x010C0027	GigE Vision specific format,

		Bayer Red-Green 10-bit packed
GX_PIXEL_FORMAT_BAYER_RG12_PACKED	0x010C002B	GigE Vision specific format, Bayer Red-Green 12-bit packed

#### 4.6. GX\_FRAME\_STATUS\_LIST

定义	值	解释
GX_FRAME_STATUS_SUCCESS	0	正常帧
GX_FRAME_STATUS_INCOMPLETE	-1	残帧

#### 4.7. GX\_FEATURE\_TYPE

定义	值	解释
GX_FEATURE_INT	0x10000000	整型功能
GX_FEATURE_FLOAT	0x20000000	浮点型功能
GX_FEATURE_ENUM	0x30000000	枚举型功能
GX_FEATURE_BOOL	0x40000000	布尔型功能
GX_FEATURE_STRING	0x50000000	字符串型功能
GX_FEATURE_BUFFER	0x60000000	寄存器型功能
GX_FEATURE_COMMAND	0x70000000	命令型功能

#### 4.8. GX\_BAYER\_CONVERT\_TYPE\_LIST

定义	值	解释
GX_RAW2RGB_NEIGHBOUR	0	邻域平均插值算法
GX_RAW2RGB_ADAPTIVE	1	边缘自适应插值算法
GX_RAW2RGB_NEIGHBOUR3	2	区域更大的邻域平均算法

#### 4.9. GX\_VALID\_BIT\_LIST

定义	值	解释
GX_BIT_0_7	0	bit 0~7
GX_BIT_1_8	1	bit 1~8
GX_BIT_2_9	2	bit 2~9
GX_BIT_3_10	3	bit 3~10
GX_BIT_4_11	4	bit 4~11
GX_BIT_5_12	5	bit 5~12
GX_BIT_6_13	6	bit 6~13
GX_BIT_7_14	7	bit 7~14
GX_BIT_8_15	8	bit 8~15

#### 4.10. GX\_IP\_CONFIGURE\_MODE\_LIST

定义	值	解释
GX_IP_CONFIGURE_LLA	4	启用 LLA 方式分配 IP 地址
GX_IP_CONFIGURE_STATIC_IP	5	使用静态 IP 方式配置 IP 地址
GX_IP_CONFIGURE_DHCP	6	启用 DHCP 模式，由 DHCP 服务器分配 IP 地址
GX_IP_CONFIGURE_DEFAULT	7	使用默认方式配置 IP 地址

#### 4.11. GX\_RESET\_DEVICE\_MODE

定义	值	解释
GX_MANUFACTURER_SPECIFIC_RECONNECT	1	设备重连，等同于软件接口关闭设备
GX_MANUFACTURER_SPECIFIC_RESET	2	设备复位，等同于给设备掉电上电一次

#### 4.12. COLOR\_TRANSFORM\_FACTOR

定义	默认值	范围	解释
float fGain00	1.0	-4.0 ~ 4.0	作用于红色像素的红色通道增益
float fGain01	0.0	-4.0 ~ 4.0	作用于红色像素的绿色通道增益
float fGain02	0.0	-4.0 ~ 4.0	作用于红色像素的蓝色通道增益
float fGain10	0.0	-4.0 ~ 4.0	作用于绿色像素的红色通道增益
float fGain11	1.0	-4.0 ~ 4.0	作用于绿色像素的绿色通道增益
float fGain12	0.0	-4.0 ~ 4.0	作用于绿色像素的蓝色通道增益
float fGain20	0.0	-4.0 ~ 4.0	作用于蓝色像素的红色通道增益
float fGain21	0.0	-4.0 ~ 4.0	作用于蓝色像素的绿色通道增益
float fGain22	1.0	-4.0 ~ 4.0	作用于蓝色像素的蓝色通道增益

#### 4.13. GX\_TL\_TYPE\_LIST

定义	值	解释
GX_TL_TYPE_UNKNOWN	0	未知类型
GX_TL_TYPE_USB	1	USB2.0 类型
GX_TL_TYPE_GEV	2	千兆网类型
GX_TL_TYPE_U3V	4	USB3.0 类型
GX_TL_TYPE_CXP	8	CXP 类型

#### 4.14. GX\_ACTUAL\_BITS

定义	值	解释
GX_ACTUAL_BITS_8	8	8bit
GX_ACTUAL_BITS_10	10	10bit
GX_ACTUAL_BITS_12	12	12bit



GX_ACTUAL_BITS_14	14	14bit
GX_ACTUAL_BITS_16	16	16bit

#### 4.15. GX\_PIXEL\_COLOR\_FILTER

定义	值	解释
GX_NONE	0	非 Bayer 格式
GX_BAYERRG	1	BayerRG
GX_BAYERGB	2	BayerGB
GX_BAYERGR	3	BayerGR
GX_BAYERBG	4	BayerBG

#### 4.16. GX\_LOG\_TYPE\_LIST

定义	值	解释
GX_LOG_TYPE_OFF	0x00000000	不生成任何类型日志
GX_LOG_TYPE_FATAL	0x00000001	生成 FATAL 类型日志
GX_LOG_TYPE_ERROR	0x00000010	生成 ERROR 类型日志
GX_LOG_TYPE_WARN	0x00000100	生成 WARNING 类型日志
GX_LOG_TYPE_INFO	0x00001000	生成 INFO 类型日志
GX_LOG_TYPE_DEBUG	0x00010000	生成 DEBUG 类型日志
GX_LOG_TYPE_TRACE	0x00100000	生成 TRACE 类型日志

#### 4.17. GX\_STATIC\_DEFECT\_CORRECTION

定义	解释
uint32_t ui32ImgWidth	当前图像的宽度
uint32_t ui32ImgHeight	当前图像的高度
uint32_t ui32ImgOffsetX	当前图像的水平偏移
uint32_t ui32ImgOffsetY	当前图像的垂直偏移
uint32_t ui32ImgWidthMax	设备支持的最大宽度
GX_PIXEL_COLOR_FILTER emBayerType	图像像素格式
GX_ACTUAL_BITS emActualBits	图像像素实际有效位

#### 4.18. GX\_IMAGE\_INFO

定义	解释
void *pBuffer	图像 buffer 指针
uint16_t nWidth	图像的宽度
uint16_t nHeight	图像的高度
GX_PIXEL_FORMAT_ENTRY emSrcFormat	图像的格式



## 5. 句柄类型定义

定义	解释
GX_DEVICE_OFFLINE_CALLBACK_HANDLE	掉线事件句柄，用户调用 IGXDevice::RegisterDeviceOfflineCallback 接口获取
GX_FEATURE_CALLBACK_HANDLE	属性更新事件句柄，用户调用 IGXFeatureControl::RegisterFeatureCallback 接口获取

## 6. 回调事件虚基类

类型	定义
采集回调	<a href="#">ICaptureEventHandler</a>
设备掉线事件	<a href="#">IDeviceOfflineEventHandler</a>
属性更新事件	<a href="#">IFeatureEventHandler</a>

## 7. 模块接口定义

### 7.1. IGXFactory

负责全局资源初始化、枚举设备、打开设备。

**接口列表：**

<a href="#">GetInstance</a>	静态函数，全局可调用。返回 IGXFactory 对象实例
<a href="#">Init</a>	初始化库资源
<a href="#">Uninit</a>	释放库资源
<a href="#">UpdateDeviceList</a>	枚举设备（对于千兆网设备是子网枚举）
<a href="#">UpdateAllDeviceList</a>	枚举设备（对于千兆网设备是全网枚举）
<a href="#">UpdateAllDeviceListEx</a>	枚举特定类型设备（对于千兆网设备是全网枚举）
<a href="#">OpenDeviceByIP</a>	依靠 IP 地址打开设备
<a href="#">OpenDeviceByMAC</a>	依靠 MAC 地址打开设备
<a href="#">OpenDeviceBySN</a>	依靠 SN 打开设备
<a href="#">OpenDeviceByUserID</a>	依靠 UserID 打开设备
<a href="#">GigElpConfiguration</a>	设置相机静态（永久）IP 地址
<a href="#">GigEForceIp</a>	执行 Force IP 操作
<a href="#">GigEResetDevice</a>	执行设备重连或复位操作
<a href="#">CreateImageFormatConvert</a>	获取图像格式转换指针
<a href="#">CreateImageSpecialFunction</a>	获取特殊图像接口
<a href="#">GetAllInterfaceInfo</a>	获取所有采集卡信息
<a href="#">GetInterface</a>	获取索引值对应的采集卡指针对象，下标从 1 开始计数
<a href="#">CreateImageProcess</a>	创建并返回图像处理对象实例
<a href="#">SetLogType</a>	设置生成日志的类型
<a href="#">GetLogType</a>	获取生成日志的类型

#### 7.1.1. GetInstance

**接口定义：** static IGXFactory& GetInstance()

**功能描述：** 静态函数，全局可调用。返回 IGXFactory 对象实例。

#### 7.1.2. Init

**接口定义：** void Init(void)

**功能描述：** 初始化库资源。

### 7.1.3. Uninit

**接口定义:** void Uninit(void)

**功能描述:** 释放库资源。

### 7.1.4. UpdateDeviceList

**接口定义:** void UpdateDeviceList(uint32\_t nTimeout, GxIAPICPP::gxdeviceinfo\_vector& vectorDeviceInfo)

**功能描述:** 枚举设备 (对于千兆网设备是子网枚举)。

参数 1: 超时时间, 单位 ms。

参数 2: 返回的设备信息列表。

### 7.1.5. UpdateAllDeviceList

**接口定义:** void UpdateAllDeviceList(uint32\_t nTimeout, GxIAPICPP::gxdeviceinfo\_vector& vectorDeviceInfo)

**功能描述:** 枚举设备 (对于千兆网设备是全网枚举)。

参数 1: 超时时间, 单位 ms。

参数 2: 返回的设备信息列表。

### 7.1.6. UpdateAllDeviceListEx

**接口定义:** void UpdateAllDeviceListEx(uint32\_t nTLType, uint32\_t nTimeout,  
GxIAPICPP::gxdeviceinfo\_vector& vectorDeviceInfo)

**功能描述:** 枚举特定类型设备 (对于千兆网设备是全网枚举)

参数 1: 枚举特定类型的设备, 参考 [GX\\_TL\\_TYPE\\_LIST](#)。

参数 2: 超时时间, 单位 ms。

参数 3: 返回的设备信息列表。

### 7.1.7. OpenDeviceByIP/MAC/SN/UserID

以按 IP 方式打开设备为例, 其他接口类似。

**接口定义:** CGXDevicePointer OpenDeviceByIP(const GxIAPICPP::gxstring& strIP,  
[GX\\_ACCESS\\_MODE](#) emAccessMode)

**功能描述:** 依靠 IP 地址打开设备。

参数 1: 设备 IP 地址。

参数 2: 打开设备方式。

返回值: CGXDevicePointer 对象实例。

### 7.1.8. GigElpConfiguration

**接口定义:** void GigElpConfiguration(const GxIAPICPP::gxstring& strDeviceMacAddress,  
GX\_IP\_CONFIGURE\_MODE emIpConfigMode,  
const GxIAPICPP::gxstring& strIpAddress,  
const GxIAPICPP::gxstring& strSubnetMask,  
const GxIAPICPP::gxstring& strDefaultGateway,  
const GxIAPICPP::gxstring& strUserID)

**功能描述：**设置相机静态（永久）IP 地址。

参数 1：设备 MAC 地址。

参数 2：IP 配置方式。

参数 3：待设置的 IP 地址值。

参数 4：待设置的子网掩码值。

参数 5：待设置的默认网关值。

参数 6：待设置的用户自定义名称。

返回值：无

### 7.1.9. GigEForceIp

**接口定义：** void GXGigEForceIp(const GxIAPICPP::gxstring& pszDeviceMacAddress,  
const GxIAPICPP::gxstring& strIpAddress,  
const GxIAPICPP::gxstring& strSubnetMask,  
const GxIAPICPP::gxstring& strDefaultGateway)

**功能描述：**执行 Force IP 操作。

参数 1：设备 MAC 地址。

参数 2：待设置的 IP 地址值。

参数 3：待设置的子网掩码值。

参数 4：待设置的默认网关值。

返回值：无

### 7.1.10. GigEResetDevice

**接口定义：** void GigEResetDevice(const GxIAPICPP::gxstring& strDeviceMacAddress,  
[GX\\_RESET\\_DEVICE\\_MODE](#) ui32FeatureInfo)

**功能描述：**执行设备重连或复位操作。详见 2.2 章节。

设备重连通常应用在调试千兆网相机时，设备已经被打开，此时程序异常，而后立即重新打开设备报错（因为调试心跳 5 分钟，设备依然处于打开状态），这时可通过设备重连功能，使设备处于未打开状态，而后再次打开设备即可成功。

设备复位通常应用在相机状态异常，此时设备重连功能也无法生效，也不具备给设备重新上电的条件，可尝试使用设备复位功能，使设备掉电再上电。设备复位后，需要重新执行枚举、打开设备操作。

**注意：**

- 1) 设备复位时间需要 1s 左右，因此需要确保 1s 后再调用枚举接口；
- 2) 如果设备正在正常采集，禁止使用设备复位和重连功能，否则会导致设备掉线。

参数 1：设备 MAC 地址。

参数 2：重置设备模式。

返回值：无。

### 7.1.11. CreateImageFormatConvert

**接口定义:** CGXImageFormatConvertPointer CreateImageFormatConvert()

**功能描述:** 创建并返回图像转换对象实例。

### 7.1.12. CreateImageSpecialFunction

**接口定义:** CGXImageSpecialFunctionPointer CreateImageSpecialFunction()

**功能描述:** 创建并返回图像特殊对象实例。

### 7.1.13. GetAllInterfaceInfo

**接口定义:** void GetAllInterfaceInfo(GxIAPICPP::gxinterfaceinfo\_vector& vectorInterfaceInfo)

**功能描述:** 获取所有采集卡信息。

参数 1: 返回的采集卡信息列表。

### 7.1.14. GetInterface

**接口定义:** CGXInterfacePointer GetInterface(const uint32\_t nIndex)

**功能描述:** 获取用户指定采集卡指针对象。

参数 1: 采集卡索引, 下标从 1 开始计数。

### 7.1.15. CreateImageProcess

**接口定义:** CGXImageProcessPointer CreateImageProcess()

**功能描述:** 创建并返回图像处理对象实例。

### 7.1.16. SetLogType

**接口定义:** void SetLogType(const uint32\_t ui32LogType)

**功能描述:** 设置生成的日志类型。

参数 1: 待设置的日志类型, 参考 [GX\\_LOG\\_TYPE\\_LIST](#)。

返回值: 无。

### 7.1.17. GetLogType

**接口定义:** void GetLogType(uint32\_t& ui32Value)

**功能描述:** 返回生成的日志类型。

参数 1: 返回的日志类型, 参考 [GX\\_LOG\\_TYPE\\_LIST](#)。

返回值: 无。

## 7.2. IGXInterfaceInfo

Interface 信息存储单元, 存储所有 Interface 信息, 可以枚举后通过 GetAllInterfaceInfo()获取

**接口列表:**

<a href="#">GetType</a>	获取 Interface 类型, 比如 USB2.0、USB3.0、GigE、CXP
<a href="#">GetModelName</a>	获取 Interface 名称

<a href="#">GetDisplayName</a>	获取 Interface 展示名称
<a href="#">GetSerialNumber</a>	获取 Interface 序列号
<a href="#">GetInterfaceID</a>	获取 Interface ID, 非 CXP 类型此值为 0
<a href="#">GetInitFlag</a>	获取 Interface 初始化标志, 非 CXP 类型此值为 0
<a href="#">GetVendorName</a>	获取 Interface 厂商名称

### 7.2.1. GetType

**接口定义:** [GX\\_TL\\_TYPE\\_LIST](#) GetType() const

**功能描述:** 获取当前的采集卡类型。

### 7.2.2. GetModelName

**接口定义:** GxIAPICPP::gxstring GetModelName() const

**功能描述:** 获取采集卡名称。

### 7.2.3. GetDisplayName

**接口定义:** GxIAPICPP::gxstring GetDisplayName() const

**功能描述:** 获取采集卡显示名称。

### 7.2.4. GetSerialNumber

**接口定义:** GxIAPICPP::gxstring GetSerialNumber() const

**功能描述:** 获取采集卡序列号。

### 7.2.5. GetInterfaceID

**接口定义:** GxIAPICPP::gxstring GetInterfaceID() const

**功能描述:** 获取InterfaceID, 非CXP类型此值为0。

### 7.2.6. GetInitFlag

**接口定义:** uint32\_t GetInitFlag() const

**功能描述:** 获取Interface初始化标识, 非CXP类型此值为0。

### 7.2.7. GetVendorName

**接口定义:** GxIAPICPP::gxstring GetVendorName() const

**功能描述:** 获取采集卡厂商名称。

## 7.3. IGXInterface

Interface 功能指针, 可以枚举后通过 GetInterface()获取。

**接口列表:**

<a href="#">GetInterfaceInfo</a>	获取 Interface 层信息指针
<a href="#">GetAllDeviceInfo</a>	获取 Interface 层下所有连接的设备列表
<a href="#">GetFeatureControl</a>	获取 Interface 属性控制器指针

### 7.3.1. GetInterfaceInfo

**接口定义：** IGXInterfaceInfoPointer GetInterfaceInfo()

**功能描述：** 获取Interface层信息指针。

### 7.3.2. GetAllDeviceInfo

**接口定义：** void GetAllDeviceInfo( GxIAPICPP::gxdeviceinfo\_vector& vectorDeviceInfo)

**功能描述：** 获取当前Interface层下所有连接的设备列表。

参数 1：返回的设备列表。

### 7.3.3. GetFeatureControl

**接口定义：** CGXFeatureControlPointer GetFeatureControl()

**功能描述：** 获取Interface层属性控制器指针。

## 7.4. IGXDeviceInfo

设备信息存储单元，存储所有设备信息。可以通过枚举接口 IGXFactory::UpdateDeviceList、IGXFactory::UpdateAllDeviceList、IGXFactory::UpdateAllDeviceListEx 获取。

**接口列表：**

<a href="#">GetVendorName</a>	获取厂商名称
<a href="#">GetModelName</a>	获取设备名称
<a href="#">GetSN</a>	获取设备序列号
<a href="#">GetDisplayName</a>	获取设备展示名称
<a href="#">GetDeviceID</a>	获取 DeviceID，此 ID 也可用来唯一标识设备
<a href="#">GetUserID</a>	获取用户自定义名称
<a href="#">GetAccessStatus</a>	获取设备当前可访问状态
<a href="#">GetDeviceClass</a>	获取设备类型，比如 USB2.0、USB3.0、Gige、CXP
<a href="#">GetMAC</a>	获取设备 MAC 地址，非千兆网设备此值为空
<a href="#">GetIP</a>	获取设备 IP 地址，非千兆网设备此值为空
<a href="#">GetSubnetMask</a>	获取设备子网掩码，非千兆网设备此值为空
<a href="#">GetGateway</a>	获取设备默认网关，非千兆网设备此值为空
<a href="#">GetNICMAC</a>	获取设备对应网卡 MAC 地址，非千兆网设备此值为空
<a href="#">GetNICIP</a>	获取设备对应网卡 IP 地址，非千兆网设备此值为空
<a href="#">GetNICSubnetMask</a>	获取设备对应网卡子网掩码，非千兆网设备此值为空
<a href="#">GetNICGateway</a>	获取设备对应网卡默认网关，非千兆网设备此值为空
<a href="#">GetNICDescription</a>	获取设备对应网卡描述信息，非千兆网设备此值为空



#### 7.4.1. GetVendorName

**接口定义:** GxIAPICPP::gxstring GetVendorName() const

**功能描述:** 获取厂商名称。

#### 7.4.2. GetModelName

**接口定义:** GxIAPICPP::gxstring GetModelName() const

**功能描述:** 获取设备型号名称。

#### 7.4.3. GetSN

**接口定义:** GxIAPICPP::gxstring GetSN() const

**功能描述:** 获取设备序列号。

#### 7.4.4. GetDisplayName

**接口定义:** GxIAPICPP::gxstring GetDisplayName() const

**功能描述:** 获取设备展示名称。

#### 7.4.5. GetDeviceID

**接口定义:** GxIAPICPP::gxstring GetDeviceID() const

**功能描述:** 获取DeviceID，此ID也可用来唯一标识设备。

#### 7.4.6. GetUserID

**接口定义:** GxIAPICPP::gxstring GetUserID() const

**功能描述:** 获取用户自定义名称。

#### 7.4.7. GetAccessStatus

**接口定义:** [GX\\_ACCESS\\_STATUS](#) GetAccessStatus() const

**功能描述:** 获取设备当前可访问状态。

#### 7.4.8. GetDeviceClass

**接口定义:** [GX\\_DEVICE\\_CLASS\\_LIST](#) GetDeviceClass() const

**功能描述:** 获取设备类型，比如 USB2.0、USB3.0、GigE。

#### 7.4.9. GetMAC

**接口定义:** GxIAPICPP::gxstring GetMAC() const

**功能描述:** 获取设备MAC地址，非千兆网设备此值为空。

#### 7.4.10. GetIP

**接口定义:** GxIAPICPP::gxstring GetIP() const

**功能描述:** 获取设备IP地址，非千兆网设备此值为空。

#### 7.4.11. GetSubnetMask

**接口定义:** GxIAPICPP::gxstring GetSubnetMask() const

**功能描述:** 获取设备子网掩码，非千兆网设备此值为空。

#### 7.4.12. GetGateway

**接口定义:** GxIAPICPP::gxstring GetGateway() const

**功能描述:** 获取设备默认网关，非千兆网设备此值为空。

#### 7.4.13. GetNICMAC

**接口定义:** GxIAPICPP::gxstring GetNICMAC() const

**功能描述:** 获取设备对应网卡MAC地址，非千兆网设备此值为空。

#### 7.4.14. GetNICIP

**接口定义:** GxIAPICPP::gxstring GetNICIP() const

**功能描述:** 获取设备对应网卡IP地址，非千兆网设备此值为空。

#### 7.4.15. GetNICSubnetMask

**接口定义:** GxIAPICPP::gxstring GetNICSubnetMask() const

**功能描述:** 获取设备对应网卡子网掩码，非千兆网设备此值为空。

#### 7.4.16. GetNICGateway

**接口定义:** GxIAPICPP::gxstring GetNICGateway() const

**功能描述:** 获取设备对应网卡默认网关，非千兆网设备此值为空。

#### 7.4.17. GetNICDescription

**接口定义:** GxIAPICPP::gxstring GetNICDescription() const

**功能描述:** 获取设备对应网卡描述信息，非千兆网设备此值为空。

### 7.5. IGXDevice

设备对象，负责获取设备信息、获取属性控制器（控制通道）、获取流对象（采集通道）、获取设备掉线事件。可以通过 IGXFactory::OpenDeviceByIP/MAC/SN/UserID 来获取。

**接口列表:**

<a href="#">GetDeviceInfo</a>	获取设备信息对象
<a href="#">GetStreamCount</a>	获取流通道个数
<a href="#">OpenStream</a>	用户指定流通道序号打开某个流，获取流通道对象
<a href="#">GetFeatureControl</a>	获取本地设备层属性控制器，通过此控制器可以控制本地设备所有功能
<a href="#">GetRemoteFeatureControl</a>	获取远端设备层属性控制器，通过此控制器可以控制远端设备所有功能
<a href="#">GetEventNumInQueue</a>	获取设备事件队列长度，表示当前事件队列里面有多少缓存数据
<a href="#">FlushEvent</a>	清空设备事件队列

<a href="#">RegisterDeviceOfflineCallback</a>	注册掉线回调函数
<a href="#">UnregisterDeviceOfflineCallback</a>	注销掉线回调函数
<a href="#">CreateImageProcessConfig</a>	创建图像处理配置参数对象
<a href="#">ExportConfigFile</a>	导出相机当前配置参数到文本文件
<a href="#">ImportConfigFile</a>	将配置文件中的参数导入到相机
<a href="#">Close</a>	关闭设备
<a href="#">GetParentInterface</a>	获取设备对应父指针

#### 7.5.1. GetDeviceInfo

**接口定义：** const CGXDeviceInfo& GetDeviceInfo()

**功能描述：** 获取设备信息对象。

#### 7.5.2. GetStreamCount

**接口定义：** uint32\_t GetStreamCount()

**功能描述：** 获取流通道个数。

#### 7.5.3. OpenStream

**接口定义：** CGXStreamPointer OpenStream(uint32\_t nStreamID)

**功能描述：** 用户指定流通道序号打开某个流，获取流通道对象。

**参数 1：** nStreamID          流通道序号，从 0 开始。

#### 7.5.4. GetFeatureControl

**接口定义：** CGXFeatureControlPointer GetFeatureControl()

**功能描述：** 获取本地设备层属性控制器，通过此控制器可以控制本地设备所有功能。

#### 7.5.5. GetRemoteFeatureControl

**接口定义：** CGXFeatureControlPointer GetRemoteFeatureControl()

**功能描述：** 获取远端设备层属性控制器，通过此控制器可以控制远端设备所有功能。

#### 7.5.6. GetEventNumInQueue

**接口定义：** uint32\_t GetEventNumInQueue()

**功能描述：** 获取设备事件队列长度，表示当前事件队列里面有多少缓存数据。

#### 7.5.7. FlushEvent

**接口定义：** void FlushEvent()

**功能描述：** 清空设备事件队列。

#### 7.5.8. RegisterDeviceOfflineCallback

**接口定义：** [GX\\_DEVICE\\_OFFLINE\\_CALLBACK\\_HANDLE](#) RegisterDeviceOfflineCallback(IDevice  
OfflineEventHandler\* pEventHandler, void\* pUserParam)

**功能描述：** 注册掉线回调函数。

### 7.5.9. UnregisterDeviceOfflineCallback

**接口定义:** void UnregisterDeviceOfflineCallback([GX\\_DEVICE\\_OFFLINE\\_CALLBACK\\_HANDLE](#) hCallBack)

**功能描述:** 注销掉线回调函数。

### 7.5.10. CreateImageProcessConfig

**接口定义:** CImageProcessConfigPointer CreateImageProcessConfig()

**功能描述:** 创建图像处理配置参数对象。

### 7.5.11. ExportConfigFile

**接口定义:** void ExportConfigFile(const GxIAPICPP::gxstring& strFilePath)

**功能描述:** 导出相机当前配置参数到文本文件。

### 7.5.12. ImportConfigFile

**接口定义:** void ImportConfigFile(const GxIAPICPP::gxstring& strFilePath)

**功能描述:** 将配置文件中的参数导入到相机。

### 7.5.13. Close

**接口定义:** void Close()

**功能描述:** 关闭设备。

### 7.5.14. GetParentInterface

**接口定义:** CGXInterfacePointer GetParentInterface()

**功能描述:** 获取当前设备对应父 Interface 指针。

## 7.6. IGXFeatureControl

属性控制器，设备对象 CGXDevicePointer 和流对象 CGXStreamPointer 都有各自的属性控制器，通过属性控制器可以控制相机的所有功能属性，比如通过其获取一个整型功能控制器，然后通过整型功能控制器再进行读写控制。还可以给属性注册属性更新回调函数。可以通过 IGXDevice::GetFeatureControl 或者 IGXDevice::GetRemoteFeatureControl 或者 IGXStream::GetFeatureControl 三种方式获取三种不同类型的属性控制器。

**接口列表:**

[GetFeatureNameList](#)

获取当前控制器所支持的所有功能名称字符串列表

[GetFeatureType](#)

获取当前字符串对应功能的数据类型：整型、浮点型、枚举型等，详见 [GX\\_FEATURE\\_TYPE](#) 定义

[IsImplemented](#)

查询当前属性控制是否支持此功能

[IsReadable](#)

查询当前功能是否可读

[IsWritable](#)

查询当前功能是否可写

<a href="#">GetIntFeature</a>	获取一个整型控制器
<a href="#">GetFloatFeature</a>	获取一个浮点型控制器
<a href="#">GetEnumFeature</a>	获取一个枚举型控制器
<a href="#">GetBoolFeature</a>	获取一个布尔型控制器
<a href="#">GetStringFeature</a>	获取一个字符串型控制器
<a href="#">GetCommandFeature</a>	获取一个命令型控制器
<a href="#">GetRegisterFeature</a>	获取一个寄存器类型控制器
<a href="#">RegisterFeatureCallback</a>	注册功能属性更新回调函数
<a href="#">UnregisterFeatureCallback</a>	注销功能属性更新回调函数
<a href="#">ReadPort</a>	读指定寄存器的值
<a href="#">WritePort</a>	写指定寄存器的值
<a href="#">ReadPortStacked</a>	批量读取指定的寄存器值
<a href="#">WritePortStacked</a>	批量写入指定的寄存器值
<a href="#">FeatureSave</a>	保存用户参数组
<a href="#">FeatureSaveW</a>	保存用户参数组
<a href="#">FeatureLoad</a>	加载用户参数组
<a href="#">FeatureLoadW</a>	加载用户参数组

#### 7.6.1. GetFeatureNameList

**接口定义:** void GetFeatureNameList(GxIAPICPP::gxstring\_vector& vectorFeatureNameList)

**功能描述:** 获取当前控制器所支持的所有功能名称字符串列表。

#### 7.6.2. GetFeatureType

**接口定义:** [GX\\_FEATURE\\_TYPE](#) GetFeatureType(const GxIAPICPP::gxstring& strFeatureName)

**功能描述:** 获取当前字符串对应功能的数据类型: 整型、浮点型、枚举型等, 详见 [GX\\_FEATURE\\_TYPE](#) 定义。

#### 7.6.3. IsImplemented

**接口定义:** bool IsImplemented(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 查询当前属性控制是否支持某功能。

#### 7.6.4. IsReadable

**接口定义:** bool IsReadable(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 查询当前功能是否可读。

#### 7.6.5. IsWritable

**接口定义:** bool IsWritable(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 查询当前功能是否可写。

#### 7.6.6. GetIntFeature

**接口定义:** CIntFeaturePointer GetIntFeature(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 获取一个整型控制器。

#### 7.6.7. GetFloatFeature

**接口定义:** CFloatFeaturePointer GetFloatFeature(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 获取一个浮点型控制器。

#### 7.6.8. GetEnumFeature

**接口定义:** CEnumFeaturePointer GetEnumFeature(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 获取一个枚举型控制器。

#### 7.6.9. GetBoolFeature

**接口定义:** CBoolFeaturePointer GetBoolFeature(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 获取一个布尔型控制器。

#### 7.6.10. GetStringFeature

**接口定义:** CStringFeaturePointer GetStringFeature(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 获取一个字符串型控制器。

#### 7.6.11. GetCommandFeature

**接口定义:** CCommandFeaturePointer GetCommandFeature(const GxIAPICPP::gxstring&strFeatureName)

**功能描述:** 获取一个命令型控制器。

#### 7.6.12. GetRegisterFeature

**接口定义:** CRegisterFeaturePointer GetRegisterFeature(const GxIAPICPP::gxstring  
&strFeatureName)

**功能描述:** 获取一个寄存器类型控制器。

#### 7.6.13. RegisterFeatureCallback

**接口定义:** [GX\\_FEATURE\\_CALLBACK\\_HANDLE](#) RegisterFeatureCallback(  
const GxIAPICPP::gxstring&strFeatureName, IFeatureEventHandler\* pEventHandler, void\* pUserParam)

**功能描述:** 注册功能属性更新回调函数。

#### 7.6.14. UnregisterFeatureCallback

**接口定义:** void UnregisterFeatureCallback([GX\\_FEATURE\\_CALLBACK\\_HANDLE](#) hCallback)

**功能描述:** 注销功能属性更新回调函数。

#### 7.6.15. ReadPort

**接口定义:** void ReadPort(uint64\_t uiAddress, void \*pBuffer, size\_t \*piSize)

**功能描述:** 读取相机指定寄存器的值。

**注意:** 读取到千兆网相机属性值为大端数据。

### 7.6.16. WritePort

**接口定义:** void WritePort(uint64\_t uiAddress, const void \*pBuffer, size\_t \*piSize)

**功能描述:** 设置相机指定寄存器的值。

**注意:**

- 1) 千兆网相机设置的属性值需转换为大端后设置。
- 2) 调用当前接口后, 使用 GetEnumFeature、GetIntFeature、GetBoolFeature 等接口获取到的节点值仍为修改前值, 可使用 ReadPort 或 ReadPortStacked 接口获取最新的属性值。

### 7.6.17. ReadPortStacked

**接口定义:** void ReadPortStacked(GX\_REGISTER\_STACK\_ENTRY\* pstEntries, size\_t \*piNumEntries)

**功能描述:** 批量读取相机指定的多个寄存器的值, 当前仅支持整型 (4 字节)。

**注意:** 获取千兆网相机属性值为大端数据。

### 7.6.18. WritePortStacked

**接口定义:** void WritePortStacked(const GX\_REGISTER\_STACK\_ENTRY\* pstEntries, size\_t \*piNumEntries)

**功能描述:** 批量设置相机指定的多个寄存器的值, 当前仅支持整型 (4 字节)。

**注意:**

- 1) 千兆网相机设置的属性值需转换为大端后设置。
- 2) 调用当前接口后, 使用 GetEnumFeature、GetIntFeature、GetBoolFeature 等接口获取到的节点值仍为修改前值, 可使用 ReadPort 或 ReadPortStacked 接口获取最新的属性值。

### 7.6.19. FeatureSave

**接口定义:** void FeatureSave(const GxIAPICPP::gxstring& strFilePath)

**功能描述:** 保存当前采集卡或相机配置参数到文本文件。

### 7.6.20. FeatureSaveW

**接口定义:** void FeatureSaveW(const wchar\_t\* pchWFilePath)

**功能描述:** 保存当前采集卡或相机配置参数到文本文件。

### 7.6.21. FeatureLoad

**接口定义:** FeatureLoad(const GxIAPICPP::gxstring& strFilePath, bool bVerify = false)

**功能描述:** 配置文件中的参数加载到采集卡或相机。

### 7.6.22. FeatureLoadW

**接口定义:** FeatureLoadW(const wchar\_t\* pchWFilePath, bool bVerify = false)

**功能描述:** 配置文件中的参数加载到采集卡或相机。



## 7.7. IIntFeature

整型功能节点控制器，与某个整型功能节点对应，通过 IGXFeatureControl::GetIntFeature 获取。

### 接口列表：

<a href="#">GetMin</a>	获取最小值
<a href="#">GetMax</a>	获取最大值
<a href="#">GetInc</a>	获取步长
<a href="#">GetValue</a>	获取当前值
<a href="#">SetValue</a>	设置当前值

### 7.7.1. GetMin

**接口定义：** int64\_t GetMin()

**功能描述：** 获取最小值。

### 7.7.2. GetMax

**接口定义：** int64\_t GetMax()

**功能描述：** 获取最大值。

### 7.7.3. GetInc

**接口定义：** int64\_t GetInc()

**功能描述：** 获取步长。

### 7.7.4. GetValue

**接口定义：** int64\_t GetValue()

**功能描述：** 获取当前值。

### 7.7.5. SetValue

**接口定义：** void SetValue(int64\_t nValue)

**功能描述：** 设置当前值。

## 7.8. IFloatFeature

浮点型功能节点控制器，与某个浮点型功能节点对应，通过 IGXFeatureControl::GetFloatFeature 获取。

### 接口列表：

<a href="#">GetMin</a>	获取最小值
<a href="#">GetMax</a>	获取最大值
<a href="#">HasInc</a>	查询是否有步长
<a href="#">GetInc</a>	获取步长。如果 HasInc 接口返回 false，调用此接口返回 0
<a href="#">GetUnit</a>	获取单位
<a href="#">GetValue</a>	获取当前值
<a href="#">SetValue</a>	设置当前值



### 7.8.1. GetMin

**接口定义:** double GetMin()

**功能描述:** 获取最小值。

### 7.8.2. GetMax

**接口定义:** double GetMax()

**功能描述:** 获取最大值。

### 7.8.3. HasInc

**接口定义:** bool HasInc()

**功能描述:** 查询是否有步长。

### 7.8.4. GetInc

**接口定义:** double GetInc()

**功能描述:** 获取步长。如果HasInc接口返回false, 调用此接口返回0。

### 7.8.5. GetUnit

**接口定义:** GxIAPICPP::gxstring GetUnit()

**功能描述:** 获取单位。

### 7.8.6. GetValue

**接口定义:** double GetValue()

**功能描述:** 获取当前值。

### 7.8.7. SetValue

**接口定义:** void SetValue(double dValue)

**功能描述:** 设置当前值。

## 7.9. IEnumFeature

枚举型功能节点控制器, 与某个枚举型功能节点对应, 通过 IGXFeatureControl::GetEnumFeature 获取。

**接口列表:**

<a href="#">GetEnumEntryList</a>	获取支持的枚举项列表
<a href="#">GetValue</a>	获取当前的枚举项值
<a href="#">SetValue</a>	设置当前的枚举项值

### 7.9.1. GetEnumEntryList

**接口定义:** GxIAPICPP::gxstring\_vector GetEnumEntryList()

**功能描述:** 获取支持的枚举项列表。

### 7.9.2. GetValue

**接口定义:** GxIAPICPP::gxstring GetValue()

**功能描述:** 获取当前的枚举项值。

### 7.9.3. SetValue

**接口定义:** void SetValue(const GxIAPICPP::gxstring&strValue)

**功能描述:** 设置当前的枚举项值。

## 7.10. IBoolFeature

布尔型功能节点控制器,与某个布尔型功能节点对应,通过 IGXFeatureControl::GetBoolFeature 获取。

**接口列表:**

<a href="#">GetValue</a>	获取当前值
<a href="#">SetValue</a>	设置当前值

### 7.10.1. GetValue

**接口定义:** bool GetValue()

**功能描述:** 获取当前值。

### 7.10.2. SetValue

**接口定义:** void SetValue(bool bValue)

**功能描述:** 设置当前值。

## 7.11. IStringFeature

字符串型功能节点控制器,与某个字符串功能节点对应,通过 IGXFeatureControl::GetStringFeature 获取。

**接口列表:**

<a href="#">GetValue</a>	获取当前值
<a href="#">SetValue</a>	设置当前值
<a href="#">GetStringMaxLength</a>	获取可设置的最大长度,不包含结束符

### 7.11.1. GetValue

**接口定义:** GxIAPICPP::gxstring GetValue()

**功能描述:** 获取当前值。

### 7.11.2. SetValue

**接口定义:** void SetValue(const GxIAPICPP::gxstring&strValue)

**功能描述:** 设置当前值。

### 7.11.3. GetStringMaxLength

**接口定义:** int64\_t GetStringMaxLength()

**功能描述:** 获取可设置的最大长度，不包含结束符。

## 7.12. ICommandFeature

命令型功能节点控制器，与某个命令型功能节点对应，通过 IGXFeatureControl::GetCommandFeature 获取。

**接口列表:**

[Execute](#) 执行命令

### 7.12.1. Execute

**接口定义:** void Execute()

**功能描述:** 执行命令。

## 7.13. IRegisterFeature

寄存器型功能节点控制器，与某个寄存器型功能节点对应，通过 IGXFeatureControl::GetRegisterFeature 获取。

**接口列表:**

[GetLength](#) 获取 buffer 的长度。用户读取此长度用来读写 buffer

[GetBuffer](#) 获取 buffer 值。输入参数 ptrBuffer 的长度必须等于 GetLength 接口获取的长度

[SetBuffer](#) 设置 buffer 值。输入参数 ptrBuffer 的长度必须等于 GetLength 接口获取的长度

### 7.13.1. GetLength

**接口定义:** int64\_t GetLength()

**功能描述:** 获取buffer的长度。用户读取此长度用来读写buffer。

### 7.13.2. GetBuffer

**接口定义:** void GetBuffer(uint8\_t\* pBuffer, int64\_t nLength)

**功能描述:** 获取buffer值。输入参数pBuffer用户空间长度必须等于GetLength接口获取的长度。

### 7.13.3. SetBuffer

**接口定义:** void SetBuffer(uint8\_t\* pBuffer, int64\_t nLength)

**功能描述:** 设置buffer值。输入参数pBuffer用户空间长度必须等于GetLength接口获取的长度。

## 7.14. IGXStream

代表流对象，负责采集图像相关操作，比如注册注销回调函数，采集单帧等。通过 IGXDevice::OpenStream 接口获取。

### 接口列表:

<a href="#">StartGrab</a>	开启流层采集，不论是回调采集还是单帧采集之前必须先调用此接口
<a href="#">StopGrab</a>	停止流层采集
<a href="#">RegisterCaptureCallback</a>	注册用户委托采集函数
<a href="#">UnregisterCaptureCallback</a>	注销采集回调函数
<a href="#">GetImage</a>	采集单帧
<a href="#">DQBuf</a>	零拷贝方式采集单帧
<a href="#">QBuf</a>	将 ImageData 交还给采集系统
<a href="#">GetFeatureControl</a>	获取流层属性控制器，用于控制流层的功能属性
<a href="#">FlushQueue</a>	清空采集队列中的缓存图像
<a href="#">SetAcquisitionBufferNumber</a>	设置采集 buffer 个数
<a href="#">Close</a>	关闭流对象
<a href="#">GetOptimalPacketSize</a>	获取设备最优包长
<a href="#">GetPayloadSize</a>	获取设备流层数据块大小

#### 7.14.1. StartGrab

**接口定义:** void StartGrab()

**接口描述:** 开启流层采集，不论是回调采集还是单帧采集，之前必须先调用 StartGrab 接口，先启动流层采集，包含启动 TLClass 层的采集线程和资源申请等动作。

#### 7.14.2. StopGrab

**接口定义:** void StopGrab()

**接口描述:** 停止流层采集，包括停止 TLClass 层的采集线程和释放资源等动作。

#### 7.14.3. RegisterCaptureCallback

**接口定义:** void RegisterCaptureCallback(ICaptureEventHandler\* pEventHandler, void \*pUserParam)

**接口描述:** 注册用户采集回调函数。参数 1 是用户回调对象；参数 2 是用户私有参数。

#### 功能要求:

- 1) 开采之后，即调用 StartGrab 接口之后不允许调用注册接口；
- 2) 允许重复注册，采用覆盖机制。

#### 7.14.4. UnregisterCaptureCallback

**接口定义:** void UnregisterCaptureCallback()

**接口描述:** 注销采集回调函数。

#### 7.14.5. GetImage

**接口定义:** CImageDataPointer GetImage(uint32\_tnTimeout)

**接口描述:** 采集单帧。参数 1 为超时时间，单位毫秒。如果超出此时间还没获取到图像则返回超时错误。

**功能需求:** 如果用户注册了采集委托函数, 则不允许使用 GetImage 接口, 如果强行调用此接口返回非法调用异常。

#### 7.14.6. DQBuf

**接口定义:** CImageDataPointer DQBuf(uint32\_t nTimeout)

**接口描述:** 零拷贝方式采集单帧。参数 1 为超时时间, 单位毫秒。如果超出此时间还没获取到图像则返回超时错误。

**功能需求:** 如果用户注册了采集委托函数, 则不允许使用 DQBuf 接口, 如果强行调用此接口返回非法调用异常。

#### 7.14.7. QBuf

**接口定义:** void QBuf(const CImageDataPointer& plmgData)

**接口描述:** 将 ImageData 交还给采集系统。

**功能需求:** 如果用户注册了采集委托函数, 则不允许使用 QBuf 接口, 如果强行调用此接口返回非法调用异常。

#### 7.14.8. GetFeatureControl

**接口定义:** CGXFeatureControlPointer GetFeatureControl()

**接口描述:** 获取流层属性控制器, 用于控制流层的功能属性。

#### 7.14.9. FlushQueue

**接口定义:** void FlushQueue()

**接口描述:** 清空采集队列中的缓存图像。

#### 7.14.10. SetAcquisitionBufferNumber

**接口定义:** void SetAcquisitionBufferNumber(uint64\_t nBufferNum)

**接口描述:** 用户设置采集 buffer 个数。

**功能要求:**

- 1) 此接口是可选接口, 不是开采流程的必须环节;
- 2) 设置值必须大于 0, 如果设置值特别大, 超过了操作系统容量, 将不能成功开采;
- 3) 如需调用此接口, 则此接口必须在开采之前调用, 不允许在采集过程中调用此接口;
- 4) 用户一旦设置过 buffer 个数, 而且成功采集过, 则此 buffer 个数值将一直有效, 直到关闭流对象为止。

#### 7.14.11. Close

**接口定义:** void Close()

**接口描述:** 关闭流对象。

#### 7.14.12. GetOptimalPacketSize

**接口定义:** uint32\_t GetOptimalPacketSize(void)

**接口描述:** 获取设备最优包长。

**功能要求:**

- 1) 该接口只支持千兆网相机获取最优包长;
- 2) 如需调用此接口, 则此接口必须在开采之前调用, 不允许在采集过程中调用此接口。

#### 7.14.13. GetPayloadSize

**接口定义:** uint64\_t GetPayloadSize(void)

**接口描述:** 获取设备流层数据块大小。

**注意:** 用户获取图像大小不能自行计算, 需要大于等于当前接口的返回值。否则可能导致异常。

### 7.15. IImageData

图像对象, 包含图像数据 buffer 和图像信息。用户注册采集回调函数之后, 当图像到来的时候, 通过采集回调函数的入口参数返回给用户。

**接口列表:**

<a href="#">GetStatus</a>	获取当前帧状态, 是否完整帧
<a href="#">GetPayloadSize</a>	获取当前帧负载, 图像大小
<a href="#">GetWidth</a>	获取当前帧的宽
<a href="#">GetHeight</a>	获取当前帧的高
<a href="#">GetPixelFormat</a>	获取当前帧的图像格式
<a href="#">GetFrameID</a>	获取当前帧 ID
<a href="#">GetTimeStamp</a>	获取当前帧时间戳
<a href="#">GetBuffer</a>	获取图像数据 buffer, 返回 IntPtr 类型指针, 直接指向非托管内存
<a href="#">ConvertToRaw8</a>	此接口专门针对非 8 位数据, 用户可以指定获取非 8 位数据中的指定的 8 个有效位
<a href="#">ConvertToRGB24</a>	从原始数据获取到插值之后的 RGB24 位数据
<a href="#">ImageProcess</a>	对当前图像做图像效果增强, 返回效果增强之后的图像数据

#### 7.15.1. GetStatus

**接口定义:** [GX\\_FRAME\\_STATUS\\_LIST](#) GetStatus()

**功能描述:** 获取当前帧状态, 是否完整帧。

#### 7.15.2. GetPayloadSize

**接口定义:** uint64\_t GetPayloadSize()

**功能描述:** 获取当前帧负载, 图像大小。如果开启帧信息则此值为图像数据大小加帧信息大小, 如果不开启帧信息, 此值就是图像数据大小。

### 7.15.3. GetWidth

**接口定义:** uint64\_t GetWidth()

**功能描述:** 获取当前帧的宽。

### 7.15.4. GetHeight

**接口定义:** uint64\_t GetHeight()

**功能描述:** 获取当前帧的高。

### 7.15.5. GetPixelFormat

**接口定义:** [GX\\_PIXEL\\_FORMAT\\_ENTRY](#) GetPixelFormat()

**功能描述:** 获取当前帧的图像格式。

### 7.15.6. GetFrameID

**接口定义:** uint64\_t GetFrameID()

**功能描述:** 获取当前帧 ID。

### 7.15.7. GetTimeStamp

**接口定义:** uint64\_t GetTimeStamp()

**功能描述:** 获取当前帧时间戳。

### 7.15.8. GetBuffer

**接口定义:** void\* GetBuffer()

**功能描述:** 获取图像数据 buffer，返回 IntPtr 类型指针，直接指向非托管内存。

### 7.15.9. ConvertToRaw8

**接口定义:** void\* ConvertToRaw8([GX\\_VALID\\_BIT\\_LIST](#) emValidBits)

**功能描述:** 此接口专门针对非 8 位数据，用户可以指定获取非 8 位数据中的指定的 8 个有效位。

### 7.15.10. ConvertToRGB24

**接口定义:** void\* ConvertToRGB24([GX\\_VALID\\_BIT\\_LIST](#) emValidBits,  
[GX\\_BAYER\\_CONVERT\\_TYPE\\_LIST](#) emConvertType, Bool bFlip)

**功能描述:** 从原始数据获取到插值之后的 RGB24 位数据。

### 7.15.11. ImageProcess

**接口定义:** void\* ImageProcess(CImageProcessConfigPointer &objImageProcessConfigPtr)

**功能描述:** 对当前图像做图像效果增强，返回效果增强之后的图像数据。

**注意:**

- 1) 如果 IImageProcessConfig::IsAccelerate 返回 true，说明此时将以加速方式处理图像，加速方式处理图像需要图像的高度必须是 4 的整倍数，否则接口报错；
- 2) 黑白相机 void\* 返回的是 8bit 图像数据，图像大小为图像宽 x 图像高；
- 3) 彩色相机 void\* 返回的是 RGB 图像数据，图像大小为图像宽 x 图像高 x 3。

## 7.16. IImageProcessConfig

图像效果增强配置参数对象，内含一组配置参数，通过 IGXDevice::CreateImageProcessConfig 接口获取。

### 接口列表：

<a href="#">SetValidBit</a>	选择获取指定 8 位有效数据位，此接口设置指定哪 8 位
<a href="#">GetValidBit</a>	查询当前指定的哪 8 位有效位
<a href="#">EnableDefectivePixelCorrect</a>	使能坏点校正
<a href="#">IsDefectivePixelCorrect</a>	查询当前是否使能坏点校正
<a href="#">EnableSharpen</a>	使能锐化
<a href="#">IsSharpen</a>	查询当前是否使能锐化
<a href="#">EnableAccelerate</a>	使能加速图像处理
<a href="#">IsAccelerate</a>	查询当前是否工作在加速使能状态
<a href="#">SetSharpenParam</a>	设置锐化强度因子
<a href="#">GetSharpenParam</a>	查询当前使用的锐化强度因子
<a href="#">SetContrastParam</a>	设置对比度调节参数
<a href="#">GetContrastParam</a>	查询当前使用的对比度调节参数
<a href="#">SetGammaParam</a>	设置 Gamma 系数
<a href="#">GetGammaParam</a>	获取 Gamma 系数
<a href="#">SetLightnessParam</a>	设置亮度调节参数
<a href="#">GetLightnessParam</a>	获取亮度调节参数
<a href="#">EnableDenoise</a>	使能降噪
<a href="#">IsDenoise</a>	查询当前是否使能降噪
<a href="#">SetSaturationParam</a>	设置饱和度系数
<a href="#">GetSaturationParam</a>	获取饱和度系数
<a href="#">SetConvertType</a>	设置图像格式转换算法
<a href="#">GetConvertType</a>	获取图像格式转换算法
<a href="#">EnableConvertFlip</a>	使能图像格式转换翻转
<a href="#">IsConvertFlip</a>	查询当前是否使能图像格式转换翻转
<a href="#">EnableColorCorrection</a>	使能颜色校正
<a href="#">IsColorCorrection</a>	查询是否使能颜色校正
<a href="#">Reset</a>	恢复默认调节参数



<a href="#">IsUserSetCCParam</a>	查询颜色校正是否用户设置模式
<a href="#">EnableUserSetCCParam</a>	使能颜色校正用户设置模式
<a href="#">SetUserCCParam</a>	设置颜色转换因子结构体
<a href="#">GetUserCCParam</a>	获取颜色转换因子结构体

#### 7.16.1. SetValidBit

**接口定义：** void SetValidBit([GX\\_VALID\\_BIT\\_LIST](#) emValidBits)

**功能描述：** 选择获取指定 8 位有效数据位，此接口针对非 8 位原始数据设立。

#### 7.16.2. GetValidBit

**接口定义：** [GX\\_VALID\\_BIT\\_LIST](#) GetValidBit()

**功能描述：** 获取指定 8 位有效数据位，此接口针对非 8 位原始数据设立。

#### 7.16.3. EnableDefectivePixelCorrect

**接口定义：** void EnableDefectivePixelCorrect(bool bEnable)

**功能描述：** 使能坏点校正。bEnable 为 true 则使能坏点校正；为 false 则禁用坏点校正。

#### 7.16.4. IsDefectivePixelCorrect

**接口定义：** bool IsDefectivePixelCorrect()

**功能描述：** 获取坏点校正状态。

#### 7.16.5. EnableSharpen

**接口定义：** void EnableSharpen(bool bEnable)

**功能描述：** 使能锐化。bEnable 为 true 则使能锐化；为 false 则禁用锐化。

#### 7.16.6. IsSharpen

**接口定义：** bool IsSharpen()

**功能描述：** 获取锐化状态。

#### 7.16.7. EnableAccelerate

**接口定义：** void EnableAccelerate(bool bEnable)

**功能描述：** 图像处理加速使能。设置 ture 为使能加速，为 false 禁用加速。如果当前 PC 的 CPU 由于本身限制就不支持加速，当用户设置 ture 的时候会报错。

**注意：** 当加速使能的时候，当前图像的高度必须是 4 的整倍数，否则 `IImageData::ImageProcess` 接口会报错。

#### 7.16.8. IsAccelerate

**接口定义：** bool IsAccelerate()

**功能描述：** 查询当前是否工作在加速使能状态，true 表示加速，false 表示不加速。

#### 7.16.9. SetSharpenParam

**接口定义:** void SetSharpenParam(double dParam)

**功能描述:** 设置锐化强度因子。

#### 7.16.10. GetSharpenParam

**接口定义:** double GetSharpenParam()

**功能描述:** 获取锐化强度因子。

#### 7.16.11. SetContrastParam

**接口定义:** void SetContrastParam(int32\_t nParam)

**功能描述:** 设置对比度调节参数。

#### 7.16.12. GetContrastParam

**接口定义:** int32\_t GetContrastParam()

**功能描述:** 获取对比度调节参数。

#### 7.16.13. SetGammaParam

**接口定义:** void SetGammaParam(double dParam)

**功能描述:** 设置 Gamma 系数。

#### 7.16.14. GetGammaParam

**接口定义:** double GetGammaParam()

**功能描述:** 获取 Gamma 系数。

#### 7.16.15. SetLightnessParam

**接口定义:** void SetLightnessParam(int32\_t nParam)

**功能描述:** 设置亮度调节参数。

#### 7.16.16. GetLightnessParam

**接口定义:** int32\_t GetLightnessParam()

**功能描述:** 获取亮度调节参数。

#### 7.16.17. EnableDenoise

**接口定义:** void EnableDenoise(bool bEnable)

**功能描述:** 使能降噪开关（黑白相机不支持）。bEnable 为 true 则使能降噪功能；false 为禁用降噪功能。

#### 7.16.18. IsDenoise

**接口定义:** bool IsDenoise()

**功能描述:** 获取降噪开关（黑白相机不支持）。

#### 7.16.19. SetSaturationParam

**接口定义:** void SetSaturationParam(int32\_t nParam)

**功能描述：**设置饱和度调节系数（黑白相机不支持）。

#### 7.16.20. GetSaturationParam

**接口定义：**int32\_t GetSaturationParam()

**功能描述：**获取饱和度调节系数（黑白相机不支持）。

#### 7.16.21. SetConvertType

**接口定义：**void SetConvertType([GX\\_BAYER\\_CONVERT\\_TYPE\\_LIST](#) emConvertType)

**功能描述：**设置图像格式转换算法（黑白相机不支持）。

#### 7.16.22. GetConvertType

**接口定义：**[GX\\_BAYER\\_CONVERT\\_TYPE\\_LIST](#) GetConvertType()

**功能描述：**获取插值算法（黑白相机不支持）。

#### 7.16.23. EnableConvertFlip

**接口定义：**void EnableConvertFlip(bool bFlip)

**功能描述：**使能插值翻转（黑白相机不支持）。

#### 7.16.24. IsConvertFlip

**接口定义：**bool IsConvertFlip()

**功能描述：**获取插值翻转标识（黑白相机不支持），true 表示翻转，false 表示不翻转。

#### 7.16.25. EnableColorCorrection

**接口定义：**void EnableColorCorrection(bool bEnable)

**功能描述：**使能颜色校正（黑白相机不支持），true 表示使能颜色校正，false 表示禁用颜色校正。

#### 7.16.26. IsColorCorrection

**接口定义：**bool IsColorCorrection()

**功能描述：**查询是否使能颜色校正（黑白相机不支持）。

#### 7.16.27. Reset

**接口定义：**void Reset()

**功能描述：**用户还可以选择恢复最佳默认参数配置。

#### 7.16.28. IsUserSetCCParam

**接口定义：**bool IsUserSetCCParam()

**功能描述：**查询颜色校正是否用户设置模式（黑白相机不支持），true 表示用户模式，false 表示非用户模式。

#### 7.16.29. EnableUserSetCCParam

**接口定义：**void EnableUserSetCCParam(bool bIsUserMode)

**功能描述：**使能颜色校正用户设置模式（黑白相机不支持），true 表示用户模式，false 表示非用户模式。

### 7.16.30. SetUserCCParam

**接口定义:** void SetUserCCParam([COLOR\\_TRANSFORM\\_FACTOR](#) stColorTransformFactor);

**功能描述:** 设置颜色转换因子结构体（颜色校正结构体参数建议范围-4~4，如果参数设置小于-4，校正效果与-4一致，如果参数设置大于4，校正效果与4一致）。

### 7.16.31. GetUserCCParam

**接口定义:** [COLOR\\_TRANSFORM\\_FACTOR](#) GetUserCCParam()

**功能描述:** 获取颜色转换因子结构体。

## 7.17. ICaptureEventHandler

采集回调执行纯虚基类，此类是纯虚基类，不可直接构建对象，用户需要自己继承此类，实现回调函数 DoOnImageCaptured 内部动作。

**接口列表:**

[DoOnImageCaptured](#) 回调响应函数，当接收完一帧的时候回调此函数

### 7.17.1. DoOnImageCaptured

**接口定义:** void DoOnImageCaptured(CImageDataPointer& objImageDataPointer, void\* pUserParam)

**功能描述:** 回调响应函数，当接收完一帧的时候回调此函数。

## 7.18. IDeviceOfflineEventHandler

掉线事件回调执行纯虚基类，此类是纯虚基类，不可直接构建对象，用户需要自己继承此类，实现回调函数 DoOnDeviceOfflineEvent 内部动作。

**接口列表:**

[DoOnDeviceOfflineEvent](#) 回调响应函数，当设备掉线的时候回调此函数

### 7.18.1. DoOnDeviceOfflineEvent

**接口定义:** void DoOnDeviceOfflineEvent(void\* pUserParam)

**功能描述:** 回调响应函数，当设备掉线的时候回调此函数。

## 7.19. IFeatureEventHandler

设备事件回调执行纯虚基类，此类是纯虚基类，不可直接构建对象，用户需要自己继承此类，实现回调函数 DoOnFeatureEvent 内部动作。

**接口列表:**

[DoOnFeatureEvent](#) 回调响应函数，当设备事件到达的时候回调此函数

### 7.19.1. DoOnFeatureEvent

**接口定义:** void DoOnFeatureEvent(const GxIAPICPP::gxstring& strFeatureName, void\* pUserParam)

## 7.20. IGXImageFormatConvert

图像格式转换接口类，可通过此对象转换图像格式。

**接口列表：**

<a href="#">SetDstFormat</a>	设置期望的转换格式
<a href="#">GetDstFormat</a>	获取期望转换格式
<a href="#">SetInterpolationType</a>	设置图像格式转换算法
<a href="#">GetInterpolationType</a>	获取图像格式转换算法
<a href="#">SetAlphaValue</a>	设置带有 Alpha 通道图像的 Alpha 值
<a href="#">GetAlphaValue</a>	获取 Alpha 通道值
<a href="#">GetBufferSizeForConversion</a>	方法一，根据输入图像指针获取期望格式的图像 Buffer 长度
<a href="#">GetBufferSizeForConversion</a>	方法二，根据输入图像数据获取期望格式的图像 Buffer 长度
<a href="#">Convert</a>	方法一，根据输入图像指针获取转换后图像 Buffer
<a href="#">Convert</a>	方法二，根据输入图像数据获取转换后图像 Buffer

### 7.20.1. 支持转换类型

表格中为当前支持的转换图像格式，源图像格式与目的图像格式均存在一对多关系（即源图像格式中任一格式可转换为对应的目的图像格式的任意一种）。

	源图像格式	目的图像格式
Bayer 8bit	GX_PIXEL_FORMAT_BAYER_GR8 GX_PIXEL_FORMAT_BAYER_RG8 GX_PIXEL_FORMAT_BAYER_GB8 GX_PIXEL_FORMAT_BAYER_BG8	GX_PIXEL_FORMAT_MONO8 GX_PIXEL_FORMAT_RGB8 GX_PIXEL_FORMAT_BGR8 GX_PIXEL_FORMAT_RGBA8 GX_PIXEL_FORMAT_BGRA8 GX_PIXEL_FORMAT_ARGB8 GX_PIXEL_FORMAT_ABGR8 GX_PIXEL_FORMAT_RGB8_PLANAR
Bayer 16bit	GX_PIXEL_FORMAT_BAYER_GR10 GX_PIXEL_FORMAT_BAYER_RG10 GX_PIXEL_FORMAT_BAYER_GB10 GX_PIXEL_FORMAT_BAYER_BG10 GX_PIXEL_FORMAT_BAYER_GR12 GX_PIXEL_FORMAT_BAYER_RG12 GX_PIXEL_FORMAT_BAYER_GB12 GX_PIXEL_FORMAT_BAYER_BG12 GX_PIXEL_FORMAT_BAYER_GR16 GX_PIXEL_FORMAT_BAYER_RG16 GX_PIXEL_FORMAT_BAYER_GB16 GX_PIXEL_FORMAT_BAYER_BG16	GX_PIXEL_FORMAT_MONO16 GX_PIXEL_FORMAT_RGB16 GX_PIXEL_FORMAT_BGR16 GX_PIXEL_FORMAT_RGB8 GX_PIXEL_FORMAT_BGR8 GX_PIXEL_FORMAT_RGB16_PLANAR
RGB 8bit	GX_PIXEL_FORMAT_RGB8 GX_PIXEL_FORMAT_BGR8	GX_PIXEL_FORMAT_YUV444_8 GX_PIXEL_FORMAT_YUV422_8 GX_PIXEL_FORMAT_YUV411_8 GX_PIXEL_FORMAT_YUV420_8_PLANAR GX_PIXEL_FORMAT_YCBCR444_8 GX_PIXEL_FORMAT_YCBCR422_8 GX_PIXEL_FORMAT_YCBCR411_8 GX_PIXEL_FORMAT_MONO8

		GX_PIXEL_FORMAT_RGB8
RGB 16bit	GX_PIXEL_FORMAT_RGB16 GX_PIXEL_FORMAT_BGR16	GX_PIXEL_FORMAT_MONO16
Mono Packed	GX_PIXEL_FORMAT_MONO10_PACKED GX_PIXEL_FORMAT_MONO12_PACKED GX_PIXEL_FORMAT_MONO10_P GX_PIXEL_FORMAT_MONO12_P GX_PIXEL_FORMAT_MONO14_P	GX_PIXEL_FORMAT_MONO8 GX_PIXEL_FORMAT_MONO10 GX_PIXEL_FORMAT_MONO12 GX_PIXEL_FORMAT_MONO14 GX_PIXEL_FORMAT_MONO16 GX_PIXEL_FORMAT_RGB8 GX_PIXEL_FORMAT_BGR8
Bayer Packed	GX_PIXEL_FORMAT_BAYER_GR10_PACKED GX_PIXEL_FORMAT_BAYER_RG10_PACKED GX_PIXEL_FORMAT_BAYER_GB10_PACKED GX_PIXEL_FORMAT_BAYER_BG10_PACKED GX_PIXEL_FORMAT_BAYER_GR12_PACKED GX_PIXEL_FORMAT_BAYER_RG12_PACKED GX_PIXEL_FORMAT_BAYER_GB12_PACKED GX_PIXEL_FORMAT_BAYER_BG12_PACKED GX_PIXEL_FORMAT_BAYER_GR10_P GX_PIXEL_FORMAT_BAYER_RG10_P GX_PIXEL_FORMAT_BAYER_GB10_P GX_PIXEL_FORMAT_BAYER_BG10_P GX_PIXEL_FORMAT_BAYER_GR12_P GX_PIXEL_FORMAT_BAYER_RG12_P GX_PIXEL_FORMAT_BAYER_GB12_P GX_PIXEL_FORMAT_BAYER_BG12_P GX_PIXEL_FORMAT_BAYER_GR14_P GX_PIXEL_FORMAT_BAYER_RG14_P GX_PIXEL_FORMAT_BAYER_GB14_P GX_PIXEL_FORMAT_BAYER_BG14_P	GX_PIXEL_FORMAT_RGB8 GX_PIXEL_FORMAT_BGR8 GX_PIXEL_FORMAT_BAYER_GR8 GX_PIXEL_FORMAT_BAYER_RG8 GX_PIXEL_FORMAT_BAYER_GB8 GX_PIXEL_FORMAT_BAYER_BG8 GX_PIXEL_FORMAT_BAYER_GR10 GX_PIXEL_FORMAT_BAYER_RG10 GX_PIXEL_FORMAT_BAYER_GB10 GX_PIXEL_FORMAT_BAYER_BG10 GX_PIXEL_FORMAT_BAYER_GR12 GX_PIXEL_FORMAT_BAYER_RG12 GX_PIXEL_FORMAT_BAYER_GB12 GX_PIXEL_FORMAT_BAYER_BG12 GX_PIXEL_FORMAT_BAYER_GR14 GX_PIXEL_FORMAT_BAYER_RG14 GX_PIXEL_FORMAT_BAYER_GB14 GX_PIXEL_FORMAT_BAYER_BG14 GX_PIXEL_FORMAT_BAYER_GR16 GX_PIXEL_FORMAT_BAYER_RG16 GX_PIXEL_FORMAT_BAYER_GB16 GX_PIXEL_FORMAT_BAYER_BG16
COORD D3D_C 16	GX_PIXEL_FORMAT_COORD3D_C16	GX_PIXEL_FORMAT_COORD3D_ABC32F GX_PIXEL_FORMAT_COORD3D_ABC32F _PLANAR

#### 7.20.2. SetDstFormat

**接口定义：** void SetDstFormat([GX\\_PIXEL\\_FORMAT\\_ENTRY](#) emDstFormat)

**功能描述：** 设置期望转换格式。

**参数：** emDstFormat: [in]      期望的转换格式，详见 [GX\\_PIXEL\\_FORMAT\\_ENTRY](#) 及 [支持转换类型](#)

#### 7.20.3. GetDstFormat

**接口定义：** [GX\\_PIXEL\\_FORMAT\\_ENTRY](#) GetDstFormat()

**功能描述：** 获取期望转换格式。

#### 7.20.4. SetInterpolationType

**接口定义：** void SetInterpolationType ([GX\\_BAYER\\_CONVERT\\_TYPE\\_LIST](#) emConvertType)

**功能描述：** 设置图像格式转换算法。

**参数：** emConvertType: [in]      转换算法，详见 [GX\\_BAYER\\_CONVERT\\_TYPE\\_LIST](#)

默认值为: [GX\\_RAW2RGB\\_NEIGHBOUR](#)

### 7.20.5. GetInterpolationType

**接口定义:** [GX\\_BAYER\\_CONVERT\\_TYPE\\_LIST](#) GetInterpolationType ()

**功能描述:** 获取图像格式转换算法。

### 7.20.6. SetAlphaValue

**接口定义:** void SetAlphaValue(uint64\_t ui64AlphaValue)

**功能描述:** 设置带有 Alpha 通道图像的 Alpha 值。

**参数:** ui64AlphaValue: [in]          Alpha 通道值, 取值范围 0~255, 默认值为: 255。

### 7.20.7. GetAlphaValue

**接口定义:** uint64\_t GetAlphaValue()

**功能描述:** 获取 Alpha 通道值

### 7.20.8. GetBufferSizeForConversion

**接口定义:** uint64\_t GetBufferSizeForConversion(CImageDataPointer pObjSrcImageData)

**功能描述:** 根据输入图像指针获取期望格式的图像 Buffer 长度。

**参数:** pObjSrcImageData: [in]      源图像指针, 可通过回调函数 [DoOnImageCaptured\(\)](#)、[DQBuf\(\)](#)、[GetImage\(\)](#)获得

### 7.20.9. GetBufferSizeForConversion

**接口定义:** uint64\_t GetBufferSizeForConversion(uint64\_t ui64SrcWidth,  
uint64\_t ui64SrcHeight,  
[GX\\_PIXEL\\_FORMAT\\_ENTRY](#) emSrcFormat)

**功能描述:** 根据输入图像数据获取期望格式的图像 Buffer 长度。

**参数:**

ui64SrcWidth:	[in]	源图像宽
ui64SrcHeight:	[in]	源图像高
emSrcFormat:	[in]	源图像格式

### 7.20.10. Convert

**接口定义:** void Convert(CImageDataPointer pObjSrcImageData,  
void\* pDstBuffer,  
size\_t nDstBufferSize,  
bool bFlip)

**功能描述:** 将输入源图像转换为期望的图像格式, 可转换类型参考[支持转换类型](#)。

**参数:**

pObjSrcImageData:	[in]	源图像指针可通过回调函数 <a href="#">DoOnImageCaptured()</a> 或 <a href="#">GetImage()</a> 获得
pDstBuffer:	[in]	目的图像 Buffer 指针, 该内存由用户申请, 长度为 nDstBufferSize
	[out]	目的图像 Buffer



nDstBufferSize: [in] 目的图像 Buffer 长度, 可通过 [GetBufferSizeForConversion\(\)](#) 获得  
bFlip: [in] 是否翻转 (true-翻转, false-不翻转)

### 7.20.11. Convert

**接口定义:** void Convert(void\* pSrcBuffer, uint64\_t ui64SrcWidth, uint64\_t ui64SrcHeight, [GX\\_PIXEL\\_FORMAT\\_ENTRY](#) emSrcFormat, void\* pDstBuffer, size\_t nDstBufferSize, bool bFlip)

**功能描述:** 将输入源图像转换为期望的图像格式, 可转换类型参考[支持转换类型](#)。

**参数:**

pSrcBuffer: [in] 源图像 Buffer 指针  
ui64SrcWidth: [in] 源图像宽度  
ui64SrcHeight: [in] 源图像高度  
emSrcFormat: [in] 源图像格式  
pDstBuffer: [in] 目的图像 Buffer 指针, 该内存由用户申请, 长度为 nDstBufferSize  
[out] 目的图像 Buffer 数据  
nDstBufferSize: [in] 目的图像 Buffer 长度, 可通过 [GetBufferSizeForConversion\(\)](#) 获得  
bFlip: [in] 图像是否翻转 (true-翻转, false-不翻转)

### 7.20.12. SetValidBits

**接口定义:** void SetValidBits([GX\\_VALID\\_BIT\\_LIST](#) emValidBits)

**功能描述:** 设置输入源图像转换为期望的图像格式时保留的有效数据位。

参数 emValidBits: [in] 有效数据位, 详见 [GX\\_VALID\\_BIT\\_LIST](#)

### 7.20.13. GetValidBits

**接口定义:** [GX\\_VALID\\_BIT\\_LIST](#) GetValidBits()

**功能描述:** 获取输入源图像转换为期望的图像格式时保留的有效数据位。

### 7.20.14. Set3DCalibParam

**接口定义:** void Set3DCalibParam(void\* pCalibParamBuffer, size\_t nBufferSize)

**功能描述:** 该函数用于加载 3D 标定参数, 将高度图转换为点云图。

**参数:**

pCalibParamBuffer: [in] 3D 标定参数内存  
nBufferSize: [in] 3D 标定参数内存长度

### 7.20.15. Get3DCalibParam

**接口定义:** void\* Get3DCalibParam()

**功能描述:** 获取 3D 标定参数。



7.20.16. SetYStep

接口定义：void SetYStep(double dY)

功能描述：设置高度图转点云的 Y 方向步长。

参数 dY: [in]     Y 方向步长

7.20.17. GetYStep

接口定义：double GetYStep()

功能描述：获取高度转点云的 Y 方向步长。

7.21. IGXImageProcess

图像处理功能类，可通过此对象提升图像质量，包括坏点校正、降噪、锐化、颜色校正、亮度对比度 Gamma 调节等功能。

接口列表：

<a href="#">StaticDefectCorrection</a>	静态坏点校正
<a href="#">ImageImprovment</a>	方法一，输入图像数据进行图像质量提升
<a href="#">ImageImprovment</a>	方法二，输入图像指针进行图像质量提升
<a href="#">CalculaLut</a>	相机查找表计算
<a href="#">ReadLutFile</a>	查找表文件读取

7.21.1. StaticDefectCorrection

接口定义：void StaticDefectCorrection(void \*pInputBuffer, void \*pOutputBuffer, const [GX\\_STATIC\\_DEFECT\\_CORRECTION](#) stDefectCorrection, void\* pDefectPosBuffer, int32\_t nDefectPosBufferSize)

功能描述：执行静态坏点校正。

参数：

pInputBuffer:	[in]	源图像 Buffer 指针
pOutputBuffer:	[in]	目的图像 Buffer 指针
	[out]	目的图像 Buffer 数据
stDefectCorrection:	[in]	源图像参数，详见 <a href="#">GX_STATIC_DEFECT_CORRECTION</a>
pDefectPosBuffer:	[in]	坏点位置数据（从坏点文件中读取数据，坏点文件通过静态坏点校正插件保存的 dp 文件）
nDefectPosBufferSize:	[in]	坏点位置数据 Buffer 大小

7.21.2. ImageImprovment

接口定义：void ImageImprovment(GX\_IMAGE\_INFO& stImgInfo, void \*pOutputBuf, CImageProcessConfigPointer& objCfg)

**功能描述：**提升图像质量。支持输入格式：RGB/BGR、Bayer、Mono。

**参数：**

stlmgInfo:	[in]	源图像信息，详见 <a href="#">GX_IMAGE_INFO</a>
pOutputBuf:	[in]	目的图像 Buffer 指针
	[out]	目的图像 Buffer 数据
objCfg:	[in]	图像处理功能配置

支持的功能		源图像格式		
		RGB/BGR	Bayer	Mono
DefectivePixelCorrect	坏点校正	×	√	√
Sharpen	锐化	×	√	√
Contrast	对比度	√	√	√
Gamma	伽马	√	√	√
Lightness	亮度	×	√	√
Denoise	降噪	×	√	×
ColorCorrection	颜色校正	√	√	×
Saturation	饱和度	×	√	×
ConvertType	Bayer 转换类型	×	√	×
Flip	翻转	√	√	√
Accelerate	加速	×	√	√

### 7.21.3. ImageImprovment

**接口定义：**void ImageImprovment (CImageDataPointer objFrameData,  
void \*pOutputBuf,  
CImageProcessConfigPointer& objCfg)

**功能描述：**提升图像质量。支持输入格式：RGB/BGR、Bayer、Mono。

**参数：**

objFrameData:	[in]	源图像指针，可通过回调函数 <a href="#">DoOnImageCaptured()</a> 、 <a href="#">DQBuf()</a> 、 <a href="#">GetImage()</a> 获得
pOutputBuf:	[in]	目的图像 Buffer 指针
	[out]	目的图像 Buffer 数据
objCfg:	[in]	图像处理功能配置，支持的功能同 7.21.2 节

### 7.21.4. CalculaLut

**接口定义：**void CalculaLut(int32\_t nContrastParam, double dGamma,  
int32\_t nLightness, void\* pLut,  
uint32\_t\* pnLutLength)

**功能描述：**相机查找表计算。

参数:

nContrastParam:	[in]	对比度, 取值范围: -50 ~ 100, 可通过 <a href="#">GetContrastParam()</a> 获得
dGamma:	[in]	Gamma, 取值范围: 0.1 ~ 10, 可通过 <a href="#">GetGammaParam()</a> 获得
nLightness:	[in]	亮度, 取值范围: -150 ~ 150, 可通过 <a href="#">GetLightnessParam()</a> 获得
pLut:	[in]	查找表内存指针, 该内存由用户申请, 长度为 pnLutLength
	[out]	通过计算得到的相机查找表, 可通过 <a href="#">GetRegisterFeature</a> ( "LUTValueAll" ) ->SetBuffer(pLut, pnLutLength)设置相机查找表
pnLutLength:	[in]	查找表长度可通过 <a href="#">GetRegisterFeature</a> ( "LUTValueAll" )-> <a href="#">GetLength()</a> 获得

7.21.5. ReadLutFile

**接口定义:** void ReadLutFile(const GxIAPICPP::gxstring& strLutFilePath,  
void\* pLut,  
uint32\_t\* pnLutLength)

**功能描述:** 读取查找表插件保存的.lut 文件, 并根据相机查找表长度, 将数据解析为可以设置到相应相机内的查找表数据。

参数:

strLutFilePath:	[in]	查找表文件路径, 查找表文件(xxx.lut)可以通过查找表插件获得。 插件路径: "Galaxy View->菜单栏->插件->查找表生成工具插件"
pLut:	[in]	查找表内存指针, 该内存由用户申请, 长度为 pnLutLength
	[out]	通过读取文件得到得相机查找表, 可通过 <a href="#">GetRegisterFeature</a> ( "LUTValueAll" )->SetBuffer(pLut, pnLutLength)设置相机查找表
pnLutLength:	[in]	查找表长度, 单位为字节, 可通过 <a href="#">GetRegisterFeature</a> ( "LUTValueAll" )-> <a href="#">GetLength()</a> 获得

7.22. IGXImageSpecialFunction (不再维护)

图像特殊接口类。

接口列表:

<a href="#">CalculaLut</a>	相机查找表计算
<a href="#">ReadLutFile</a>	查找表文件读取

7.22.1. CalculaLut

**接口定义:** void CalculaLut(int32\_t nContrastParam, double dGamma, int32\_t nLightness,  
void\* pLut, uint32\_t\* pnLutLength)

**功能描述：**相机查找表计算。

**参数：**

nContrastParam:	[in]	对比度，取值范围：-50 ~ 100，可通过 <a href="#">GetContrastParam()</a> 获得
dGamma:	[in]	Gamma，取值范围：0.1 ~ 10，可通过 <a href="#">GetGammaParam()</a> 获得
nLightness:	[in]	亮度，取值范围：-150 ~ 150，可通过 <a href="#">GetLightnessParam()</a> 获得
pLut:	[in]	查找表内存指针，该内存由用户申请，长度为 pnLutLength
	[out]	通过计算得到的相机查找表 可通过 <a href="#">GetRegisterFeature</a> ("LUTValueAll")->SetBuffer(pLut, pnLutLength)设置相机查找表
pnLutLength:	[in]	查找表长度，可通过 <a href="#">GetRegisterFeature</a> ("LUTValueAll")-> <a href="#">GetLength()</a> 获得

### 7.22.2. ReadLutFile

**接口定义：** void ReadLutFile(const GxIAPICPP::gxstring& strLutFilePath,  
void\* pLut, uint32\_t\* pnLutLength)

**功能描述：**读取查找表插件保存的.lut 文件，并根据相机查找表长度，将数据解析为可以设置到相应相机内的查找表数据。

**参数：**

strLutFilePath:	[in]	查找表文件路径，查找表文件（xxx.lut）可以通过查找表插件获得 插件路径：“Galaxy View->菜单栏->插件->查找表生成工具插件”
pLut:	[in]	查找表内存指针，该内存由用户申请，长度为 pnLutLength
	[out]	通过读取文件得到得相机查找表 可通过 <a href="#">GetRegisterFeature</a> ("LUTValueAll")->SetBuffer(pLut, pnLutLength)设置相机查找表
pnLutLength:	[in]	查找表长度，单位为字节 可通过 <a href="#">GetRegisterFeature</a> ("LUTValueAll")-> <a href="#">GetLength()</a> 获得

## 8. 示例工程

下面给出了一个完整的控制台程序示例代码，可以注册设备掉线事件、注册远端设备事件、注册采集回调事件。

```
#include "stdafx.h"
#include <iostream>
using namespace std;

//请用户提前配置好工程头文件目录, 需要包含 GalaxyIncludes.h
#include "GalaxyIncludes.h"

//用户继承掉线事件处理类
class CSampleDeviceOfflineEventHandler : public IDeviceOfflineEventHandler
{
public:
    void DoOnDeviceOfflineEvent(void* pUserParam)
    {
        cout<<"收到设备掉线事件!"<<endl;
    }
};

//用户继承属性更新事件处理类
class CSampleFeatureEventHandler : public IFeatureEventHandler
{
public:
    void DoOnFeatureEvent(const GxIAPICPP::gxstring&strFeatureName, void* pUserParam)
    {
        cout<<"收到曝光结束事件!"<<endl;
    }
};

//用户继承采集事件处理类
class CSampleCaptureEventHandler : public ICaptureEventHandler
{
public:
    void DoOnImageCaptured(CImageDataPointer&objImageDataPointer, void* pUserParam)
    {
        cout<<"收到一帧图像!"<<endl;
        cout<<"ImageInfo: "<<objImageDataPointer->GetStatus() <<endl;
        cout<<"ImageInfo: "<<objImageDataPointer->GetWidth() <<endl;
        cout<<"ImageInfo: "<<objImageDataPointer->GetHeight() <<endl;
        cout<<"ImageInfo: "<<objImageDataPointer->GetPayloadSize() <<endl;
    }
};

int _tmain(int argc, _TCHAR* argv[])
{

```

```

//声明事件回调对象指针
IDeviceOfflineEventHandler* pDeviceOfflineEventHandler = NULL; // <掉线事件回调对象
IFeatureEventHandler* pFeatureEventHandler = NULL; // <远端设备事件回调对象
ICaptureEventHandler* pCaptureEventHandler = NULL; // <采集回调对象

//初始化
IGXFactory::GetInstance().Init();

try
{
    do
    {
        //枚举设备
        gxdeviceinfo_vector vectorDeviceInfo;
        IGXFactory::GetInstance().UpdateDeviceList(1000, vectorDeviceInfo);
        if (0 == vectorDeviceInfo.size())
        {
            cout<<"无可设备!"<<endl;
            break;
        }

        //打开第一台设备以及设备下面第一个流
        CGXDevicePointer ObjDevicePtr = IGXFactory::GetInstance().OpenDeviceBySN(
            vectorDeviceInfo[0].GetSN(),
            GX_ACCESS_EXCLUSIVE);
        CGXStreamPointer ObjStreamPtr = ObjDevicePtr->OpenStream(0);

        //获取远端设备属性控制器
        CGXFeatureControlPointer ObjFeatureControlPtr =
            ObjDevicePtr->GetRemoteFeatureControl();

        //获取流层属性控制器
        CGXFeatureControlPointer objStreamFeatureControlPtr =
            ObjStreamPtr->GetFeatureControl();

        //提高网络相机的采集性能, 设置方法参考以下代码 (目前只有千兆网系列相机支持设置最优包长)。
        GX_DEVICE_CLASS_LIST objDeviceClass =
            ObjDevicePtr->GetDeviceInfo().GetDeviceClass();
        if (GX_DEVICE_CLASS_GEV == objDeviceClass)
        {
            //判断设备是否支持流通道数据包功能
            if (true == ObjFeatureControlPtr->IsImplemented("GevSCPSPacketSize"))
            {
                //获取当前网络环境的最优包长值
                int nPacketSize = ObjStreamPtr->GetOptimalPacketSize();
            }
        }
    } while (true);
}

```

```

//将最优包长值设置为当前设备的流通道包长值
ObjFeatureControlPtr->GetIntFeature(
    "GevSCPSPacketSize")->SetValue(nPacketSize);
}

//设置 Buffer 处理模式
objStreamFeatureControlPtr->GetEnumFeature(
    "StreamBufferHandlingMode")->SetValue("OldestFirst");

//注册设备掉线事件（目前只有千兆网系列相机支持此事件通知）
GX_DEVICE_OFFLINE_CALLBACK_HANDLE hDeviceOffline = NULL;
pDeviceOfflineEventHandler =
    new CSampleDeviceOfflineEventHandler();
hDeviceOffline = ObjDevicePtr->RegisterDeviceOfflineCallback(
    pDeviceOfflineEventHandler, NULL);

//设置曝光时间(示例中写死 us,只是示例,并不代表真正可工作参数)
//ObjFeatureControlPtr->GetFloatFeature("ExposureTime")->SetValue(50);

//注册远端设备事件:曝光结束事件(目前只有千兆网系列相机支持曝光结束事件)
//选择事件源
ObjFeatureControlPtr->GetEnumFeature(
    "EventSelector")->SetValue("ExposureEnd");

//使能事件
ObjFeatureControlPtr->GetEnumFeature(
    "EventNotification")->SetValue("On");
GX_FEATURE_CALLBACK_HANDLE hFeatureEvent = NULL;
pFeatureEventHandler = new CSampleFeatureEventHandler();
hFeatureEvent = ObjFeatureControlPtr->RegisterFeatureCallback(
    "EventExposureEnd", pFeatureEventHandler, NULL);

//注册回调采集
pCaptureEventHandler = new CSampleCaptureEventHandler();
ObjStreamPtr->RegisterCaptureCallback(pCaptureEventHandler,
    NULL);

//发送开采命令
ObjStreamPtr->StartGrab();
ObjFeatureControlPtr->GetCommandFeature(
    "AcquisitionStart")->Execute();

//此时开采成功,控制台打印信息,直到输入任意键继续
getchar();

//发送停采命令
ObjFeatureControlPtr->GetCommandFeature(

```

```

"AcquisitionStop")->Execute();

ObjStreamPtr->StopGrab();

//注销采集回调
ObjStreamPtr->UnregisterCaptureCallback();

//注销远端设备事件
ObjFeatureControlPtr->UnregisterFeatureCallback(hFeatureEvent);

//注销设备掉线事件
ObjDevicePtr->UnregisterDeviceOfflineCallback(hDeviceOffline);

//释放资源
ObjStreamPtr->Close();
ObjDevicePtr->Close();
} while (0);
}
catch (CGalaxyException&e)
{
    cout<<"错误码: "<<e.GetErrorCode() <<endl;
    cout<<"错误描述信息: "<<e.what() <<endl;
}
catch (std::exception&e)
{
    cout<<"错误描述信息: "<<e.what() <<endl;
}

//反初始化库
IGXFactory::GetInstance().Uninit();

//销毁事件回调指针
if (NULL != pCaptureEventHandler)
{
    delete pCaptureEventHandler;
    pCaptureEventHandler = NULL;
}
if (NULL != pDeviceOfflineEventHandler)
{
    delete pDeviceOfflineEventHandler;
    pDeviceOfflineEventHandler = NULL;
}
if (NULL != pFeatureEventHandler)
{
    delete pFeatureEventHandler;
    pFeatureEventHandler = NULL;
}
return 0;
}

```



## 9. 版本说明

序号	修订版本号	所做改动	发布日期
1	V1.0.1	1. 初始发布	2015-11
2	V1.0.4	1. 对 GetImage 采单帧接口添加大帧率使用限制说明 2. 添加说明远端设备事件只有千兆网支持 3. 添加回调事件虚基类说明和链接 4. 添加独立的一章节展示展示完整的调用过程	2016-02-01
3	V1.0.6	1. 修改对 GetImage 采单帧接口添加大帧率使用限制说明	2016-02-04
4	V1.0.7	1. 修改 C++系统测试第一轮 BUG 中 674、675、676BUG	2016-03-21
5	V1.0.9	1. 添加 GigEIPConfiguration 和 GigEForceIp 两个接口的功能描述	2016-07-20
6	V1.0.11	1. 增加采集 buffer 可设接口说明	2016-10-24
7	V1.0.12	1. 增加相机内部错误事件的说明：内部错误事件 ID 和时间戳	2016-12-14
8	V1.1.1	1. 修改 2.11.5 章节 Gamma 默认值为 1.0，对比度默认值为 0，饱和度和默认值为 64	2019-07-01
9	V1.1.2	1. 新增设备复位重连接口 2. ImageProcess 接口新增 void*描述信息	2019-07-16
10	V1.1.3	1. 新增获取设备最优包长接口 2. 在所有涉及打开流，开始采集的代码中增加设置设备最优包长的代码	2019-08-15
11	V1.1.4	1. 新增数据类型：COLOR_TRANSFORM_FACTOR 2. ImageProcessConfig 中新增 4 个用户设置颜色校正因子相关接口	2019-09-29
12	V1.2.0	1. 添加 StreamBufferHandlingMode 相关参数信息	2019-12-25
13	V1.2.1	1. 增加 2.10.5 节，通过演示程序获取属性读写示例代码功能说明	2020-08-28
14	V1.2.2	1. 添加用户修改心跳超时时间方式	2020-10-12
15	V1.2.3	1. 新增 IGXImageFormatConvert 接口类 2. 新增 IGXImageSpecialFunction 接口类 3. 添加 GX_PIXEL_FORMAT_ENTRY 枚举类型	2022-01-12
16	V1.2.4	1. 新增 ReadLutFile 接口	2022-03-21
17	V1.2.5	1. 新增读取、设置寄存器接口 2. 修改 7.20.1 节，新增支持转换类型	2022-09-21

序号	修订版本号	所做改动	发布日期
18	V1.3.0	<ol style="list-style-type: none"> <li>1. 增加 IGXInterface 接口类</li> <li>2. 增加 IGXInterfaceInfo 接口类</li> <li>3. 新增 2.6、2.7 节，Interface 使用方式说明</li> <li>4. 增加 GX_TL_TYPE_LIST 枚举类型</li> <li>5. 修改 2.5 节，更新新增枚举接口</li> <li>6. 修改 2.10.1 小节，新增 Interface 获取属性控制器的方式</li> <li>7. 增加 IGXFactory::GetAllInterfaceInfo, IGXFactory::GetInterface, IGXFactory::UpdateAllDeviceListEx 接口</li> <li>8. 增加 IGXDevice::GetParentInterface 接口</li> <li>9. 增加 IGXStream::GetPayloadSize 接口</li> <li>10. 增加 IGXFeatureControl::FeatureSave、IGXFeayureControl::FeatureSaveW、IGXFeatureControl::FeatureLoad、IGXFeatureControl::FeatureLoadW 接口</li> </ol>	2024-03-19
19	V1.3.1	<ol style="list-style-type: none"> <li>1. 增加 GX_PIXEL_COLOR_FILTER、GX_ACTUAL_BITS 枚举类型，补充 GX_VALID_BIT_LIST 枚举类型</li> <li>2. 增加 GX_STATIC_DEFECT_CORRECTION 、GX_IMAGE_INFO 数据结构</li> <li>3. 新增 7.1.15 节，新增创建图像处理对象说明</li> <li>4. 修改 7.20.1 节，补充和修正支持的格式转换类型</li> <li>5. 新增 7.20.12 节 IGXImageFormatConvert::SetValidBits 接口</li> <li>6. 新增 7.20.13 节 IGXImageFormatConvert::GetValidBits 接口</li> <li>7. 新增 7.21 节，添加 IGXImageProcess 接口类</li> </ol>	2024-03-22
20	V1.3.2	<ol style="list-style-type: none"> <li>1. 扩展 GX_PIXEL_FORMAT_ENTRY 枚举类型</li> <li>2. 新增 7.20.14 节 IGXImageFormatConvert::Set3DCalibParam 接口</li> <li>3. 新增 7.20.15 节 IGXImageFormatConvert::Get3DCalibParam 接口</li> <li>4. 新增 7.20.16 节 IGXImageFormatConvert::SetYStep 接口</li> <li>5. 新增 7.20.17 节 IGXImageFormatConvert::GetYStep 接口</li> </ol>	2024-04-03
21	V1.3.3	<ol style="list-style-type: none"> <li>1. 增加 GX_LOG_TYPE_LIST 枚举类型</li> <li>2. 新增 7.1.16 节 IGXFactory::SetLogType 接口</li> <li>3. 新增 7.1.17 节 IGXFactory::GetLogType 接口</li> </ol>	2024-04-23

序号	修订版本号	所做改动	发布日期
22	V1.3.4	<ol style="list-style-type: none"> <li>1. 修改 2.11.1、2.11.2 节</li> <li>2. 新增 7.14.6 节 IGXStream::DQBuf 接口</li> <li>3. 新增 7.14.7 节 IGXStream::QBuf 接口</li> </ol>	2024-06-19