浙江大华技术股份有限公司
ZHEJIANG DAHUA TECHNOLOGY CO.,LTD.

工业相机 SDK 开发手册

（C#篇）

Ver1.0

变更履历

编号	变更日期	版本号	变更内容概要	配套 SDK 版本
1	2019/02/28	Ver1.0	初版做成	

目录

1. SDK 概要说明

1.1. 下载 SDK 安装包

大华工业相机 SDK 可以通过华睿官网下载。

下载链接 ： http://download.huaraytech.com/pub/sdk/

该页面罗列了各个版本的 SDK。根据需要，下载相应的版本。

1.2. SDK 支持的操作系统

SDK 支持以下版本的操作系统

操作系统类型	操作系统版本号	备注
Windows	WinXP SP3	注意需要 SP3 补丁
	Win7 SP1 32/64bit	注意需要 SP1 补丁
	Win8 32/64bit	
	Win10 32/64bit	
Linux (x86)	Ubuntu 12.04 32/64bit	支持的内核版本范围参考*1

	Ubuntu 14.04 32/64bit	支持的内核版本范围参考*1
	Ubuntu 16.04 32/64bit	支持的内核版本范围参考*1
	Ubuntu 17.04 32/64bit	支持的内核版本范围参考*1
	Ubuntu 18.04 32/64bit	支持的内核版本范围参考*1
	CentOS 6.5~7.5 32/64bit	支持的内核版本范围参考*1

*1. 目前支持的 Linux 内核版本号从 2.6.32（含）到 4.18.0（含）。

可以通过 uname –a 命令来确定系统内核版本号。

*2. Linux ARM 版本的 SDK 需要定制化开发。

但针对 NVIDIA Jetson TK1、TX1、TX2 平台，已和 x86 平台一样标准发布 SDK。

1.3. SDK 安装目录

1.3.1. Windows 版 SDK

Windows 版本 SDK 的默认安装路径（*1）为“c:\Program Files\DahuaTech\MV Viewer”，安装目录结构如下：

*1. 从 Ver2.1.0 版本开始，支持用户自己选择安装路径。

文件夹	描述
Application	1. 相机客户端软件（MV Viewer），包含 32 位和 64 位 2. 相机配套工具软件（CamTools），包含 32 位和 64 位
Development	存放了 SDK 二次开发所需的各种资源文件 1. SDK 头文件、Lib 库

	2. dotNet SDK 库 3. 第三方算法平台（Halcon、Sherlock）插件库 4. SDK 例程(C、C++、MFC、C++ Builder、C#、VB.Net、QT、Delphi、VB6、Python)
Documentations	SDK 的各种开发文档（SDK 开发手册、API Doc 等）
Drivers	1. GigE 驱动安装、卸载程序 2. U3V 驱动安装、卸载程序 3. x86 智能相机驱动安装、卸载程序 4. DShowFilter 安装、卸载程序
Runtime	SDK 运行时库（包含 GenICam V3.0 第三方库） 包含 32 位和 64 位

1.3.2. Linux 版 SDK

Linux 版本 SDK 的安装路径为“/opt/DahuaTech/MVviewer”,文件(夹)结构如下：

```
/opt/DahuaTech/MVviewer
    |
    ├ bin          : 相机客户端软件（MV Viewer）
    |
    ├ include      : SDK头文件、转码头文件
    |
    ├ lib          : SDK动态库、图像转码库、第三方GenICam库
    |
    ├ module
    |   ├ GigEDriver : GigE相机驱动程序，以及安装、卸载脚本
    |   └ USBDriver  : USB相机的驱动程序，以及安装、卸载脚本
    |
    └ share
        ├ C
        |  └ SimpleSample  : C例程
        |
        ├ C++
        |  ├ SimpleSample  : C++例程
        |  └ QtSample      : QT例程
        |
        └ Python           : Python例程
```

1.4. SDK（C#接口）对应的资源

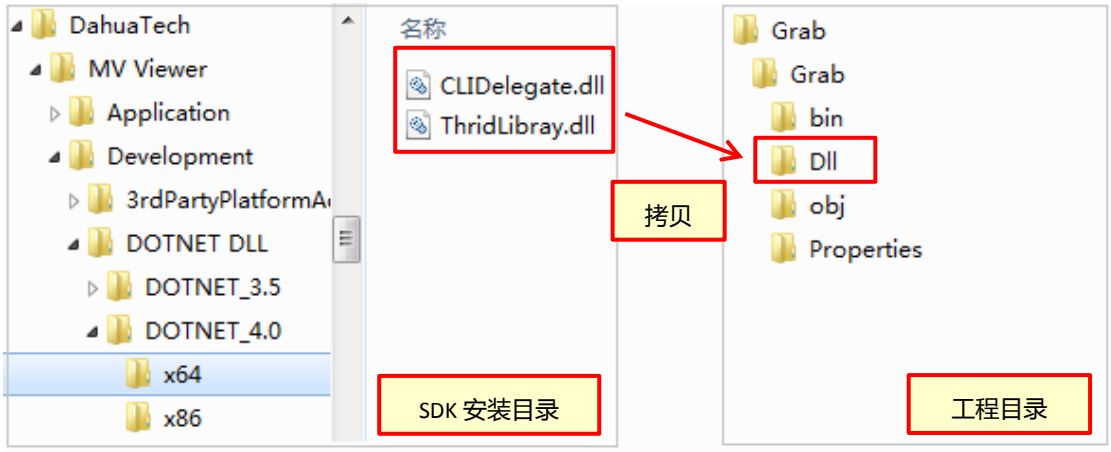
大类别	小类别	文件位置	文件名
SDK 库	DOTNET_3.5（32 位）	【SDK 安装目录】\Development\DOTNET DLL\DOTNET_3.5\x86	CLIDelegate.dll ThridLibrary.dll
	DOTNET_3.5（64 位）	【SDK 安装目录】\Development\DOTNET DLL\DOTNET_3.5\x64	CLIDelegate.dll ThridLibrary.dll
	DOTNET_4.0（32 位）	【SDK 安装目录】\Development\DOTNET DLL\DOTNET_4.0\x86	CLIDelegate.dll ThridLibrary.dll
	DOTNET_4.0（64 位）	【SDK 安装目录】\Development\DOTNET DLL\DOTNET_4.0\x64	CLIDelegate.dll ThridLibrary.dll
开发文档	中文	【SDK 安装目录】\Documentations\	GenICam_API_.Net_CHS.chm 工业相机 SDK 开发手册(C#篇).pdf
	英文	【SDK 安装目录】\Documentations\	GenICam_API_.Net_ENG.chm

1.5. SDK 开发工程配置

1.5.1. VS2010 ~ VS2015

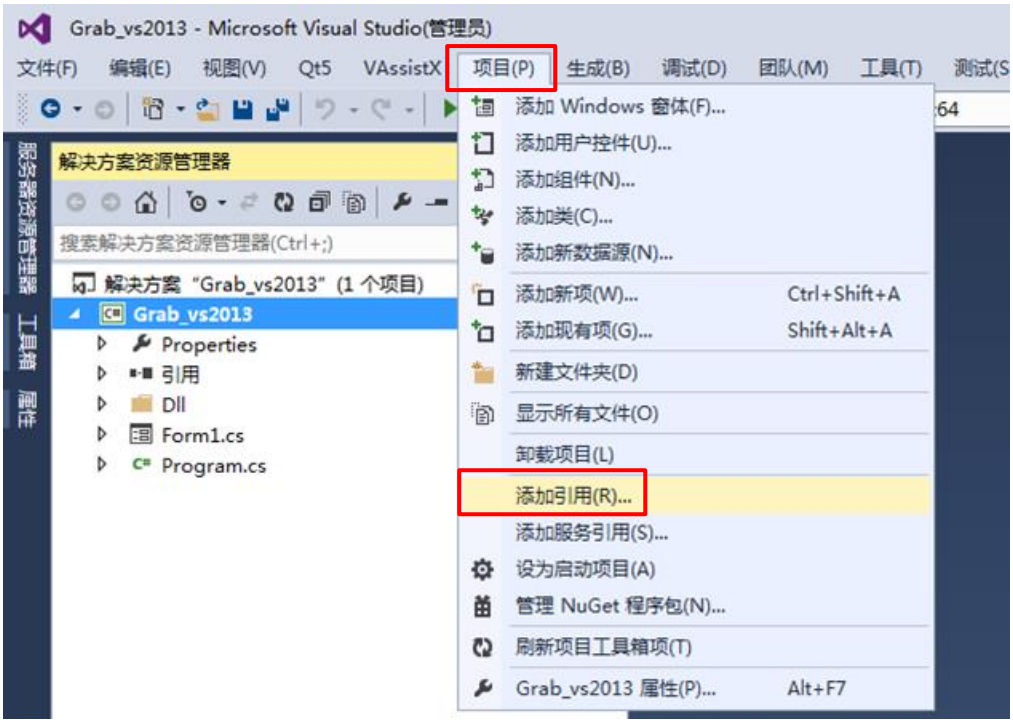
以 DOTNET4.0 64 位工程配置为例，说明配置工程方法。

(1) 拷贝 SDK 相关资源文件

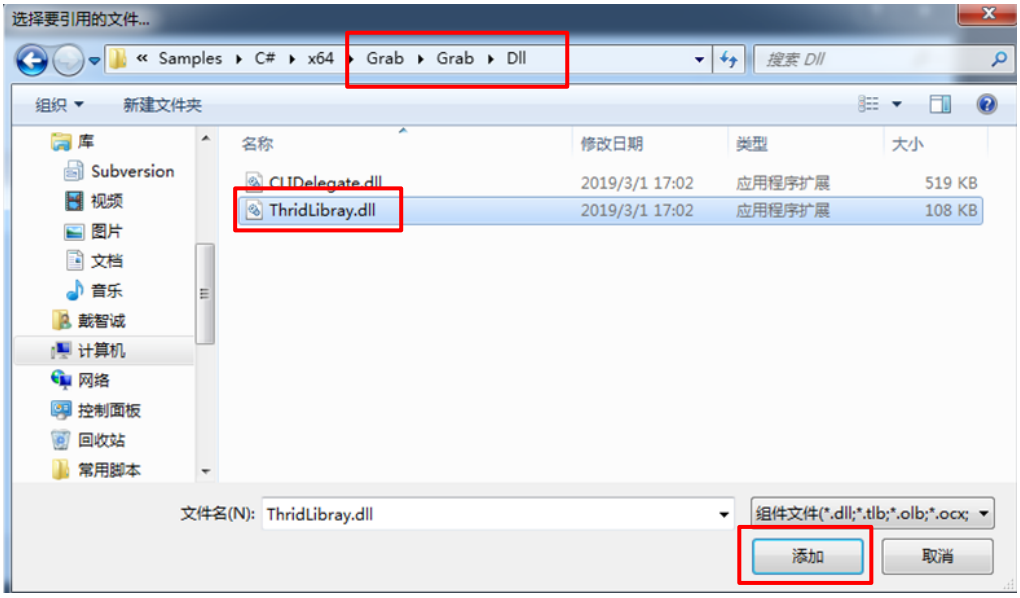


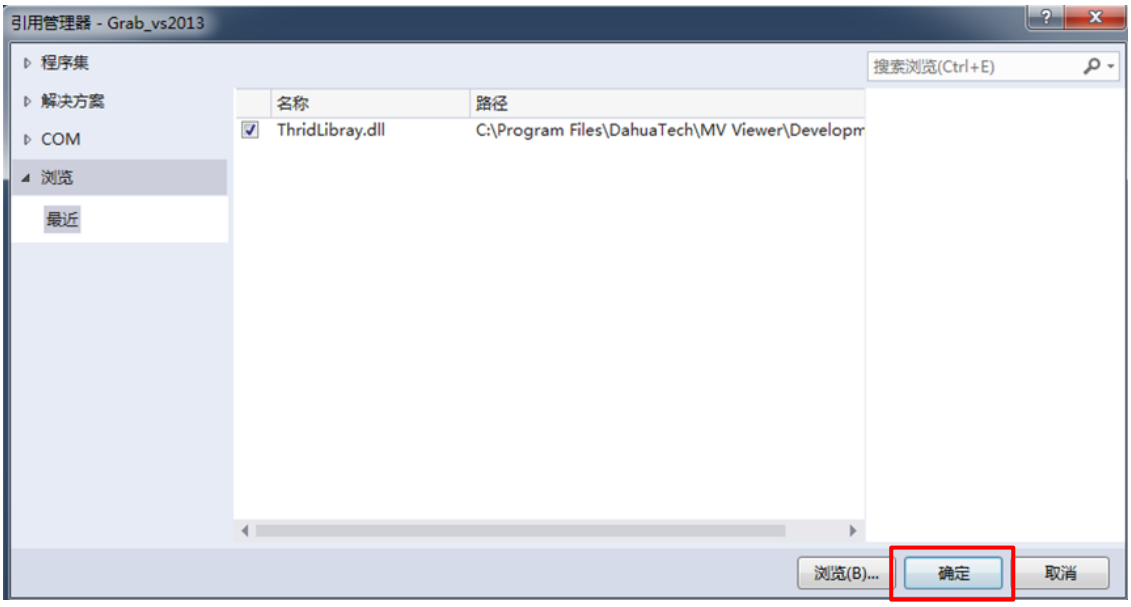
(2) 配置工程

① 项目->添加引用，添加对 ThridLibrary.dll 的引用；

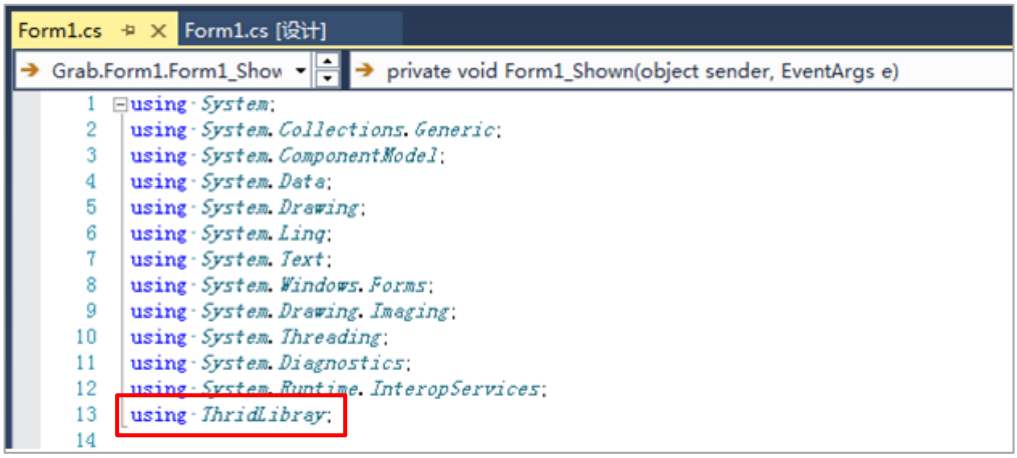



引用管理器 - Grab_vs2013
程序集
解决方案
COM
浏览
最近
搜索浏览(Ctrl+E)
找不到任何项。
浏览(B)...
确定
取消

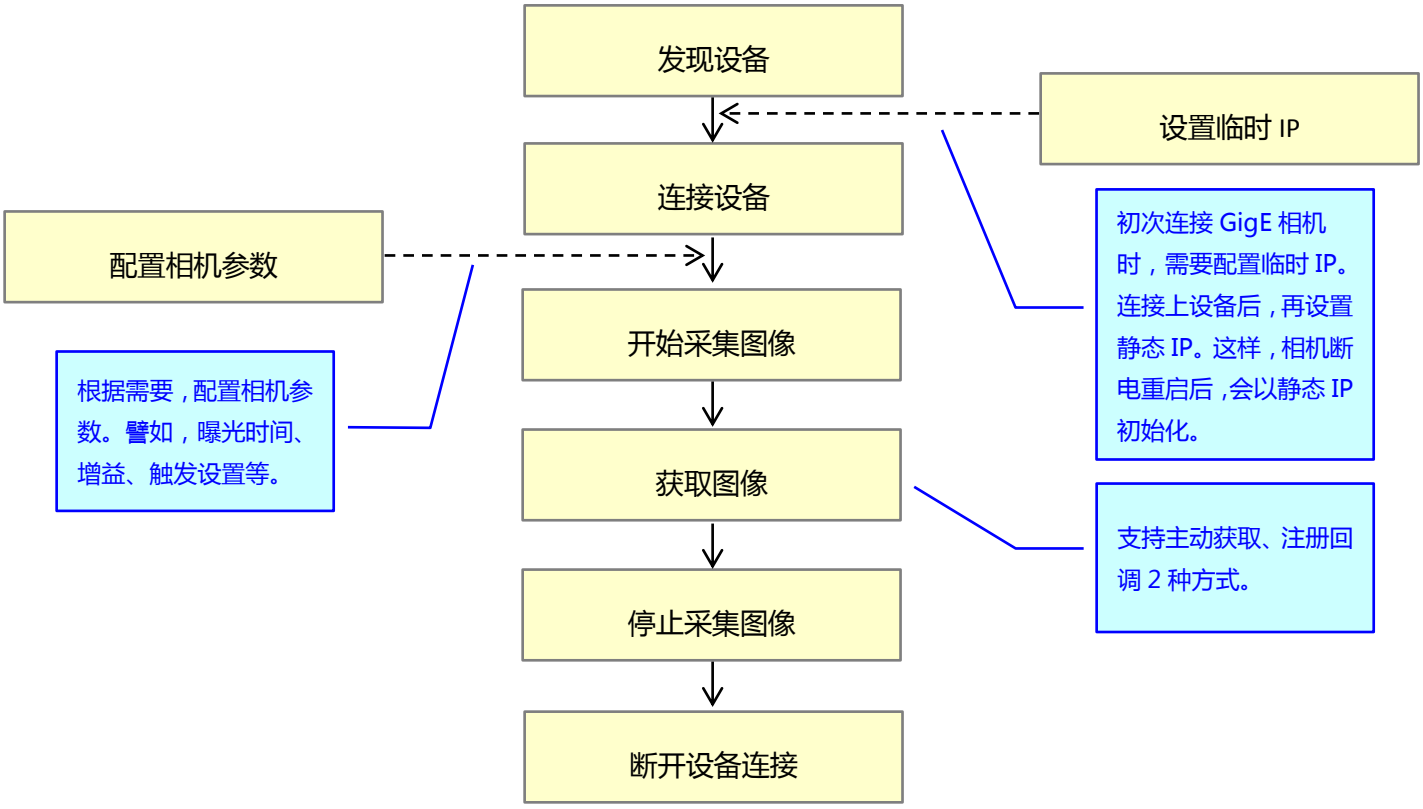

选择要引用的文件...
Samples ▸ C# ▸ x64 ▸ Grab ▸ Grab ▸ Dll
搜索 Dll
组织
新建文件夹
库
Subversion
视频
图片
文档
音乐
戴智诚
计算机
网络
控制面板
回收站
常用脚本
名称
修改日期
类型
大小
CLIDelegate.dll
2019/3/1 17:02
应用程序扩展
519 KB
ThridLibray.dll
2019/3/1 17:02
应用程序扩展
108 KB
文件名(N): ThridLibray.dll
组件文件(*.dll;*.tlb;*.olb;*.ocx;
添加
取消


引用管理器 - Grab_vs2013
程序集
解决方案
COM
浏览
最近
搜索浏览(Ctrl+E)
名称
路径
ThridLibray.dll
C:\Program Files\DahuaTech\MV Viewer\Developm
浏览(B)...
确定
取消

② 在调用代码中添加：using ThridLibray;



2. SDK 的整体调用流程



3. SDK 的各功能接口调用说明

3.1. 设备发现

接口类 ：Enumerator

接口 ：public static List<IDeviceInfo> EnumerateDevices()

功能说明：设备发现接口，枚举所有发现设备。

代码范例：

```
List<IDeviceInfo> li = Enumerator.EnumerateDevices();
```

3.2. 获得设备对象

接口类 ：Enumerator

接口 ：public static IDevice GetDeviceByIndex(int idx)

public static IDevice GetDeviceByKey(string key)
public static IDevice GetDeviceByGigeIP(string ip)

功能说明：需要在执行完“3.1.设备发现”操作后，再调用获得设备对象的接口函数。

根据索引值获取指定设备，索引值从 0 开始，表明是发现的第几台设备；

根据设备 key 值获取指定设备，key 值格式为“设备厂商名:设备序列号”；

根据 GigE 设备 IP 地址获取指定设备。

代码范例：

```
/* 获取搜索到的第一个设备 */
IDevice m_dev = Enumerator.GetDeviceByIndex(0);
```

3.3. 设备连接/断开

接口类 ：IDevice

接口 ：bool Open()

功能说明：连接设备

代码范例：

```
if (!m_dev.Open())
{
    MessageBox.Show(@"连接相机失败");
    return;
}
```

接口 ：bool Close()

功能说明：断开设备连接

代码范例：

```
m_dev.Close();
```

3.4. 设置属性

注意，读写属性之前必须先连接相机。

SDK 提供了 2 种方式读写属性：

- 通过预先给定的常用属性节点
- 通过通用属性方法

3.4.1. 使用常用属性节点

C# SDK 的 ParametrizeNameSet 类中，定义了经常使用的属性的访问函数。

函数名、返回值类型与相机属性一致，如下图所示。

可以通过这个函数操作对应的属性。

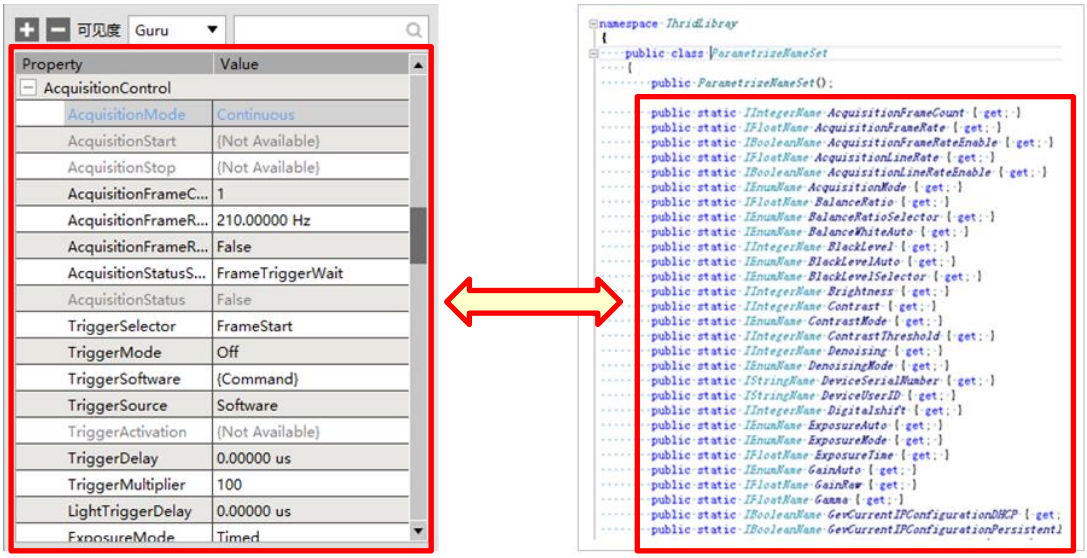


图 2 相机属性与 C# SDK ParametrizeNameSet 类的对应关系

代码范例：

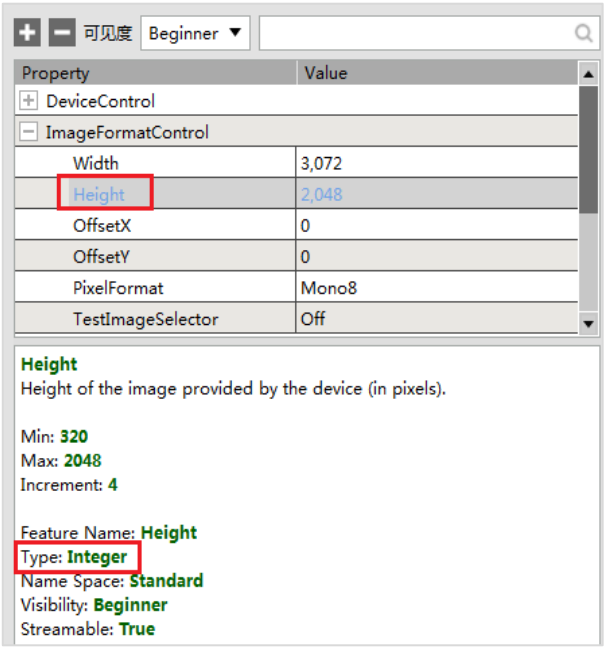
```
/* 以设置图像格式为例（枚举型节点）。m_dev 为已连接上的相机对象（类型：IDevice） */
using (IEnumParameter p = m_dev.ParameterCollection[ParametrizeNameSet.ImagePixelFormat])
{
    /* 属性设置 */
    p.SetValue("Mono8");

    /* 属性读取 */
    string strFormat = p.GetValue();
}
```

3.4.2. 使用通用属性节点

通用属性方法可以使用属性名（字符串，需要注意大小写）构建相应类型的属性节点对象。然后在使用该属性节点对象，读写属性值。具体使用方法如下：

(1) 在相机客户端软件的属性列表里，获取属性的名称、类型和范围等信息

以 Height 属性为例，Type 为 Integer，说明 Height 是整数类型。



(2) 按照属性的类型，选择相应的接口。

相机属性类型 （可以在 MVviewer 上查看）	DotNet SDK 对应的属性类型
Integer	IIntegraParameter
Float	IFloatParameter
Bool	IBooleanParameter
Enumeration	IEnumParameter
String	IStringParameter
Command	ICommandParameter

① 整型属性

```
/* 以 Width 属性为例。m_dev 为已连接上的相机对象（类型：IDevice）*/
using (IIntegraParameter p = m_dev.ParameterCollection[new IntegerName("Width")])
{
    Int nValue = p.GetValue();
    p.SetValue(nValue);
}
```

② 浮点型属性

```
/* 以 ExposureTime 属性为例。m_dev 为已连接上的相机对象（类型：IDevice） */
using (IFloatParameter p = m_dev.ParameterCollection[new FloatName("ExposureTime")])
{
    double dValue = p.GetValue();
    p.setValue(dValue);
}
```

③ 布尔型属性

```
/* 以 SharpnessAuto 属性为例。m_dev 为已连接上的相机对象（类型：IDevice） */
using (IBooleanParameter p = m_dev.ParameterCollection[new BooleanName("SharpnessAuto")])
{
    bool bValue = p.GetValue();
    p.SetValue(true);
}
```

④ 枚举型属性

```
/* 以 SharpnessEnabled 属性为例。m_dev 为已连接上的相机对象（类型：IDevice） */
using (IEnumParameter p = m_dev.ParameterCollection[new EnumName("SharpnessEnabled")])
{
    p.SetValue("On");
    string strValue = p.GetValue();
}
```

⑤ 字符型属性

```
/* 以 DeviceModelName 属性为例。m_dev 为已连接上的相机对象（类型：IDevice） */
using (IStringParameter p = m_dev.ParameterCollection[new StringName("DeviceModelName")])
{
    string strValue= p.GetValue();
}
```

⑥ 命令型属性

```
/* 以 TriggerSoftware 属性为例。m_dev 为已连接上的相机对象（类型：IDevice） */
using (ICommandParameter p = m_dev.ParameterCollection[new
CommandName("TriggerSoftware")])
{
    p.Execute();
}
```

3.5. 采集图像

3.5.1 设置图像接收缓存个数

接口类 ：IStream

接口 ：bool SetBufferCount(int nCount)

功能说明：设置图像帧接收缓存大小。

该接口要在执行开始采集图像前调用。在图像采集状态下调用该接口会失败。

默认缓存个数为 8。相机图像分辨率较高的情况下，可设置成 2 或者 4 比较小的值，，防止内存分配失败。

[IN]nCount，缓冲帧个数。

代码范例：

```
m_dev.StreamGrabber.SetBufferCount(4);
```

3.5.2 开始/停止采集图像

(1) 主动方式采图

接口类 ：IStream

接口 ：bool Start(GrabStrategyEnum strategy, GrabLoop grabLoopType);

功能说明：开始拉流。

[IN] strategy,拉流策略，按序列顺序取帧或取最新帧策略；

[IN] grabLoopType，主动取图使用 ProvidedByUser。

代码范例：

```
if (!m_dev.StreamGrabber.Start(GrabStrategyEnum.grabStrartegySequential,
GrabLoop.ProvidedByUser))
{
        MessageBox.Show(@"开启码流失败");
        return;
 }
```

接口 ：bool ShutdownGrab()

功能说明：停止拉流。

代码范例：

```
m_dev.ShutdownGrab();
```

(2) 注册回调方式采图

接口类 ：IDevice

接口 ：bool GrabUsingGrabLoopThread()

功能说明：使用内部线程捕获帧数据，适用于注册回调方式取图。

代码范例：

```
if (!m_dev.GrabUsingGrabLoopThread())
{
    MessageBox.Show(@"开启码流失败");
    return;
}
```

接口 ：bool ShutdownGrab()

功能说明：停止拉流。

代码范例：

```
m_dev.ShutdownGrab();
```

3.5.3 获取图像

(1) 主动方式采图

接口类 ：IDevice

接口 ：bool WaitForFrameTriggerReady(out IGrabbedRawData data, int timeoutMs)

功能说明：同步等待帧数据，内部做数据拷贝直接返回原始码流数据。

配套 3.5.2-(1)的开始/停止采集图像的函数使用。

[OUT] data，返回捕获的流数据;

[IN]捕获数据等待超时时间。

代码范例：

```
IGrabbedRawData data;
if (true == m_dev.WaitForFrameTriggerReady(out data, 1000))
{
    Trace.WriteLine(data.BlockID);
    var bitmap = data.ToBitmap(false);
}
```

(2) 注册回调方式采图

接口类 ：IStream

接口 ：event EventHandler<GrabbedEventArgs> ImageGrabbed;

功能说明：发布取图上报事件。GrabbedEventArgs,流数据回调事件参数，包含图像宽、高、格式、裸数据等信息。配套 3.5.2-(2)的开始/停止采集图像的函数使用。

代码范例：

```
/* 注册码流回调事件 */
m_dev.StreamGrabber.ImageGrabbed += OnImageGrabbed;

/* 码流数据回调 */
private void OnImageGrabbed(Object sender, GrabbedEventArgs e)
{
    long nBlockId = e.GrabResult.BlockID;
}
```

(3) 获取图像裸数据

回调函数或主动取图，获取到 IGrabbedRawData 类型的对象，通过该对象即可获得对应的图像裸数据。

① IGrabbedRawData::Image，获得 BYTE 数组，存放着图像裸数据。

② IGrabbedRawData::Raw，获得指向图像数据的指针。

注：异步处理不适用，指针指向的数据随时可能被释放。

(4) 图像格式转换

① 直接转成 Bitmap 对象

接口类 ：IGrabbedRawData

接口 ：Bitmap ToBitmap(bool color);

功能说明：生成 Bitmap 图片对象。回调函数或主动取图，获取到 IGrabbedRawData 类型的对象，通过该对象即可直接转成 Bitmap 对象。

[IN]color, 转换后图片是否为彩色。

代码范例：

```
var bitmap = frame.ToBitmap(false);
```

② 将图像裸数据转换成 RGB 图像数据

接口类 ：RGBFactory

接口 ：public static int EncodeLen(int width, int height, bool color);

功能说明：获取 RGB 编码后数据长度。

[IN] width, 图像宽度

[IN] height, 图像高度

[IN] color, 是否为彩色

代码范例：

```
int nRGB = RGBFactory.EncodeLen(frame.Width, frame.Height, bColor);
```

接口类：RGBFactory

接口：public static void ToRGB(byte[] raw,int nWidth,int nHeight,bool color, string pixel,IntPtr pRGB,int nRGBSize)
public static void ToRGB(byte[] raw,int nWidth,int nHeight,bool color, GvspPixelFormatType tPixel,IntPtr pRGB,int nRGBSize)
public static void ToRGB(IntPtr pRaw, int nLen, int nWidth, int nHeight, bool color, String pixel, IntPtr pRGB, int nRGBSize)
public static void ToRGB(IntPtr pRaw, int nLen, int nWidth, int nHeight, bool color, GvspPixelFormatType tPixel, IntPtr pRGB, int nRGBSize)

功能说明：将裸数据转换到 RGB 数据。

代码范例：

```
IntPtr pData = Marshal.AllocHGlobal(nRGB);
RGBFactory.ToRGB(frame.Image,
                 frame.Width,
                 frame.Height,
                 true,
                 frame.PixelFmt,
                 pData,
                 nRGB);
```

3.6事件通知

3.6.1 设备连接状态事件

接口类 ：IDevice

接口 ：event EventHandler<EventArgs> CameraClosed;
event EventHandler<EventArgs> CameraOpened;
event EventHandler<EventArgs> ConnectionLost;

功能说明：发布相机连接、断开和断线事件。

代码范例：

```
/* 注册连接事件 */
m_dev.CameraOpened      += OnCameraOpen;
m_dev.ConnectionLost += OnConnectLoss;
m_dev.CameraClosed      += OnCameraClose;

// 相机打开回调。进入该回调表示相机打开
private void OnCameraOpen(object sender, EventArgs e)
{
    // 用户自定义操作
}

// 相机关闭回调，进入该回调表示相机关闭相机打开
private void OnCameraClose(object sender, EventArgs e)
{
    // 用户自定义操作
}

// 相机断线回调
private void OnConnectLoss(object sender, EventArgs e)
{
    // 用户自定义操作

    // 关闭相机操作
    m_dev.ShutdownGrab();
    m_dev.Dispose();
    m_dev = null;
}
```

3.6.2 流事件

接口类 ：IDevice

接口 ：event EventHandler<StreamEventArgs> StreamArgEvent;

功能说明：发布流事件。流通道事件包括以下事件类型：

EN_EVENT_STATUS_NORMAL	正常
EN_EVENT_STATUS_LOST_FRAME	缓存被覆盖导致的丢帧
EN_EVENT_STATUS_LOST_PACKET	丢包导致的丢帧
EN_EVENT_STATUS_IMAGE_ERROR	图像错误
EN_EVENT_STATUS_CHANNEL_ERROR	通道错误

代码范例：

```
m_dev.StreamArgEvent += OnStreamChannelEvent;

private void OnStreamChannelEvent(object sender, StreamEventArgs e)
{
    ulong uTime = e.Timestamp;
    ulong nId = e.BlockID;
    uint uChannel = e.Channel;
    StreamEventStatus status = e.StreamEventStatus;
    ulong ulTime = e.Timestamp;
}
```

3.6.3 相机消息事件

接口类 ：IDevice

接口 ：event EventHandler<MsgChannelArgs> MsgChannelArgEvent;

功能说明：发布消息通道事件。消息通道事件包括以下事件类型：

MSG_CHANNEL_EVENT_EXPOSURE_END
MSG_CHANNEL_EVENT_FRAME_TRIGGER
MSG_CHANNEL_EVENT_FRAME_START
MSG_CHANNEL_EVENT_ACQ_START
MSG_CHANNEL_EVENT_ACQ_TRIGGER
MSG_CHANNEL_EVENT_DATA_READ_OUT

代码范例：

```
m_dev.MsgChannelArgEvent += OnMsgChannelEvent;

private void OnMsgChannelEvent(object sender, MsgChannelArgs e)
{
    ulong    ulBolckId = e.BlockID ;
    ushort usChannelId = e. ChannelID ;
    MsgChannelEvent msgId = e.EventID;
    List<string> oList = e.ParamNames;
    ulong ulTime = e.Timestamp;
}
```

4 第三方平台的图像对象转换方法

4.1. Halcon 对象（HObject）

4.1.1. 相机图像格式为 Mono8 时

相机图像格式为 Mono8 时，直接用采集到的图像数据构建 Halcon 对象。

此处，假设采集到的图像帧（IGrabbedRawData）对象为 frame。裸数据指针

```
int nRGB = RGBFactory.EncodeLen(frame.Width, frame.Height, false);
IntPtr pData = Marshal.AllocHGlobal(nRGB);
Marshal.Copy(frame.Image, 0, pData, frame.ImageSize);

HObject ho_image;

HOperatorSet.GenImage1Extern(out ho_image,
                             "byte",
                             nWidth,
                             nHeight,
                             (HTuple)pData,
                             0);
```

4.1.2. 相机图像格式为 Mono8 以外时（主要是彩色格式）

(1) 将采集到的相机图像数据，转换成 BGR24 格式

(2) 使用转换后的 BGR24 格式图像数据，构建 Halcon 图像对象（HObject）

```
int nRGB = RGBFactory.EncodeLen(frame.Width, frame.Height, true);
IntPtr pData = Marshal.AllocHGlobal(nRGB);
RGBFactory.ToRGB(frame.Image,
                 frame.Width,
                 frame.Height,
                 true,
                 frame.PixelFmt,
                 pData,
                 nRGB);

HObject ho_image;
HOperatorSet.GenImageInterleaved(out ho_image,
                                 (HTuple)pData,
                                 "bgr",
                                 nWidth,
                                 nHeight,
                                 0,
                                 "byte",
                                 nWidth,
                                 nHeight,
                                 0, 0, 8, 0);
```

5 配套例程说明

SDK（DOTNET 接口）提供了基于 DOTNET4.0 及以上版本的例程（含 32/64 位）。具体如下：

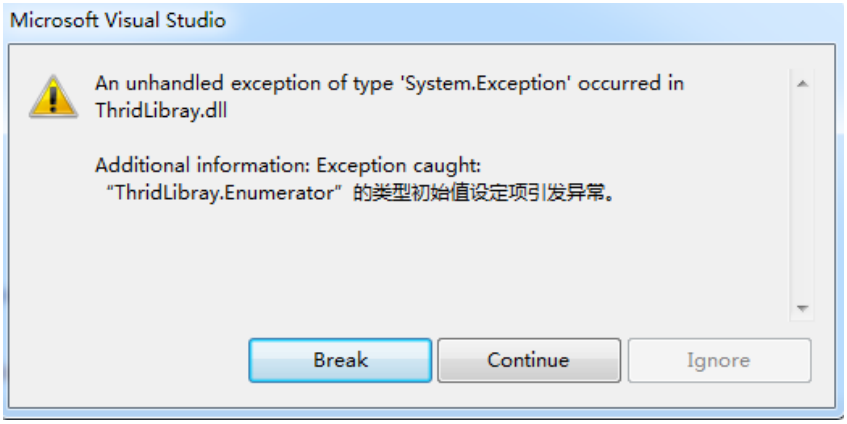
例程路径： 【SDK 安装目录】\Development\Samples\C#\

例程名	说明
CommonInterface	通用属性接口 Demo
Grab	自由取图显示 Demo

LineTrigger	硬件触发取图显示 Demo
SoftwareTrigger	软件触发取图 Demo

6 常见问题&注意点

6.1. 程序启动时 ,执行设备发现处理提示“ThridLibrary.Enumerator 的类型初始值设定项引发异常



原因 1 ：运行环境中缺少 vc100 运行时库

解决方法 1：安装 vc100 运行时库。或者更新为 Ver2.0.0 以及之后版本的 SDK。

原因 2 ：path 环境有问题，无法通过环境变量找到 MVSDKmd.dll 库。此时，打开 MV Viewer 也会失败，提示找不到 MVSDKmd.dll 库。

解决方法 2：重新安装 MV Viewer（SDK）或者手动编辑 path 环境变量。

原因 3 ：Win10 操作系统权限问题

解决方法 3：以管理员身份运行 VS 编译环境 。

6.2. 程序运行过程中报“OutOfMemory”异常

原因 ：图像显示处理的速度跟不上图像采集的速度，导致待显示的图像不断增加，

内存不断上升，最终内存占用过大。

解决方法 1 ：降低显示帧率。没有必要帧帧显示，帧率 23FPS 时，肉眼看就已经流畅。

解决方法 2 ：不使用 picturebox，使用 GDI 显示方式

6.3. 在 ParametrizeNameSet 中找不到某些属性的访问接口，不知道该如何读取/设置这些属性。

原因： SDK 仅提供了常用属性的直接访问接口函数

解决方法：使用通用属性接口，将属性名作为字符串参数传入，构建对应属性的节点，然后进行属性值的读取和设置。具体可以参照按照下述例程：

[安装目录]\MV Viewer\Development\Samples\C#\x64\CommonInterface

6.4. 使用通用属性接口获取相机属性，无法正常访问和修改属性值。

客户使用的是VB.Net开发。
可以编译通过，但取出的属性值为空。

```
Using ep As ThridLibray.IEnumParameter = m_dev.ParameterCollection(New EnumName(" SharpnessAuto "))
   If Not ep Is Nothing Then
      Dim kk As List(Of String) = ep.GetAllValues
      ep.SetValue(kk(0))
      ep.Dispose()
   End If
End Using
```

原因 ：访问的属性值的类型与实际类型不一致。如 SharpnessAuto 是 Bool 型的属性，客户在代码中以将其当做 Enum（枚举）类型访问。

解决方法：在 MVViewer 的属性窗口中确认对象属性的类型，然后构建对应类型的变量。

6.5. 使用回调方式采图。取到图像后，在另一个线程中使用图像数据指针做数据处理，程序崩溃。

原因　　：图像采集的回调处理已经返回，图像内存其实已经被释放，导致客户程序崩溃。

解决方法：在回调函数中，将图像数据拷贝到客户程序自己申请的内存中。在图像处理结束后，再由客户程序自主释放内存。

- END -