Basler pylon 2.0

PROGRAMMER'S GUIDE AND API REFERENCE

Document Number: AW000131
Version: 05 Language: 000 (English)
Release Date: 18 February 2008

BASLER
VISION TECHNOLOGIES

For customers in the U.S.A.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

You are cautioned that any changes or modifications not expressly approved in this manual could void your authority to operate this equipment.

The shielded interface cable recommended in this manual must be used with this equipment in order to comply with the limits for a computing device pursuant to Subpart J of Part 15 of FCC Rules.

For customers in Canada

This apparatus complies with the Class A limits for radio noise emissions set out in Radio Interference Regulations.

Pour utilisateurs au Canada

Cet appareil est conforme aux normes Classe A pour bruits radioélectriques, spécifiées dans le Règlement sur le brouillage radioélectrique.

Life Support Applications

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Basler customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Basler for any damages resulting from such improper use or sale.

Warranty Note

Do not open the housing of the camera. The warranty becomes void if the housing is opened.

All material in this publication is subject to change without notice and is copyright Basler Vision Technologies.

Contacting Basler Support Worldwide

Europe:

Basler AG
An der Strusbek 60 - 62
22926 Ahrensburg
Germany

Tel.: +49-4102-463-500
Fax.: +49-4102-463-599

vc.support.europe@baslerweb.com

Americas:

Basler, Inc.
855 Springdale Drive, Suite 160
Exton, PA 19341
U.S.A.

Tel.: +1-877-934-8472
Fax.: +1-877-934-7608

vc.support.usa@baslerweb.com

Asia:

Basler Asia Pte. Ltd
8 Boon Lay Way
03 - 03 Tradehub 21
Singapore 609964

Tel.: +65-6425-0472
Fax.: +65-6425-0473

vc.support.asia@baslerweb.com

www.basler-vc.com

Contents

1 pylon 2.0 Programmer's Guide **1**

1.1 Introduction 1

1.2 Building Applications with pylon 3

1.3 Programming with pylon 4

1.4 Migrating From pylon 1.0 to pylon 2.0 43

1.5 How to Control the Location of the Camera Description Files Used by pylon 46

1.6 Software Licensing Information 47

2 Module Documentation **49**

2.1 Public interface and definitions in Pylon 49

2.2 The GigE specific definitions 52

2.3 The IEEE1394 specific definitions 53

2.4 Abstract interface classes of the GenApi module 54

2.5 Public utilities of the GenApi module 58

2.6 Implementation parts of the GenApi module which must be public 60

2.7 Public utilities of the GenApi Log module 63

2.8 Public utilities of the GenICam Base module 64

2.9 Implementation parts of the GenICam Base module 66

3 Namespace Documentation **69**

3.1 Pylon Namespace Reference 69

3.2 Pylon::Key Namespace Reference 78

4 Class Documentation **79**

4.1 GenApi::AttachStatistics_t Struct Reference 79

4.2 GenApi::CChunkAdapter Class Reference 80

4.3 GenApi::CChunkAdapterDcam Class Reference 82

4.4 GenApi::CChunkAdapterGEV Class Reference 84

4.5 GenApi::CNodeMapRef Class Reference 85

4.6 GenApi::CPointer< T, B > Class Template Reference 87
4.7 GenApi::Function_NodeCallbackBody< Function > Class Template Reference 89
4.8 GenApi::GVCP_CHUNK_TRAILER Struct Reference 90
4.9 GenApi::IBase Struct Reference 91
4.10 GenApi::IBoolean Struct Reference 92
4.11 GenApi::ICategory Struct Reference 93
4.12 GenApi::ICommand Struct Reference 94
4.13 GenApi::IEnumEntry Struct Reference 95
4.14 GenApi::IEnumeration Struct Reference 96
4.15 GenApi::IEnumerationT< EnumT > Struct Template Reference 99
4.16 GenApi::IFloat Struct Reference 101
4.17 GenApi::IInteger Struct Reference 102
4.18 GenApi::INode Struct Reference 103
4.19 GenApi::INodeMap Struct Reference 105
4.20 GenApi::IPort Struct Reference 106
4.21 GenApi::IRegister Struct Reference 107
4.22 GenApi::ISelector Struct Reference 108
4.23 GenApi::IString Struct Reference 109
4.24 GenApi::IValue Struct Reference 110
4.25 GenApi::Member_NodeCallbackBody< Client, Member > Class Template Reference 111
4.26 GenApi::NodeCallback Class Reference 112
4.27 GenApi::NodeCallbackBase Class Reference 113
4.28 GenICam::AccessException Class Reference 114
4.29 GenICam::DynamicCastException Class Reference 115
4.30 GenICam::ExceptionReporter< E > Class Template Reference 116
4.31 GenICam::gcstring Class Reference 117
4.32 GenICam::GenericException Class Reference 118
4.33 GenICam::InvalidArgumentException Class Reference 119
4.34 GenICam::LogicalErrorException Class Reference 120
4.35 GenICam::OutOfRangeException Class Reference 121
4.36 GenICam::PropertyException Class Reference 122
4.37 GenICam::RuntimeException Class Reference 123
4.38 GenICam::TimeoutException Class Reference 124
4.39 Pylon::AccessModeSet Class Reference 125
4.40 Pylon::AlertableWaitObject Class Reference 126
4.41 Pylon::AutoLock Class Reference 127

4.42 Pylon::Base_Callback1Body< P1 > Class Template Reference 128
4.43 Pylon::Callback1< P1 > Class Template Reference 129
4.44 Pylon::CBasler1394Camera Class Reference 130
4.45 Pylon::CBasler1394DeviceInfo Class Reference 153
4.46 Pylon::CBasler1394EventGrabber Class Reference 155
4.47 Pylon::CBasler1394StreamGrabber Class Reference 156
4.48 Pylon::CBasler1394TI Class Reference 158
4.49 Pylon::CBaslerGigECamera Class Reference 159
4.50 Pylon::CBaslerGigEDeviceInfo Class Reference 195
4.51 Pylon::CBaslerGigEEventGrabber Class Reference 197
4.52 Pylon::CBaslerGigEStreamGrabber Class Reference 199
4.53 Pylon::CBaslerGigETLParams Class Reference 201
4.54 Pylon::CChunkParser Class Reference 202
4.55 Pylon::CDeviceInfo Class Reference 205
4.56 Pylon::CEventGrabberProxy Class Reference 207
4.57 Pylon::CInfoBase Class Reference 208
4.58 Pylon::CLock Class Reference 209
4.59 Pylon::CNodeMapProxy Class Reference 210
4.60 Pylon::CPixelFormatConverter Class Reference 211
4.61 Pylon::CPixelFormatConverterBayer Class Reference 214
4.62 Pylon::CPixelFormatConverterGamma Class Reference 217
4.63 Pylon::CPixelFormatConverterGammaPacked Class Reference 220
4.64 Pylon::CPixelFormatConverterMonoPacked Class Reference 224
4.65 Pylon::CPixelFormatConverterMonoXX Class Reference 228
4.66 Pylon::CPixelFormatConverterRGB Class Reference 231
4.67 Pylon::CPixelFormatConverterTruncate Class Reference 234
4.68 Pylon::CPixelFormatConverterTruncatePacked Class Reference 238
4.69 Pylon::CPixelFormatConverterYUV422 Class Reference 242
4.70 Pylon::CPixelFormatConverterYUV422YUYV Class Reference 245
4.71 Pylon::CPylonDeviceProxy Class Reference 248
4.72 Pylon::CStreamGrabberProxy Class Reference 249
4.73 Pylon::CTlFactory Class Reference 250
4.74 Pylon::DeviceInfoList Class Reference 252
4.75 Pylon::EventResult Class Reference 253
4.76 Pylon::Function_CallbackBody< Function, P > Class Template Reference 254
4.77 Pylon::GrabResult Class Reference 255

4.78 Pylon::IChunkParser Interface Reference 258
4.79 Pylon::IDevice Interface Reference 261
4.80 Pylon::IDeviceFactory Interface Reference 263
4.81 Pylon::IEventAdapter Interface Reference 265
4.82 Pylon::IEventGrabber Interface Reference 266
4.83 Pylon::IInfo Interface Reference 268
4.84 Pylon::IProperties Interface Reference 269
4.85 Pylon::IPylonDevice Interface Reference 270
4.86 Pylon::IStreamGrabber Interface Reference 275
4.87 Pylon::LockableObject< Object > Class Template Reference 278
4.88 Pylon::LockableObject< Object >::Lock Class Reference 279
4.89 Pylon::Member_CallbackBody< Client, Member, P > Class Template Reference 280
4.90 Pylon::PylonAutoInitTerm Class Reference 281
4.91 Pylon::SystemWaitObject< NativeObject > Class Template Reference 282
4.92 Pylon::TlInfoList Class Reference 283
4.93 Pylon::WaitObject Class Reference 284
4.94 Pylon::WaitObjects Class Reference 285
4.95 Pylon::WaitObjectSet< NativeObject > Class Template Reference 287

Chapter 1

pylon 2.0 Programmer's Guide

1.1 Introduction

The pylon API is the Basler Components C++ application programming interface (API) for FireWire IEEE 1394 and Gigabit Ethernet (GigE) cameras. The pylon API offers a generic, unified programming interface that hides most of the differences between interface technologies. The pylon API allows easy migration from one camera interface to another without significant code changes.

1.1.1 Where to Find the Camera Parameters

All available parameters for the Basler GigE Vision and IEEE1394 FireWire cameras are listed in the documentation of the

- `Pylon::CBaslerGigECamera`
- `Pylon::CBasler1394Camera`

classes.

1.1.2 Build Environment (Includes, Compiler Switches, ...)

The Building Applications with pylon section describes all necessary modifications of your build environment to build pylon based applications.

1.1.3 Programmer's Guide

This documentation includes a comprehensive Programmer's Guide: Programming with pylon

1.1.4 Restricted Compatibility of the API

The programming interface for pylon 2.0 is in some respects not backwards compatible. The Migrating From pylon 1.0 to pylon 2.0 section describes the modifications needed to rebuild an application that was developed with pylon 1.x.

1.1.5 Controllig the Location of Camera Description Files

With pylon 2.0 a mechanism has become available allowing to specify the location of camera description files used to access camera parameters. The mechanism is described in the How to Control the Location of the Camera Description Files Used by pylon section.

1.1.6 Precautions

Caution: The code snippets provided in this manual are included as sample code only. Inappropriate code may cause your camera to function differently than expected and may compromise your application. To ensure that the snippets will work properly in your application, you must adjust them to meet your specific needs and must test them thoroughly prior to use.

1.2 Building Applications with pylon

The installation package for the pylon SDK installs all necessary header and library files to build applications using the pylon API. In the following, we denote the folder where the pylon SDK was installed as <SDK ROOT>, which may proxy for e.g. C:\Program Files\Basler\Pylon 2.0. The pylon installation procedure sets the environment variable PYLON_ROOT pointing to <SDK ROOT>\pylon.

As part of the pylon SDK, sample programs are also installed at <SDK ROOT>\Samples. These samples include Microsoft Visual Studio 2003 & 2005 solution and project files demonstrating how to set up the build environment to build pylon based applications.

1.2.1 Include Path

The following directories must be added to the compiler's include path:

- $(PYLON_ROOT)\include
- $(PYLON_ROOT)\..\genicam\library\cpp\include

These folders must be added to the compiler's include path.

1.2.2 Library Path and Library Files

Applications using the pylon API must be linked against the several import libaries. This is automatically done by including the PylonIncludes.h file that contains the necessary #pragma comment lib statements.

```
#include <pylon/PylonIncludes.h>
```

The following directories must be added to the linker's search path for .lib files.

For building 32 bit applications:

- $(PYLON_ROOT)\lib\Win32
- $(PYLON_ROOT)\..\genicam\library\CPP\Lib\Win32_i86

For building 64 bit applications:

- $(PYLON_ROOT)\lib\x64
- $(PYLON_ROOT)\..\genicam\library\CPP\Lib\Win64_x64

1.2.3 Compiler Settings

The pylon API relies on C++ runtime type information. To compile applications using the pylon API, the generation of runtime type information must be enabled for the compiler. For the Visual Studio compiler, either use the /GR compiler switch or set the "Enable Runtime Type Info" parameter in the Visual Studio project settings to 'yes' (Configuration Properties->C/C++->Language property page).

Since C++ exceptions are used for error handling, C++ exceptions must be enabled in the compiler settings (default setting for Visual Studio 2003 and 2005).

1.2.4 Environment Variables

The pylon installation procedure sets the following environment variables, that are used by pylon based applications at runtime:

- PYLON_ROOT: used to locate transport layer DLLs
- GENICAM_ROOT_V1_1: used to locate DLLs in which GenApi Node Maps are residing
- GENICAM_CACHE: specifies the location for GenApi's XML file cache

The pylon installer extends the PATH environment by the following directories:

For the 32 bit Windows operating system:

- <SDK ROOT>\genicam\bin\win32_i86
- <SDK ROOT>\pylon\bin\Win32

For the 64 bit Windows operating system:

- <SDK ROOT>\genicam\bin\win64_x64
- <SDK ROOT>\pylon\bin\x64
- <SDK ROOT>\genicam\bin\win32_i86
- <SDK ROOT>\pylon\bin\win32

The extensions are needed to find the pylon dynamic link libraries (.dll files) at runtime. An alternative to changing the PATH environment variable is planned for the future.

1.3 Programming with pylon

1.3.1 The Architecture of pylon

This section gives a short introduction to the most important concepts of the pylon API.

1.3.1.1 Transport Layers

The term transport layer is used as an abstraction for a physical interface such as FireWire or GigE. For each of these interfaces, there are drivers that allow access to camera devices. As an abstraction of these drivers, a transport layer provides the following functionality:

- Device discovery (also called device enumeration)
- Reading and writing camera registers
- Grabbing images
- Retrieving event messages
- Configuring the transport layer itself (e.g., timeouts)

- Creating camera objects
- Deleting camera objects

In pylon, each transport layer is housed in a separate dynamic link library (DLL). The following DLLs are currently available:

- Pylon1394 for IIDC 1394 compliant FireWire cameras
- PylonGigE for Gigabit Ethernet Cameras using the GigE Vision protocol

An application program doesn't access transport layer DLLs directly. The Transport Layer Factory (see below) is used to create Transport Layer objects, each of which represents a transport layer DLL.

Transport Layer objects are used to:

- Enumerate camera devices accessible by the transport layer
- Create Camera objects used to access camera devices
- Access transport layer specific parameters

1.3.1.2 Transport Layer Factory

Enumeration of available transport layers and creation of Transport Layer objects is done by the `Pylon::CTlFactory` class. For convenience, the `Pylon::CTlFactory` class also provides methods for enumerating all available cameras across multiple transport layers and for creating camera devices without using Transport Layer objects.

1.3.1.3 Camera Objects

In pylon, physical camera devices are represented by Camera objects. Camera objects are created by either a Transport Layer object or the Transport Layer Factory (`Pylon::CTlFactory`).

A Camera object is used for:

- Accessing a camera (i.e., opening the Camera object)
- Accessing the camera's parameters
- Accessing a Stream Grabber object used for grabbing images (see below)
- Accessing an Event Grabber object used for retrieving event messages (see below)
- Accessing a Chunk Parser object used for analyzing a self-descriptive, structured data stream (see below)

1.3.1.4 Stream Grabbers

The pylon architecture allows a camera object to deliver one or more streams of image data. To grab images from a stream, a Stream Grabber object is needed. Stream Grabber objects cannot be created directly by an application. They are managed by Camera objects, which create and pass out Stream Grabbers. All Stream Grabbers implement the Pylon::IStreamGrabber interface. This means that images are grabbed from streams for all transport layers in exactly the same way. More details about grabbing images are described below in the Grabbing Images section.

1.3.1.5 Event Grabbers

Basler GigE Vision cameras can send event messages. Event Grabber objects are used to receive event messages. How to retrieve and process event messages is described below in the Handling Camera Events section.

1.3.1.6 Chunk Parsers

Basler Cameras can send additional information appended to the image data. A so called Chunk Mode can be activated. When in Chunk Mode, the cameras send an extended datastream including the image data combined with added information like, e.g. a frame number or a time stamp. The extended data stream is self descriptive. Pylon Chunk Parser objects are used for parsing the extended data stream and for providing access to the added information. Chunk Parser objects are described in the Chunk Parser: Accessing Chunk Features section.

1.3.1.7 GenApi Node Maps

For camera configuration and for accessing other parameters, the pylon API uses the technologies defined by the GenICam standard hosted by the European Machine Vision Association (EMVA). The GenICam specification (http://www.GenICam.org) defines a format for camera description files. These files describe the configuration interface of GenICam compliant cameras. The description files are written in XML (e-Xtensible Markup Language) and describe camera registers, their interdependencies, and all other information needed to access high-level features such as `Gain`, `Exposure Time`, or `Image Format` by means of low-level register read and write operations.

The elements of a camera description files are represented as software objects called Nodes. For example, a node can represent a single camera register, a camera parameter such as Gain, a set of available parameter values, etc. Each node implements the `GenApi::INode` interface.

The nodes are linked together by different relationships as explained in the GenICam standard document available at www.GenICam.org. The complete set of nodes is stored in a data structure called Node Map. At runtime, a Node Map is instantiated from a XML description.

With the code generators of GenICam's GenApi module, a generic programming interface is created from a camera description file.

In pylon, Node Maps are not only used to represent camera parameters. Parameters of other pylon objects like Transport Layer objects, Stream Grabbers, and Event Grabbers are also exposed via GenApi Node Maps.

Examples:

- The `Pylon::IPylonDevice` interface has the `Pylon::IPylonDevice::GetNodeMap()` method, which returns the Node Map containing all GenApi nodes representing the whole set of camera parameters.
- The `Pylon::IStreamGrabber::GetNodeMap()` method is used to access a Stream Grabber's parameters.

1.3.2 Error Handling

In case of errors, the methods of pylon classes may throw C++ exceptions. The pylon API throws exceptions that are subclasses of `GenICam::GenericException`. You should guard pylon calls with exception handlers catching `GenICam::GenericException`. Example:

```
try
{
  Camera.ExposureTimeRaw.SetValue( 42 );
}
catch ( GenICam::GenericException& e)
{
  cerr << "Failed to set the exposure time. Reason: "
  << e.GetDescription() << endl;
}
```

Since `GenICam::GenericException` inherits from `std::exception`, an alternative exception handler for exceptions of the `std::exception` type can be used:

```
try
{
  Camera.ExposureTimeRaw.SetValue( 42 );
}
catch ( std::exception& e)
{
  cerr << "Failed to set the exposure time. Reason: "
    << e.what() << endl;
}
```

1.3.3 Initialization/Uninitialization of the pylon Runtime Library

The pylon runtime system must be initialized before use. A pylon based application must call the `PylonInitialize()` method before using any functions of the pylon runtime system.

Before an application exits, it must call the `PylonTerminate()` method to free resources allocated by the pylon runtime system.

In pylon 2.0, the `Pylon::PylonAutoInitTerm` convenience class has been introduced. The constructor of `PylonAutoInitTerm` calls `PylonInitialize()`, the destructor calls `PylonTerminate()`. This ensures that the pylon runtime system is initialized during the lifetime of an object of the `PylonAutoInitTerm` type.

Example:

```
#include <pylon/PylonIncludes.h>
using namespace Pylon;

int main(int argc, char* argv[])
{

  Pylon::PylonAutoInitTerm autoInitTerm;  // PylonInitialize() will be called now

  // use %pylon
  // ..

}  // autoInitTerm's destructor calls PylonTerminate() now
```

When writing a MFC application, simply adding a member of type `PylonAutoInitTerm` to the application's `WinApp` class is sufficient to perform the initialization and cleanup of the pylon runtime system.

Alternatively, one could call `PylonInitialize()` in the MFC application's `InitInstance()` method, and `PylonTerminate()` in the application's `ExitInstance()` method.

Even when a MFC application calls `PylonTerminate()` before exiting, the MFC runtime system reports several pretended memory leaks for debug builds. This is due to the fact, that the memory tracking feature dumps the memory state before some of pylon's static objects are destroyed.

To suppress the dumping of memory leaks, issue a call to the MFC `AfxEnableMemoryTracking()` function:

```
// within the application's InitInstance() method
AfxEnableMemoryTracking ( false );
```

1.3.4 Enumerating and Creating Camera Objects

pylon offers two ways for enumerating and creating camera objects. The first approach uses the Transport Layer Factory to enumerate cameras across multiple transport layers. The second approach lets a Transport Layer object enumerate and create devices for a specific transport layer. Before describing the different enumeration schemes, the terms Device Class and Device Info object are introduced.

1.3.4.1 Device Classes

Each transport layer can create a specific type of camera objects, for example, the Pylon1394 transport layer will create camera objects representing IIDC 1394 cameras. Each type of device is associated with a unique identifier called Device Class.

1.3.4.2 Device Info Objects

The device enumeration procedure returns a list of Device Info objects. The base class for Device Info objects is `Pylon::CDeviceInfo`. A Device Info object uniquely describes a camera device. Device Info objects are used by a Transport Layer and the Transport Layer Factory to create camera objects representing the device described by the Device Info objects.

A `Pylon::CDeviceInfo` object stores a set of string properties. The data type of the values is `Pylon::String_t`. The following properties are available for all Device Info Objects:

FriendlyName	A human readable name for the device (e.g., the camera's model name). Friendly names are not unique.
FullName	A unique identifier for the device. No two devices will have the same full name.
VendorName	The name of the vendor
DeviceClass	Each transport layer can create a specific type (or class) of camera devices (e.g., IIDC 1394 or GigE Vision devices). The device types are identified by the Device Class property
SerialNumber	The device's serial number. The device's serial number availability during the enumeration process is not guaranteed, so the Serial Number Property may be undefined.
UserDefinedName	For some device classes, it is possible to assign a user defined name to a camera device. The value of this property is not necessarily unique.
DeviceFactory	The unique full name of the Transport Layer object that can create the device.

In addition, specific transport layers will require additional properties. These properties can be accessed in a generic way by using the `Pylon::IProperties` interface.

A more comfortable way is downcasting a Device Info object to the concrete class. This is illustrated in the following example, which prints out the IP address of a GigE Device Info object:

```
CBaslerGigECamera::DeviceInfo_t& GigEDeviceInfo =
  static_cast<CBaslerGigECamera::DeviceInfo_t&>(DeviceInfo);
 cout << GigEDeviceInfo.GetIpAddress() << endl;
```

1.3.4.3 Using the Transport Layer Factory for Enumerating cameras

The `Pylon::CTlFactory::EnumerateDevices()` method is used to retrieve a list of all available devices, regardless of which transport layer is used to access the device. The list contains Device Info objects that must be used for creating Camera objects.

The returned lists are of the `Pylon::DeviceInfoList_t` type and are used similarly to the STL `std::vector` class.

The following example prints out the unique names of all connected devices:

```
#include <pylon/PylonIncludes.h>
#include <ostream>
using namespace Pylon;
using namespace std;

int main()
{
  PylonAutoInitTerm autoInitTerm;

  CTlFactory& TlFactory = CTlFactory::GetInstance();
  DeviceInfoList_t lstDevices;
  TlFactory.EnumerateDevices( lstDevices );
  if ( ! lstDevices.empty() ) {
    DeviceInfoList_t::const_iterator it;
    for ( it = lstDevices.begin(); it != lstDevices.end(); ++it )
      cout << it->GetFullName();
  }
  else
    cerr << "No devices found!" << endl;

  return 0;
}
```

The Transport Layer Factory can now be used to create Camera objects. The following example illustrates how to create a Camera object for the first element in the device list:

```
  IPylonDevice *pDevice = TlFactory.CreateDevice( lstDevices[0] );
```

Attention:

Never call `free` or `delete` on pointers created by the Transport Layer Factory. Instead, use the `Pylon::CTlFactory::ReleaseTl()` method to free Transport Layer objects and the `Pylon::CTlFactory::DestroyDevice()` method to delete a `Pylon::IPylonDevice` pointer.

A list of all available transport layers can be retrieved by calling the `Pylon::CTlFactory::EnumerateTls()` method. This method fills a list with Transport Layer Info objects (`Pylon::CTlInfo`). The data structures are very similar to Device Info Objects. Transport Layer Info objects are used as arguments for the `Pylon::CTlFactory::CreateTl()` method that creates Transport Layer objects.

1.3.4.4 Using a Transport Layer Object for enumerating cameras

Transport Layer objects can be used to enumerate all devices accessible by a specific transport layer. Transport Layer objects are created by the Transport Layer Factory. This is shown in the following example, which creates a Transport Layer object for the Pylon1394 transport layer:

```
#include <pylon/PylonIncludes.h>
#include <pylon/1394/Basler1394Camera.h>
using namespace Pylon;

int main()
{
  PylonAutoInitTerm autoInitTerm;

  CTlFactory& TlFactory = CTlFactory::GetInstance();
  ITransportLayer* pTl
    = TlFactory.CreateTl( CBasler1394Camera::DeviceClass() );

  return 0;
}
```

As described above, Transport Layer objects can also be created by passing in a Transport Layer Info object.

The Transport Layer Object is now used for enumerating all the devices it can access:

```
    DeviceInfoList_t lstDevices;
    pTl->EnumerateDevices( lstDevices );
    if ( lstDevices.empty() ) {
        cerr <<  "No devices found" << endl;
        exit(1);
    }
```

`Pylon::ITransportLayer::EnumerateDevices` adds the discovered devices to the passed in Device Info List.

The Transport Layer object is now used for creating a Camera object. In the following example, a Camera Object for the first enumerated camera device is created:

```
    IPylonDevice* pDevice = pTl->CreateDevice( lstDevices[0] );
```

Attention:

Never call `free` or `delete` on pointers created by a Transport Layer object. Instead, use the `Pylon::ITransportLayer::DestroyDevice()` method to delete `Pylon::IPylonDevice` pointers created by a Transport Layer object.

1.3.5 Accessing Parameters

Before describing how to configure camera parameters (see the Configuring Cameras section), we provide some general information about how parameters are used in pylon.

pylon uses the GenApi reference implementation of GenICam for providing access to camera and other parameters. As mentioned in the GenApi Node Maps section, parameters are implemented as `GenApi::INode` objects. The nodes have different types, for example, there are nodes representing integer values and other nodes representing strings. For each type of parameter, there is an interface in GenApi. These interfaces are described in the Parameter Types section. The Access Modes for Parameters section introduces the concept of a parameter's access mode. An access mode is used to determine whether a parameter is available, readable, or writable.

1.3.5.1 Parameter Types

Integer Parameters The `GenApi::IInteger` interface is used to deal with integer parameters. An integer parameter is characterized by a minimum value, a maximum value, the current value, and an increment. All GenApi integer parameters are 64 bit signed integers. The following example prints all valid values for the `Image Width` parameter:

```
int64_t w = pCamera->Width.GetMin();
while ( w <= pCamera->Width.GetMax() )
{
    cout << w;
    w += pCamera->Width.GetInc();
}
```

There are two equivalent possibilities for setting a value using the `GenApi::IInteger` interface:

- Using the assignment operator, e.g.:

  ```
  pCamera->ExposureTimeRaw = 500;
  ```

- Using the `SetValue()` method, e.g.:

  ```
  pCamera->ExposureTimeRaw.SetValue( 500 );
  ```

There are also two equivalent ways to get a parameter's current value:

- Using the function call operator, e.g.:

  ```
  t = pCamera->ExposureTimeRaw();
  ```

- Using the GetValue() method, e.g.:

  ```
  t = pCamera->ExposureTime.GetValue();
  ```

Floating Point Parameters Floating point parameters are represented by `GenApi::IFloat` objects. GenApi uses the `double` data type for representing floating point numbers. The `GenApi::IFloat` interface is equivalent to the `GenApi::IInteger` interface.

Boolean Parameters The GenApi::IBoolean interface is used for parameters that can only have two values such as on/off or true/false.

As for the `GenApi::IInteger` interface, use the `GenApi::IBoolean::SetValue()` method or the assignment operator to set a Boolean parameter. For getting the value of a Boolean parameter, use either the `GenApi::IBoolean::GetValue()` method or the function call operator.

Enumeration Parameters The `GenApi::IEnumeration` interface is used to represent camera parameters for whom a fixed set of values is allowed. Features like Pixel Format or Testimage Type my serve as examples. In the Configuring Cameras section, there are examples showing how to handle enumerations.

Command Parameters Command parameters (`GenApi::ICommand`) are used for parameters which trigger an action or an operation inside of the camera, e.g. issuing a software trigger.

The action is issued by calling the `GenApi::ICommand::Execute()` method. The `GenApi::ICommand::IsDone()` method can be used to determine whether a running operation has finished.

String Parameters The `GenApi::IString` interface provides access to string parameters. The `GenICam::gcstring` class is used to represent strings. The `gcstring` class is a clone of the STL `std::string` class and can be used across dynamic link library (DLL) boundaries.

1.3.5.2 Access Modes for Parameters

Each parameter has an access mode that describes whether a feature is implemented, available, readable, and writable.

For a given camera, a feature may not be implemented at all, for example, a monochrome camera will not have a white balance feature.

Depending on the camera's state, a feature may be temporarily not available, for example, a parameter related to external triggering may not be available when the camera is in free run mode.

Available features can be read-only, write-only, or readable and writable.

The current state of a parameter can be queried by calling the parameter's `GetAccessMode()` method, which returns a `GenApi::EAccessMode` enumeration value described in the following table:

EAccessMode	Implemented	Available	Readable	Writable
NI	No	No	No	No
NA	Yes	No	No	No
WO	Yes	Yes	No	Yes
RO	Yes	Yes	Yes	No
RW	Yes	Yes	Yes	Yes

A parameter's access mode can change at runtime. For example, the `AOI width` and `AOI height` will be read-only when the camera is acquiring images. So checking a parameter's access mode before using it is recommended. Therefore, the `GenApi` namespace contains the following helper functions taking either an `EAccessMode` value, a pointer, or a reference to a parameter:

- `GenApi::IsImplemented`
- `GenApi::IsAvailable`
- `GenApi::IsReadable`
- `GenApi::IsWritable`

Example:

```
#include <pylon/gige/BaslerGigECamera.h>
using namespace GenApi;
using namespace Basler_GigECameraParams;
```

```
  ....
  // If the exposure time is writable, set the exposure time
  if ( GenApi::IsWritable ( Camera.ExposureTimeRaw ) )
    Camera.ExposureTimeRaw = 250;
  // If the mode is supported, set the camera to Mono16 mode
  if ( GenApi::IsAvailable( Camera.PixelFormat.GetEntryByName("Mono16") ) )
    Camera.PixelFormat.SetValue( PixelFormat_Mono16 );
  ....
```

1.3.6 Configuring Cameras

This section describes how a Camera Object is used to configure camera device parameters.

1.3.6.1 Programming Approaches: 'Generic' vs. 'Native'

The 'Native' Approach: Configuring Cameras Using a Camera Class In the previous section, we described how to create a Camera object using a Transport Layer object or the Transport Layer Factory. The `CreateDevice` methods return camera objects in the form of pointers to `Pylon::IPylonDevice` objects. The `Pylon::IPylonDevice` interface lets you deal with camera devices in a very generic way without knowing what kind of camera device is currently used.

Since the set of available camera parameters depends on the specific device type, the `Pylon::IPylonDevice` cannot provide named methods or members for setting and getting camera parameters. For comfortable access to the camera parameters, the pylon API provides Camera classes for the different camera types.

The following table shows the currently available classes:

Transport Layer	Name of Class	Kind of Devices
Pylon1394	`Pylon::CBasler1394Camera`	IIDC 1394 compliant cameras
PylonGigE	`Pylon::CBaslerGigECamera`	GigE Vision compliant camera

The Camera classes implement the methods of the `Pylon::IDevice` interface. In addition, a Camera class has a member for every camera parameter. The members are references (or handles) to the GenApi Nodes representing the parameters.

Camera classes behave like smart pointers. They take the ownership of a `Pylon::IPylonDevice` pointer returned by the `CreateDevice()` methods. When a Camera class is attached to a `Pylon::IPylonDevice` pointer, this pointer must not be freed by calling one of the `DestroyDevice()` methods. The Camera class is responsible for freeing the pointer.

The following examples shows how to attach a `Pylon::IPylonDevice` pointer to a reference to a `Pylon::CBasler1394Camera` object:

```
#include <pylon/PylonIncludes.h>
#include <pylon/1394/Basler1394Camera.h>
using namespace Pylon;

{
  ....
  // create device
  IPylonDevice *pDevice = TlFactory.CreateDevice( lstDevices[0] );

  // create a camera class and assign the device pointer to it
  CBasler1394Camera Camera = pDevice;
```

```
  // use the camera
  .....
} // Camera's destructor now frees the \c IPylonDevice pointer
```

Creating a camera class and attaching a `Pylon::IPylonDevice` pointer must frequently be performed at different places in your code. In these cases, use the Camera class' `Attach()` method as demonstrated in the following example.

```
#include <pylon/PylonIncludes.h>
#include <pylon/1394/Basler1394Camera.h>
#include <ostream>
using namespace Pylon;
using namespace std;

class BaslerDevice
{
public:
   // creates and opens a camera object for the first availble
   // 1394 camera
   bool Init()
   {
      CTlFactory& TlFactory = CTlFactory::GetInstance();
      ITransportLayer* pTl = TlFactory.CreateTl( CBasler1394Camera::DeviceClass() );
      DeviceInfoList_t lstDevices;
      pTl->EnumerateDevices( lstDevices );
      if ( lstDevices.empty() ) {
        cerr <<  "No devices found" << endl;
        return false;
      }

      m_Camera.Attach( TlFactory.CreateDevice( lstDevices[0] ) );
   }

private:
  // takes care about initialization and cleanup of %pylon runtime system
  PylonAutoInitTerm m_AutoInitTerm;
  // the Camera class for a 1394 camera
  CBasler1394Camera m_Camera;
};

int main()
{
   BaslerDevice device;
   if ( ! device.Init() )
     return 1;  // no device found
   // use device
   //...

   return 0;
}
```

In this example, the destrucor of the `Basler` Device class calls the destructors of `m_Camera` and `m_Auto-`

`InitTerm`. The destructor of `m_Camera` frees the `IPylonDevice` pointer. The destructor of `m_AutoInit-Term` calls the `Pylon::PylonTerminate()` method to perform all necessary cleanup.

It is possible to attach a Camera class to a `IPylonDevice` pointer without letting the Camera class take the ownership of the `IPylonDevice` pointer. In that case, the `IPylonDevice` pointer must be explicitely freed by calling the Transport Layer's or the Transport Layer Factory's `DestroyDevice()` methods. Example:

```
bool use1394Device( IPylonDevice* pDevice )
{
  if ( pDevice->GetDeviceInfo().GetDeviceClass()
        != CBasler1394Camera::DeviceClass() )
  {
    cerr << "The camera isn't a 1394 camera!" << endl;
    return false;
  }
  CBasler1394Camera Camera;
  Camera.Attach( pDevice, false ); // doesn't take over ownership

  // use Camera

  return true;
}

int main()
{
   CTlFactory& TlFactory = CTlFactory::GetInstance();
   ITransportLayer* pTl = TlFactory.CreateTl( CBasler1394Camera::DeviceClass() );
   DeviceInfoList_t lstDevices;
   pTl->EnumerateDevices( lstDevices );
   if ( lstDevices.empty() ) {
     cerr <<  "No devices found" << endl;
     return 1;
   }
   IPylonDevice *pDevice = pTl->CreateDevice( lstDevices[0] );

   use1394Device( pDevice );

   if ( pDevice->IsOpen() )
     pDevice->Close();

   pTl->DestroyDevice( pDevice );

   return 1;
}
```

A Camera Class provides access to its parameters through its members, for example, the `CBasler1394Camera::ExposureTimeRaw` member is used to set the exposure time to the minimum allowed value:

```
if ( GenApi::IsAvailable( Camera.ExposureTimeRaw ) )
  Camera.ExposureTimeRaw.SetValue( Camera.ExposureTimeRaw.GetMin() );
```

With the exception of enumeration values, all camera parameters representing members of the Camera

are references to GenApi::IBoolean, GenApi::IInteger, GenApi::IFloat, ... parameter objects. The different types of parameters and how they are used is described in the Parameter Types section.

Enumeration parameters are represented by references to instances of GenApi::IEnumerationT <EnumT> class templates, where EnumT is a C/C++ enum. For each enumeration parameter, a corresponding enum type is declared. The IEnumerationT::SetValue() method and the assignment operator of IEnumerationT take these enum values as arguments. The IEnumerationT::GetValue() method and the function call operator for IEnumerationT return enum values. The enumerations are defined within the C++ namespaces listed in the following table:

Class Name	Namespace of Enums
Pylon::CBasler1394Camera	Basler_IIDC1394CameraParams
Pylon::CBaslerGigECamera	Basler_GigECameraParams

The following example demonstrates the use of enumeration parameters:

```
#include <pylon/PylonIncludes.h>
#include <pylon/gige/BaslerGigeCamera.h>
#include <ostream>

using namespace Basler_GigECameraParams;
using namespace Pylon;
using namespace std;

int main()
{

  Pylon::PylonAutoInitTerm autoInitTerm;

  // create and open a camera object
  typedef CBaslerGigECamera Camera_t;
  CTlFactory& TlFactory = CTlFactory::GetInstance();
  ITransportLayer *pTl = TlFactory.CreateTl( Camera_t::DeviceClass() );
  DeviceInfoList_t devices;
  if ( 0 == pTl->EnumerateDevices( devices ) )
  {
    cerr << "No camera present!" << endl;
    return 1;
  }
  Camera_t Camera = pTl->CreateDevice( devices[ 0 ] );
  Camera.Open();

  // set and get values of enumeration parameters
  Camera.TriggerMode.SetValue( TriggerMode_Off );
  Camera.AcquisitionMode.SetValue( AcquisitionMode_SingleFrame );
  Camera.ExposureMode.SetValue( ExposureMode_Timed );
  PixelFormatEnums Format = Camera.PixelFormat.GetValue();

  Camera.Close();

  return 0;
}
```

The 'Generic' Approach: Configuring Cameras Using Node Maps The Node Maps introduced in the GenApi Node Maps section provide a completely generic way of accessing parameters. This generic method is especially useful when writing code that must support different camera interface types.

The following example illustrates how to set the exposure time and acquisition mode:

```
#include <pylon/PylonIncludes.h>
#include <ostream>
using namespace GenApi;
using namespace Pylon;
using namespace std;


int main(int argc, char* argv[])
{
  PylonAutoInitTerm autoInitTerm;

  // create camera object
  DeviceInfoList_t deviceList;
  CTlFactory::GetInstance().EnumerateDevices( deviceList );
  IPylonDevice *pCamera =
    CTlFactory::GetInstance().CreateDevice( deviceList[0] );

  // configure device
  pCamera->Open();
  INodeMap* pNodeMap = pCamera->GetNodeMap();
  CIntegerPtr ptrExposureTime = pNodeMap->GetNode("ExposureTimeRaw");
  if ( ! ptrExposureTime ) {
    cerr << "There is no exposure time parameter" << endl;
    exit( 1 );
  }
  if ( ! IsWritable( ptrExposureTime ) ) {
    cerr << "The exposure time parameter is not writable" << endl;
    exit( 1 );
  }
  ptrExposureTime->SetValue( ptrExposureTime->GetMin() );

  CEnumerationPtr ptrAcqMode = pNodeMap->GetNode("AcquisitionMode");
  if ( ! ptrAcqMode ) {
    cerr << "There is no acquisition mode parameter" << endl;
    exit( 1 );
  }
  if ( ! IsWritable( ptrAcqMode ) ) {
    cerr << "The acquisition mode parameter is not writable" << endl;
    exit( 1 );
  }

  ptrAcqMode->FromString("SingleFrame");

  pCamera->Close();

  return 0;
}
```

In the previous example, the GenApi::CIntegerPtr and GenApi::CEnumerationPtr smart pointer classes are used. These helper classes are wrappers for GenApi interface pointers and take the burden from the user for casting GenApi::INode to the desired interface type. There are also smart pointers for other GenApi interfaces such as GenApi::CBooleanPtr, GenApi::CFloatPtr, GenApi::CStringPtr, ...

As another example, the following code snippet prints a list of a camera's implemented parameters. The example uses GenApi::ICategory nodes, used to build hierarchies of parameters. The special category named "Root" is the top-level category used to navigate through the hierarchy of categories and features.

```
#include <pylon/PylonIncludes.h>
#include <ostream>
using namespace GenApi;
using namespace Pylon;
using namespace std;


void PrintNames(const GenApi::CNodePtr& node )
{
  CCategoryPtr ptrCategory ( node );
  // if node is a category, visit all children,
  // else print the name of the node
  if ( ptrCategory ) {
    FeatureList_t features;
    ptrCategory->GetFeatures( features );
    FeatureList_t::const_iterator it;
    for ( it = features.begin(); it != features.end(); ++it )
      PrintNames( *it );
  }
  else
  {
  // if the feature is implemented, print name of node
    if ( GenApi::IsImplemented( node ) )
      cout << node->GetName() << endl;
  }
}

int main(int argc, char* argv[])
{
  PylonAutoInitTerm autoInitTerm;

  // create camera object
  DeviceInfoList_t deviceList;
  CTlFactory::GetInstance().EnumerateDevices( deviceList );
  IPylonDevice *pCamera =
    CTlFactory::GetInstance().CreateDevice( deviceList[0] );
  pCamera->Open();

  // print the parameter names
  INodeMap* pNodeMap = pCamera->GetNodeMap();
  INode* pNode = pNodeMap->GetNode( "Root" );
  PrintNames ( pNode );

  pCamera->Close();
```

```
  return 0;
}
```

1.3.6.2 Opening and Closing a Camera

Before reading or writing parameters, the involved drivers must be initialized and a connection to the physical camera device must be established. This is done by the `IPylonDevice::Open()` method.

To release the connection to a device and to free all related resources, call the `IPylonDevice::Close()` method.

1.3.6.3 Transport Layer Parameters

The `IPylonDevice::GetTLNodeMap()` method returns a pointer to the node map containing transport layer related parameters. For the PylonGigE transport layer, e.g. timeouts for read requests and write requests can be configured for each device.

If there are no transport layer parameters for a transport layer, the `GetTLNodeMap()` method returns a NULL pointer. Currently, there are no transport layer parameters for FireWire cameras.

To provide easy access to the parameters, there are classes that can be used to access the parameters. The camera classes declare a typedef for the transport layer classes.

The following example illustrates how to configure the heart beat timeout for a GigE camera:

```
#include <pylon/PylonIncludes.h>
#include <pylon/gige/BaslerGigeCamera.h>
#include <ostream>

using namespace Basler_GigETLParams;
using namespace Pylon;
using namespace GenApi;
using namespace std;

int main()
{
  PylonAutoInitTerm autoInitTerm;

  typedef CBaslerGigECamera Camera_t;
  CTlFactory& TlFactory = CTlFactory::GetInstance();
  ITransportLayer *pTl = TlFactory.CreateTl( Camera_t::DeviceClass() );
  DeviceInfoList_t devices;
  if ( 0 == pTl->EnumerateDevices( devices ) )
  {
    cerr << "No camera present!" << endl;
    return 1;
  }

  Camera_t Camera = pTl->CreateDevice( devices[ 0 ] );
  Camera.Open();
  INodeMap* pTlNodeMap = Camera.GetTLNodeMap();
  if ( NULL == pTlNodeMap )
  {
```

```
    cerr << "This camera class doesn't provide a transport layer nodemap" << endl;
    return 1;
  }
  Camera_t::TlParams_t TlParams = pTlNodeMap;

  TlParams.HeartbeatTimeout.SetValue(5000); // 5 sec timeout

  return 0;
}
```

1.3.6.4 Getting Informed About Parameter Changes

The pylon API provides the functionality for installing callback functions that will be called when a parameter's value or state (e.g., the access mode or value range) may have been changed. It is possible to either install a C function or a C++ class member function as a callback.

Each callback is installed for a specific parametre. If the parameter itself has been touched or if another parameter that can influence the state of the parameter has been changed, the callback will be fired.

The following example demonstrates how to install callbacks for the `ExposureTimeRaw` parameter:

```
#include <pylon/PylonIncludes.h>
#include <pylon/gige/BaslerGigECamera.h>
#include <ostream>
using namespace Pylon;
using namespace std;

 C callback function
void f(GenApi::INode* pNode )
{
    cout << "Perhaps the value or state of " << pNode->GetName() << "has changed." << endl;
    if ( GenApi::IsReadable( pNode ) )
    {
      GenApi::CValuePtr ptrValue( pNode );
      cout << "The current value is " << ptrValue->ToString() << endl;
    }
}

class C
{
public:
  // member function as callback function
  void f(GenApi::INode* pNode )
  {
    cout << "Perhaps the value or state of " << pNode->GetName() << "has changed." << endl;
    if ( GenApi::IsReadable( pNode ) )
    {
      GenApi::CValuePtr ptrValue( pNode );
      cout << "The current value is " << ptrValue->ToString() << endl;
    }
  }
};

int main()
```

```
{
  PylonAutoInitTerm autoInitTerm;

  C c;  // c.f() will be installed as callback

  // create and open a camera object
  typedef CBaslerGigECamera Camera_t;
  CTlFactory& TlFactory = CTlFactory::GetInstance();
  ITransportLayer *pTl = TlFactory.CreateTl( Camera_t::DeviceClass() );
  DeviceInfoList_t devices;
  if ( 0 == pTl->EnumerateDevices( devices ) )
  {
    cerr << "No camera present!" << endl;
    return 1;
  }
  Camera_t Camera = pTl->CreateDevice( devices[ 0 ] );
  Camera.Open();

  // install the C function as callback
  GenApi::CallbackHandleType h1 =
    GenApi::Register( Camera.ExposureTimeRaw.GetNode(), &f );

  // install a member function as callback
  GenApi::CallbackHandleType h2 =
    GenApi::Register( Camera.ExposureTimeRaw.GetNode(), c, &C::f );

  // this will trigger the callback functions
  Camera.ExposureTimeRaw.SetValue( 128 );

  // uninstall the callback functions
  Camera.ExposureTimeRaw.GetNode()->DeregisterCallback(h1);
  Camera.ExposureTimeRaw.GetNode()->DeregisterCallback(h2);

  // close the camera object
  Camera.Close();

}
```

1.3.7 Grabbing Images

1.3.7.1 Terminology

Acquire, Transfer, and Grab Images In this document we distinguish between image acquisition, image data transfer, and image grabbing.

We denote the processes inside the camera as image acquisition. When a camera starts image acquisition, the sensor is exposed. When exposure is complete, the image data is read out from the sensor.

The acquired image data is transferred from the camera's memory to the PC using an interface such as FireWire or Gigabit Ethernet.

The process of writing the image data into the PC's main memory is referred to as "grabbing" an image.

Image Data Stream and Stream Grabber A camera may provide different sources for image data, where each source can deliver a stream of image data. In pylon, so called Stream Grabber objects are responsible for managing the process of grabbing data from a stream, i.e., writing the data into the PC's main memory.

A Stream Grabber only grabs images from a single data stream. To grab data from multiple streams, several Stream Grabbers are needed.

1.3.7.2 Using Stream Grabbers

The following sections describe the use of Stream Grabber objects. The order of this section reflects the sequence in which a typical grab application will use a Stream Grabber object.

Getting a Stream Grabber Stream Grabber objects are managed by Camera objects. The `IPylonDevice::GetStreamGrabber()` method returns a pointer to the `Pylon::IStreamGrabber` object. Before retrieving a Stream Grabber object, the Camera object must be opened. The Stream Grabber object itself also must be opened before it is used.

Example:

```
Camera.Open();
IStreamGrabber* pGrabber = Camera.GetStreamGrabber(0);
pGrabber->Open();
```

Attention:

Never try to call `delete` or `free` on a stream grabber pointer retrieved from a Camera object. The Camera object retains the ownership of a Stream Grabber object and manages its lifetime.

The number of available stream grabbers can be determined with the `IPylonDevice::GetNumStreamGrabberChannels()` method:

```
const int numGrabbers = Camera.GetNumStreamGrabberChannels();
```

Each camera provides at least one stream grabber. This grabber has the index 0.

Configuring a Stream Grabber Independent of the transport layer used, each stream grabber provides two mandatory parameters:

- MaxBufferSize - Maximum size of a buffer used for grabbing images in bytes
- MaxNumBuffer - Maximum number of buffers used for grabbing images

A grab application **must** set the above two parameters before grabbing starts.

Depending on the transport layer, a Stream Grabber provides further parameters such as streaming related timeouts. All of these parameters are set to default values and image grabbing can be performed without tweaking the defaults.

There are two ways for accessing a Stream Grabber's parameters:

The most comfortable way is to use a concrete class for a Stream Grabber object. Each Camera class provides a `typedef` for the corresponding Stream Grabber class. A Stream Grabber class takes the ownership of a `IStreamGrabber` pointer returned by the `GetStreamGrabber()` method. The Stream Grabber class has members to access the Stream Grabber object's parameters.

Example:

```
Camera.Open();
IStreamGrabber* pGrabber = Camera.GetStreamGrabber(0);
CBaslerGigECamera::StreamGrabber_t StreamGrabber = pGrabber;
// First open the stream grabber
StreamGrabber.Open();
// Set the maximum buffer size according to the amount of data
// the camera will send
StreamGrabber.MaxBufferSize = Camera.PayloadSize();
// We are going to use 10 buffers
StreamGrabber.MaxNumBuffer = 10;
```

When using the generic programming approach, i.e., using the `Pylon::IPylonDevice` and `Pylon::IStreamGrabber` interfaces instead of Camera and Stream Grabber classes, the `IStreamGrabber::GetNodeMap()` method must be used to retrieve the GenApi node map holding the stream grabber's parameters.

Stream Grabber node maps are used in the same way as node maps for Camera objects. The use of node maps for Camera objects is described in the The 'Generic' Approach: Configuring Cameras Using Node Maps section.

Preparing a Stream Grabber for Grabbing Different system resources are required depending on the transport layer used for grabbing images, for example:

- DMA resources
- Memory for the driver's data structures
- Isochronous channel for Firewire cameras
- Isochronous bandwidth for Firewire cameras

The Stream Grabber's `PrepareGrab()` method is used to allocate the needed resources.

In addition to resource allocation, the `PrepareGrab()` call causes the camera object to perform a change in state. Typically, the camera parameters controlling the image size (AOI, pixel format, binning, etc.) will be read-only after `PrepareGrab()` has been called. These parameters must be set up before calling `PrepareGrab()` and must not be changed while image grabbing is active.

Providing Memory for Grabbing All pylon transport layers can grab image data into memory buffers allocated by a user application. The user allocated memory buffers must be registered at the Stream Grabber object. The registration step is needed for performance reasons, allowing the Stream Grabber to prepare and cache internal data structures used for dealing with user provided memory.

The buffer registration returns handles to the registered buffers, which are used in the steps following the buffer registration.

Example:

```
StreamGrabber.Open();
const int bufferSize = (int) Camera.PayloadSize();
const int numBuffers = 10;
unsigned char* ppBuffers[numBuffers];
StreamBufferHandle handles[numBuffers];
```

```
StreamGrabber.MaxBufferSize = bufferSize;
StreamGrabber.MaxNumBuffer = numBuffers;

StreamGrabber.PrepareGrab();
for ( int i = 0; i < numBuffers; ++i )
{
  ppBuffers[i] = new unsigned char[bufferSize];
  handles[i] = StreamGrabber.RegisterBuffer( ppBuffers[i], bufferSize);
}
```

The buffer registration mechanism restricts the ownership of the buffers. Although the content of registered buffers can be changed by the user application, the application must not delete the memory of buffers that are registered. Freeing the memory is not allowed until the buffers are deregistered by using `IStreamGrabber::DeregisterBuffer()` .

Feeding the Stream Grabber's Input Queue Each Stream Grabber maintains two different buffer queues. The buffers to be filled must be fed into the Grabber's input queue. Grabbed buffers can be retrieved from the Grabber's output queue.

The `IStreamGrabber::QueueBuffer()` method is used to put a buffer into the grabber's input queue. The `QueueBuffer()` method accepts two parameters, a buffer handle and an optional, user provided pointer to user provided context information. Together with the buffer, the context pointer is passed back to the user when retrieving the grabbed buffer from the grabber's output queue. A Stream Grabber never changes the memory that the context pointer is pointing to.

Example:

```
MyContext context[numBuffers];
for ( int i = 0; i < numBuffers; ++i )
{
  // Enqueue image buffers and use the buffer's index as context
  // information
  StreamGrabber.QueueBuffer( handles[i], & context[i] );
}
```

Attention:

A stream grabber temporarily takes ownership of an enqueued buffer. Never try to modify or delete a buffer when it is placed into the stream grabber's input queue.

Queueing images into the SreamGrabber's input queue doesn't start image acquisition inside the camera! After queueing the buffers, the Stream Grabber is prepared to grab data from the camera into the queued buffers. Image acquisition must be explicitly started.

Starting and Stopping Image Acquisition To start image acquisition, use the Camera object's `AcquisitionStart` parameter. `AcquisitionStart` is a command parameter, i.e., calling the `Execute()` method of the `AcquisitionStart` parameter sends an acquisition start command to the camera.

A camera device typically provides two acquisition modes:

- Single Frame mode where the camera acquires one image.
- Continuous mode where the camera continuously acquires and transfers images until acquisition is stopped.

To be precise, the acquisition start command doesn't necessarily start immediate acquisition inside of the camera. When either external triggering or software triggering is enabled, the acquisition start command prepares the camera to acquire images. Actual acquisition starts when the camera senses an external trigger signal or receives a software trigger command.

When the camera's continuous acquisition mode is enabled, the `AcquisitionStop` parameter is used to stop image acquisition.

Normally, a camera starts transferring the image as soon as possible after starting the acquisition, no special command to start the image transfer is needed.

Example:

```
using namespace Basler_GigECameraParams;
Camera.AcquisitionMode.SetValue( AcquisitionMode_Continuous );
Camera.AcquisitionStart.Execute();
```

Retrieving Grabbed Images The transferred image data is written to the buffer(s) in the stream grabber's input queue. When a buffer is filled with grabbed image data, the stream grabber places it into its output queue, from which it can be retrieved by the user application.

There is a wait object associated with the Stream Grabber's output queue. This wait object allows the application to wait until either a grabbed image arrives at the output queue or a timeout expires.

When the wait operation successfully returns, the grabbed buffer can be returned with the Stream Grabber object's `RetrieveResult()` method. The `RetrieveResult()` method fills a `Pylon::GrabResult` object, containing among other things, the following information:

- Status of the grab (succeeded, canceled, failed)
- The buffer's handle
- The pointer to the buffer
- The user provided context pointer
- AOI and image format
- Error number and error description if the grab has failed

When getting a buffer from the grabber's output queue, the ownership of the buffer is given over to the application. A buffer retrieved from the output queue will never be overwritten until it is again placed into the grabber's input queue.

Remember, a buffer retrieved from the output queue must be deregistered before its memory can be freed.

We recommended using the buffer handle from the Grab Result object to requeue a buffer into the grabber's input queue.

When the camera doesn't send data, a buffer remains in the grabber's input queue until the Stream Grabber object's `CancelGrab()` method is called. The `CancelGrab()` puts all buffers from the input queue to the output queue, including any buffer currently being filled. By checking the status of a Grab Result object, you can determine whether a buffer has been canceled.

The following example shows a typical grab loop:

```
const int numGrabs = 100;
GrabResult Result;
```

```
for ( int i = 0; i < numGrabs; ++i )
{
  // Wait for the grabbed image with a timeout of 3 seconds
  if ( StreamGrabber.GetWaitObject().Wait( 3000 ))
  {
     // Get an item from the grabber's output queue
     if ( ! StreamGrabber.RetrieveResult( Result ) )
     {
       cerr << "Failed to retrieve an item from the output queue" << endl;
       break;
     }

     if ( Result.Succeeded() )
     {
       // Grabbing was successful. Process the image.
       ProcessImage( (unsigned char*) Result.Buffer() );
     }
     else
     {
       cerr << "Grab failed: " << Result.GetErrorDescription() << endl;
       break;
     }
     // Requeue the buffer
     if ( i + numBuffers < numGrabs )
       StreamGrabber.QueueBuffer( Result.Handle(), Result.Context() );
  }
  else
  {
    cerr << "timeout occurred when waiting for a grabbed image" << endl;
    break;
  }
}
```

Finish Grabbing If the camera is set for continuous acquisition mode, acquisition should first be stopped:

```
Camera.AcquisitionStop.Execute();
```

If you are not sure that the grabber's input queue **really** is empty, the Stream Grabber object's CancelGrab() method should be issued to flush the input queue. The canceled buffers are now available at the grabber's output queue.

An application should retrieve all items from the grabber's output queue before closing a Stream Grabber object.

Before freeing their memory, deregister the buffers.

When all buffers are deregistered, call the Stream Grabber object's `FinishGrab()` method to release all resources related to grabbing. `FinishGrab()` must not be called when there are still buffers in the grabber's input queue!

When grabbing has been finished, a Stream Grabber object should be closed.

Example:

```
// The camera is in continuous mode, stop the image acquisition
```

```
  Camera.AcquisitionStop.Execute();
  // Flush the input queue
  StreamGrabber.CancelGrab();
  // Consume all items from the output queue
  while ( StreamGrabber.GetWaitObject().Wait(0) )
  {
    if ( ! StreamGrabber.RetrieveResult( Result ) )
      cerr << "Faile to retrieve item from output queueu" << endl;
    else if ( Result.Status() == Canceled )
    {
      cout << "Got canceled buffer" << endl;
    }
  }
  for ( int i = 0; i < numBuffers; ++i )
  {
    StreamGrabber.DeregisterBuffer(handles[i]);
    delete [] ppBuffers[i];
  }
  StreamGrabber.FinishGrab();
  StreamGrabber.Close();
```

1.3.7.3 Complete Sample program

Here is the complete sample program for acquiring images from a GigE camera in continuous mode. For grabbing images from a FireWire camera, only a few lines of code must be changed. This is shown in the sample programs installed as part of the pylon SDK. These sample programs are installed at <SDK ROOT>\samples.

```
#include <pylon/PylonIncludes.h>
#include <pylon/gige/BaslerGigECamera.h>
#include <ostream>
using namespace Pylon;
using namespace Basler_GigECameraParams;
using namespace std;

typedef CBaslerGigECamera Camera_t;

void ProcessImage( unsigned char* pImage, int imageSizeX, int imageSizeY )
{
  // Do something with the image data
}

struct MyContext
{
  // Define some application specific context information here
};

int main()
{
  PylonAutoInitTerm autoInitTerm;

  try
  {
```

```
// Enumerate GigE cameras
CTlFactory& TlFactory = CTlFactory::GetInstance();
ITransportLayer *pTl = TlFactory.CreateTl( Camera_t::DeviceClass() );
DeviceInfoList_t devices;
if ( 0 == pTl->EnumerateDevices( devices ) )
{
  cerr << "No camera present!" << endl;
  return 1;
}
// Create a camera object
Camera_t Camera = pTl->CreateDevice( devices[ 0 ] ) ;

// Open the camera object
Camera.Open();

// Parameterize the camera

// Mono8 pixel format
Camera.PixelFormat.SetValue( PixelFormat_Mono8 );

// Maximized AOI
Camera.OffsetX.SetValue( 0 );
Camera.OffsetY.SetValue( 0 );
Camera.Width.SetValue( Camera.Width.GetMax() );
Camera.Height.SetValue( Camera.Height.GetMax() );

// Continuous mode, no external trigger used
Camera.TriggerSelector.SetValue( TriggerSelector_AcquisitionStart );
Camera.TriggerMode.SetValue( TriggerMode_Off );
Camera.AcquisitionMode.SetValue( AcquisitionMode_Continuous );

// Configure exposure time and mode
Camera.ExposureMode.SetValue( ExposureMode_Timed );
Camera.ExposureTimeRaw.SetValue( 100 );

// Get and open a stream grabber
IStreamGrabber* pGrabber = Camera.GetStreamGrabber(0);
CBaslerGigECamera::StreamGrabber_t StreamGrabber =
Camera.GetStreamGrabber(0);
StreamGrabber.Open();

// Parameterize the stream grabber
const int bufferSize = (int) Camera.PayloadSize();
const int numBuffers = 10;
StreamGrabber.MaxBufferSize = bufferSize;
StreamGrabber.MaxNumBuffer = numBuffers;
StreamGrabber.PrepareGrab();

// Allocate and register image buffers, put them into the
// grabber’s input queue
unsigned char* ppBuffers[numBuffers];
MyContext context[numBuffers];
StreamBufferHandle handles[numBuffers];
for ( int i = 0; i < numBuffers; ++i )
```

```
{
  ppBuffers[i] = new unsigned char[bufferSize];
  handles[i] = StreamGrabber.RegisterBuffer( ppBuffers[i], bufferSize);
  StreamGrabber.QueueBuffer( handles[i], &context[i] );
}

// Start image acquisition
Camera.AcquisitionStart.Execute();

// Grab and process 100 images
const int numGrabs = 100;
GrabResult Result;
for ( int i = 0; i < numGrabs; ++i )
{
  // Wait for the grabbed image with a timeout of 3 seconds
  if ( StreamGrabber.GetWaitObject().Wait( 3000 ))
  {
    // Get an item from the grabber’s output queue
   if ( ! StreamGrabber.RetrieveResult( Result ) )
   {
     cerr << "Failed to retrieve an item from the output queue" << endl;
     break;
   }
    if ( Result.Succeeded() )
    {
      // Grabbing was successful. Process the image.
      ProcessImage( (unsigned char*) Result.Buffer(), Result.GetSizeX(), Result.GetSizeY() );
    }
    else
    {
      cerr << "Grab failed: " << Result.GetErrorDescription() << endl;
      break;
    }
    // Requeue the buffer
    if ( i + numBuffers < numGrabs )
      StreamGrabber.QueueBuffer( Result.Handle(), Result.Context() );

  }
  else
  {
    cerr << "timeout occurred when waiting for a grabbed image"
         << endl;
    break;
  }
}

// Finished. Stop grabbing and do clean-up

// The camera is in continuous mode, stop image acquisition
Camera.AcquisitionStop.Execute();

// Flush the input queue, grabbing may have failed
StreamGrabber.CancelGrab();
```

```
    // Consume all items from the output queue
    while ( StreamGrabber.GetWaitObject().Wait(0) )
    {
      StreamGrabber.RetrieveResult( Result );
      if ( Result.Status() == Canceled )
      {
        cout << "Got canceled buffer" << endl;
      }
    }

    // Deregister and free buffers
    for ( int i = 0; i < numBuffers; ++i )
    {
      StreamGrabber.DeregisterBuffer(handles[i]);
      delete [] ppBuffers[i];
    }

    // Clean up
    StreamGrabber.FinishGrab();
    StreamGrabber.Close();
    Camera.Close();
    TlFactory.ReleaseTl( pTl );
  }
  catch( GenICam::GenericException &e )
  {
    // Error handling
    cerr << "An exception occurred!" << endl << e.GetDescription() << endl;
    return 1;
  }
  // Quit application
  return 0;
}
```

1.3.7.4 Waiting for Multiple Events

In applications, a separate thread is often dedicated to grabbing images. Typically, this grab thread must be synchronized with other threads of the application. For example, an application may want to signal the grab thread to terminate.

Synchronization can be realized by using Wait Objects. The concept of 'Wait Objects' introduced in the Retrieving Grabbed Images section allows not only waiting until a grabbed buffer is available, but also getting informed about other events.

Wait Objects are an abstraction of operating system specific objects that can be either signalled or non-signalled. Wait Objects provide a wait operation that blocks until the Wait Object is signalled.

For the Windows operating system, the Pylon::AlertableWaitObject class is a wrapper for native Win32 objects. The `AlertableWaitObject` class provides two constructors:

- One creates an `AlertableWaitObject` by passing in a handle for an already existing Win32 object
- The other allows creation of `AlertableWaitObjects` that wrap their own Win32 events.

```
using namespace Pylon;
```

```
// Create a wait object wrapping its own handle
AlertableWaitObject w0;

// Create or retrieve a handle for an Win32 object whose handle can be used
// to wait for, e.g., an event, mutex, thread, waitable timer
HANDLE h = CreateOrGetAHandle();
// Create a wait object from a given handle
AlertableWaitObject w1(h);
```

The Pylon::WaitObjects class is a container for Wait Objects and provides two methods of waiting for Wait Objects stored in the container:

- The Pylon::WaitObjects::WaitForAny() method returns when at least one object in the container is signalled.
- The Pylon::WaitObjects::WaitForAll() method returns when all objects in the container are signalled.

```
// Create a container and insert two wait objects
WaitObjects waitObjects;
waitObjects.Add(w0);
waitObjects.Add(w1);

// Wait for three seconds until any of the wait objects get signalled
unsigned int index;
if ( waitObjects.WaitForAny( 3000, &index) )
{
    cout << "WaitObject w" << index << " has been signalled" << endl;
}
else
{
    cout << "Timeout occurred when waiting for wait objects" << endl;
}

// Wait for three seconds until all of the wait objects are signalled
if ( waitObjects.WaitForAll(3000) )
{
    cout << "All wait objects are signalled" << endl;
}
else
{
    cout << "Timeout occurred when waiting for wait objects" << endl;
}
```

The AlertableWaitObject::Signal() method is used to signal a wait object. The AlertableWaitObject::Reset() can be used to put the Wait Object into the non-signalled state.

```
// Put w0 into the signalled state
w0.Signal()

// Put w0 into the non-signalled state
w0.Reset()
```

The following code snippets illustrate how a grab thread uses the WaitForAny() method to simultaneously wait for buffers and a termination request. The snippets are taken from the 'WaitForMultiple' sample program installed as part of the pylon SDK. The sample programs are installed at <SDK ROOT>\samples.

After preparing for grabbing, the application's main thread starts the grab thread and sleeps for 5 seconds.

```
// Start the grab thread. The grab thread starts the image acquisition
// and grabs images
cout << "Going to start the grab thread" << endl;
StartThread();

// Let the thread grab images for 5 seconds
Sleep(5000);
```

The grab thread sets up a Wait Object container holding the StreamGrabber's Wait Object and a Pylon::AlertableWaitObject. The latter is used by the main thread to request the termination of grabbing:

```
// create and prepare the wait object container
WaitObjects waitObjects;

waitObjects.Add( m_pStreamGrabber->GetWaitObject() );  // getting informed about buffers
waitObjects.Add( m_TerminationEvent ); // getting informed about termination request
```

Then the grab thread enters an infinite loop that starts waiting for any of the Wait Objects:

```
GrabResult result;   // grab result
bool terminate = false;
while ( ! terminate )
{
    if ( ! waitObjects.WaitForAny( INFINITE, &index ) )
    {
        // timeout occurred, should never happen when using INFINITE
        cerr << "Timeout occurred????" << endl;
        break;
    }
```

When the WaitForAny() method returns with true, the value of `index` is used to determine whether a buffer has been grabbed or a request to terminate grabbing is pending:

```
    switch ( index )
    {
    case 0:  // a grabbed buffer is available
        if ( m_pStreamGrabber->RetrieveResult( result ) )
        {
            if ( result.Succeeded() )
            {
                cout << "Successfully grabbed image " << ++nSucc << endl;
                unsigned char* pPixel = (unsigned char*) result.Buffer();
                // process buffer .....
            }

            // reuse the buffer for grabbing
            m_pStreamGrabber->QueueBuffer(result.Handle(), result.Context() );
        }
        else
        {
```

```
            cerr << "Failed to retrieve result" << endl;
            terminate = true;
        }
        break;
    case 1:  // received a termination request
        terminate = true;
        break;
    }
```

The main thread signals the grab thread to terminate by calling the `AlertableWaitObject's` Signal() method:

```
// Signal the thread to terminate
cout << "Going to issue termination request" << endl;
m_TerminationEvent.Signal();
```

The main thread waits until the grab thread has terminated. How to create a Pylon::WaitObject from the native Win32 thread handle illustrated in the following code snippet:

```
// Join with the thread, i.e., wait until it has terminated
cout << "Going to join with the thread" << endl;
WaitObject woThread( m_hThread );
woThread.Wait(INFINITE);  // waits until the thread terminates
```

1.3.8 Handling Camera Events

Basler GigE Vision cameras can send event messages. For example, when a sensor exposure has finished, the camera can send an end-of-exposure event to the PC. The event can be received by the PC before the image data for the finished exposure has been completely transferred. The retrieving and processing of event messages is described in this section.

1.3.8.1 Event Grabbers

The Grabbing Images section describes how Stream Grabber objects are used to grab images from a camera. Analogously, Event Grabber objects are used to receive event messages from a camera.

1.3.8.2 Creating and Preparing Event Grabbers

Event Grabber objects are created and returned by Camera objects.

```
// Get the event grabber
Camera_t::EventGrabber_t EventGrabber = pCamera->GetEventGrabber( );
if ( ! EventGrabber.IsAttached() )
{
    cerr << "The camera doesn't supports event grabbing" << endl;
    return false;
}
```

Never try to call `free` or `delete` on IEventGrabber pointers. The camera object owns Event Grabbers and manages their lifetime.

Event Grabbers use internal memory buffers for receiving event messages. The number of buffers can be parametrized using the Event Grabber's NumBuffer member:

```
EventGrabber.NumBuffer.SetValue(20);
```

Note:
The number of buffers must be parametrized before calling the Event Grabber's Open() method!

A connection to the device and all necessary resources for receiving events are allocated by calling the Event Grabber's Open() method:

```
EventGrabber.Open();
```

1.3.8.3 Enabling Events

To let the camera send event messages, the sending of event messages must be enabled using the Camera object.

First, the EventSelector must be set to the type of event to be enabled. In the following example the selector is set to the end-of-exposure event:

```
// select the end-of-exposure event
Camera.EventSelector = EventSelector_ExposureEnd;
```

When the Event Selector is set, sending events of the desired type can be enabled by using the EventNotification parameter:

```
// enable sending of events of the selected event type
Camera.EventNotification.SetValue( EventNotification_GenICamEvent );
```

To be sure that you don't miss an event, the Event Grabber should be prepared before events are enabled (see the Creating and Preparing Event Grabbers section above).

The following code snippet illustrates how to disable the sending end-of-exposure events:

```
// select the end-of-exposure event
Camera.EventSelector = EventSelector_ExposureEnd;
// disable sending of end-of-exposure events
Camera.EventNotification.SetValue( EventNotification_Off );
```

1.3.8.4 Receiving Events

Receiving of events is very similar to grabbing images. The Event Grabber provides a wait object that is signalled when an event message is available. When an event message is available, it can be retrieved by calling the Event Grabber's RetrieveEvent() method.

In contrast to grabbing images, memory buffers for receiving events need not be provided by the application. Memory buffers to store event messages are organized by the Event Grabber itself.

In typical applications, waiting for grabbed images and event messages is done in one common loop. This is demonstrated in the following code snippet:

```
// Add the stream grabber’s and the event grabber’s wait objects to a container
WaitObjects waitset;
waitset.Add( EventGrabber.GetWaitObject() );
waitset.Add( StreamGrabber.GetWaitObject() );

while ( doGrabbing )
{
    // wait for an image or an event to occure (5 sec timeout)
    int idx;
    if ( waitset.WaitForAny( 5000, &idx ) )
    {
        // got event or image
        switch ( idx )
        {
            case 0: // event available, get the message
            {
                EventResult EvResult;
                if ( EventGrabber.RetrieveEvent( EvResult ) )
                {
                    if ( EvResult.Succeeded() )
                    {
                        // Successfully got the event message.
                        // EvResult.Buffer points to the message
                    }
                    else
                    {
                        // error occurred
                        cerr << "Error retrieving event:" << EvResult.ErrorDescription()
                        << endl;
                    }
                }
                else
                {
                    // No event available?
                    // Should never happen in this sample because the wait object
                    // was in signalled state when reaching this point.
                }
                break;
            } // case 0
            case 1: // image available, process it
            {
                GrabResult GrResult;
                if (StreamGrabber.RetrieveResult( GrResult ))
                {
                     if (GrResult.Succeeded())
                     {
                         // process the image

                         // reuse the buffer for further grabbing
                         StreamGrabber.QueueBuffer( GrResult.Handle(), GrResult.Context() );
                       }
                       else
                       {
                           // error handling
```

```
                        }
                    }
                } // case 1
            }  // switch
        } // if
        else
        {
            // timeout occurred
        }
    } // while
```

1.3.8.5 Parsing and Dispatching Events

The previous section explained how to receive an event message. This section describes how to interpret an event message.

The specific layout of event messages depends on the event type and the camera type. The pylon API uses GenICam support for parsing event messages. This means that the message layout is described in the camera's XML description file.

As described in the GenApi Node Maps section, a GenApi nodemap is created from the XML camera description file. That nodemap contains node objects representing the elements of the XML file. Since the layout of event messages is described in the camera description file, the information carried by the event messages is exposed as nodes in the nodemap. The camera object provides members used for accessing the event related nodes in the same way as camera parameter related nodes.

For example, an end-of-exposure event carries the following information:

- ExposureEndEventFrameID: indicates the number of the image frame that has been exposed
- ExposureEndEventTimestamp: indicates the moment when the event was generated
- ExposureEndEventStreamChannelIndex: indicates the number of the image data stream used to transfer the exposed frame

Example: The camera object's Pylon::CBasler1394Camera::ExposureEndEventFrameID member is used to access the number of the frame the event is associated with:

```
int64_t frameNr = Camera.ExposureEndEventFrameID.GetValue();
```

As described in the The 'Generic' Approach: Configuring Cameras Using Node Maps section, the ExposureEndEventFrameID could also be retrieved by using the camera object's nodemap directly:

```
GenApi::INodeMap* pNodeMap = pCamera->GetNodeMap();
CIntegerPtr ptrExposureEndFrameId = pNodeMap->GetNode("ExposureEndFrameId");
if ( ! ptrExposureEndFrameId ) {
    cerr << "There is no exposure time parameter" << endl;
    exit( 1 );
}
int64_t frameNr = ptrExposureEndFrameId->GetValue();
```

An Event Adapter object is used to update the event related nodes of the camera object's nodemap. Updating the nodes is done by passing the event message to an Event Adapter.

Event Adapters are created by Camera objects:

```
    Pylon::IEventAdapter *pEventAdapter = Camera.CreateEventAdapter();
    if ( ! pEventAdapter )
    {
        cerr << "Failed to create an event adapter" << endl;
    }
```

To update the event related nodes, call the Event Adapter's DeliverMessage() method for each received event message:

```
// Retrieve the event result
EventResult EvResult;
if ( EventGrabber.RetrieveEvent( EvResult ) )
{
    if ( EvResult.Succeeded() )
    {
         cout << "Successfully got an event message!" << endl;
         // Let the event adapter update the camera object's nodemap
         EventAdapter.DeliverMessage( EvResult.Buffer, sizeof EvResult.Buffer );
    }
    else
    {
         cerr << "Error retrieving event:" << EvResult.ErrorDescription() << endl;
    }
}
```

It is not possible to determine whether a message contains an end-of-exposure event by passing the event message to the Event Adapter. The next section describes how node callbacks are used to get informed about the occurrence of specific events.

1.3.8.6 Event Callbacks

The previous section described how Event Adapters are used to push the content of event messages into a camera object's nodemap. The IEventAdapter::DeliverMessages() method updates all nodes related to events contained in the message passed in.

As described in the Getting Informed About Parameter Changes section, it is possible to register callback functions that are fired when nodes may have been changed.

These callbacks can be used to determine if an event message contains a certain event type. For example, to get informed about end-of-exposure events, a callback for one of the end-of-exposure event related nodes must be installed. The following code snippet illustrates how to install a callback function for the ExposureEndFrameId node:

```
// Member function of this class will be registered as callback
struct CallbackTarget
{
    CallbackTarget( Camera_t& Camera )
        : m_Camera( Camera )
    { }

    // Will be fired when an end-of-exposure event occurres
    void EndOfExposureCallback( GenApi::INode* pNode )
     {
```

```
            try
            {
                cout << "The message contains an end-of-exposure event." << endl;
                cout << "Timestamp: " << m_pCamera->ExposureEndEventTimestamp.GetValue()
                     << "Frame number: " << m_pCamera->ExposureEndEventFrameID.GetValue() << endl;
            }
            catch ( GenICam::GenericException& e )
            {
                cerr << "Failed to get event information. Exception occurred:"
                     << e.GetDescription() << endl;
            }
        }

        Camera_t& m_Camera;
    } callbackTarget( Camera );

    // Register the callback for the ExposureEndEventTimestamp node.
    GenApi::CallbackHandleType hCb = GenApi::Register(
        pCamera->ExposureEndEventTimestamp.GetNode(),
        callbackTarget,
        &CallbackTarget::EndOfExposureCallback );
```

The registered callback will be fired from the context of the IEventAdapter::DeliverMessage() function.

Note:

Since one event message can aggreate multiple events, DeliverMessage will issue multiple calls to a callback function when multiple events of the same type are available.

1.3.8.7 Cleanup

Before closing and deleting the Camera object, the event related objects must be closed and destroyed as illustrated in the following code snippet:

```
// disable sending of end-of-exposure events
Camera.EventSelector = EventSelector_ExposureEnd;
Camera.EventNotification.SetValue( EventNotification_Off );

// Cleanup of event grabber and event adapter
// Deregister the callback first, or else when shutting down the event grabber
Camera.ExposureEndEventTimestamp.GetNode()->DeregisterCallback( hCb );
// Close the event grabber to tear down the connection
// and free the resources used for receiving events
EventGrabber.Close();
// Delete the event adapter object
Camera.DestroyEventAdapter( pEventAdapter );
```

1.3.8.8 Sample Program

The code snippets of this chapter are taken from the 'CameraEvents' sample program installed as part of the pylon SDK. The complete sample program is installed at <SDK ROOT>\samples.

1.3.9 Chunk Parser: Accessing Chunk Features

Basler Cameras can send additional information appended to the image data, such as frame counters, time stamps, and CRC checksums. How to enable Chunk Features and how to access the added data is explained in this section.

1.3.9.1 Enabling Chunks

Before a feature producing a chunk can be activated, the camera's chunk mode must be activated:

```
// Open the camera
Camera.Open();

// Enable chunks in general
if ( GenApi::IsWritable( Camera.ChunkModeActive ) )
{
   Camera.ChunkModeActive.SetValue( true );
}
else
{
   cerr << "The camera doesn't support chunk features" << endl;
   return 1;
 }
```

When the camera is in chunk mode, it transfers data blocks that are partitioned into chunks. The first chunk is always the image data. When chunk features are enabled, the image data chunk is followed by chunks containing the information generated by the chunk features.

When the chunk mode is activated, chunk features can be enabled:

```
Camera.ChunkSelector.SetValue( ChunkSelector_Timestamp );
Camera.ChunkEnable.SetValue( true );
```

1.3.9.2 Grabbing Buffers

Grabbing from an image stream with chunks is very similar to grabbing from an image stream without chunks. Memory buffers must be provided that are large enough to store both, the image data, and the added chunk data.

The camera's `PayloadSize` parameter reports the necessary buffersize (in bytes):

```
// ask for the buffer size
const size_t ImageSize = (size_t) ( Camera.PayloadSize.GetValue() );

// allocate buffer(s)
uchar8_t *pBuffer = new uchar8_t[ ImageSize ];

// inform the stream grabber about the buffer size
StreamGrabber.MaxBufferSize.SetValue( ImageSize );

// tell the stream grabber how many buffers will be used
// ( in this example only 1 )
StreamGrabber.MaxNumBuffer.SetValue( 1 );
```

Now an image plus added chunks can be grabbed:

```
// allocate resources related to grabbing
StreamGrabber.PrepareGrab();

// subscribe the buffer
const StreamBufferHandle hBuffer =
  StreamGrabber.RegisterBuffer( pBuffer, ImageSize );

// Put buffer into the grab queue for being filled with data
StreamGrabber.QueueBuffer( hBuffer, NULL );

// Let the camera acquire one image
Camera.AcquisitionMode.SetValue( AcquisitionMode_SingleFrame );
Camera.AcquisitionStart.Execute();


GrabResult Result;
// Wait for the buffer to be filled
if ( StreamGrabber.GetWaitObject().Wait( 3000 ) )
{
  // Get the grab result from the grabber's result queue
  StreamGrabber.RetrieveResult( Result );

}
else {
  // Timeout
  cerr << "Timeout occurred!" << endl;
  return 1;
}

if ( ! Result.Succeeded() )
{
  // Error Handling
  cerr << "No image acquired!" << endl;
  cerr << "Error code : 0x" << hex
    << Result.GetErrorCode() << endl;
  cerr << "Error description : "
    << Result.GetErrorDescription() << endl;
  return 1;
}

// Check if a buffer containing chunk data has been received
if ( PayloadType_ChunkData != Result.GetPayloadType () )
{
  cerr << "Unexpected payload type received" << endl;
  return 1;
}
```

1.3.9.3 Accessing the Chunk Data

The data block containing the image chunk and the other chunks has a self-descriptive layout. Before accessing the data of the added chunks, the data block must be parsed by a Chunk Parser object.

The Camera object is responsible for creating a Chunk Parser:

```
// Create ChunkParser
IChunkParser &ChunkParser = *Camera.CreateChunkParser();
```

Once a Chunk Parser is created, grabbed buffers can be attached to the Chunk Parser. When a buffer is attached to a chunk parser, it is parsed and the chunk data access is provided by members of the Camera object.

```
// Attach image buffer with chunk data to the parser. The parser extracts
// the included data from the chunk.
pChunkParser->AttachBuffer( Result.Buffer(), ImageSize );

// Access the chunk data.
// Before accessing the chunk data, it should be checked
// if the chunk is readable. When it is readable, the buffer
// contains the requested chunk data.
if ( IsReadable(Camera.ChunkTimestamp) )
  cout << "TimeStamp : " << Camera.ChunkTimestamp.GetValue() << endl;
```

For checking the result of the CRC Checksum chunk feature, use the Chunk Parser's `HasCRC()` and `CheckCRC()` methods. Note, that the camera only sends a CRC when the CRC Checksum feature is enabled.

```
// Enable crc chunks  ( before PrepareGrab()! )
Camera.ChunkSelector.SetValue( ChunkSelector_PayloadCRC16 );
Camera.ChunkEnable.SetValue( true );

...

// Check the CRS sum ( after having the buffer attached to the Chunk Parser)
if (pChunkParser->HasCRC() &&
  pChunkParser->CheckCRC() == false)
{
  cerr << "Image was damaged!" << endl;
  return 1;
}
```

Before reusing a buffer for grabbing, the buffer must be detached from the Chunk Parser.

```
// After detaching the buffer, the chunk data is no longer accessible!
pChunkParser->DetachBuffer();
```

After detaching a buffer, the next grabbed buffer can be attached and the included chunk data can be read.

When having finished grabbing, the Chunk Parser must be deleted:

```
// Destroy the chunk parser
Camera.DestroyChunkParser( pChunkParser );
```

1.3.9.4 Sample Program

The code snippets of this chapter are taken from the 'AcquireSingleFrame_ChunkImage' sample program installed as part of the pylon SDK. The complete sample program is installed at <SDK ROOT>\samples.

1.3.10 Getting Informed About Device Removal

Callback functions can be installed that are triggered, when a Camera device has been removed. As soon as the Camera object's `Open()` method has been called, either a C or C++ class member functions can be installed as callbacks.

Installing a C function:

```
  void RemovalCallbackFunction( IPylonDevice* pDevice )
  {
    cout << endl << "Callback function for removal of device "
        << pDevice->GetDeviceInfo().GetFullName().c_str() << " has been fired" << endl;
  }
  ...
  {
    // Open the camera
    pCamera->Open();

    // Register a "normal" function
    DeviceCallbackHandle hCb2 = RegisterRemovalCallback( pCamera, &RemovalCallbackFunction);
  }
```

Installing a C++ class member function:

```
 A class with a member function that can be registered for device removal notifications
class AClass
{
public:
    // The member function to be registered
    void RemovalCallbackMemberFunction( IPylonDevice* pDevice )
    {
        cout << endl << "Member function callback for removal of device "
            << pDevice->GetDeviceInfo().GetFullName().c_str() << " has been fired" << endl;
    }
};
...

  {
    AClass a;  // a member function of this class will be registered as a removal callback function
   ...

    // Open the camera
    pCamera->Open();

   // Register a member function
   DeviceCallbackHandle hCb1
       = RegisterRemovalCallback( pCamera, a, &AClass::RemovalCallbackMemberFunction);
  }
```

All registered callbacks must be deregistered before calling the Camera object's `Close()` method.

```
    if ( ! pCamera->DeregisterRemovalCallback( hCb1 ) )
        cerr << "Failed to deregister the callback function" << endl;
```

```
pCamera->Close();
```

The code snippets of this section are taken from the 'RegisterRemovalCallback' sample program installed as part of the pylon SDK. The complete sample program is installed at <SDK ROOT>\samples.

1.4 Migrating From pylon 1.0 to pylon 2.0

The programming interface for pylon 2.0 is in some respects not backwards compatible with pylon 1.0. Breaking the backward compatibility was necessary to allow handling of camera description XML files in a more flexible way.

With pylon 2.0, a transport layer can support different camera types, each of them having a different XML file. For example, the Basler pilot and Basler runner GigE cameras have different XML files. In pylon 2.0, these XML files are downloaded from the camera by default if the camera supports downloading XML files.

The following chapters describe the necessary changes for compiling an application using the pylon 2.0 SDK, when the application was originally developed using pylon 1.0.

1.4.1 Update to GenICam 1.1

Pylon 1.0 was based on GenICam 1.0. In pylon 2.0, the GenICam version 1.1 is used.

GenICam 1.0 used the `GENICAM_ROOT` environment variable. This variable is no longer used by GenICam 1.1 (and is therefore not used by pylon 2.0).

Instead, the `GENICAM_ROOT_V1_1` environment variable must be used. Replace all references to `GENICAM_ROOT` by `GENICAM_ROOT_V1_1`, for example in include or library path settings as shown below.

1.4.2 Convenience Header File

In Pylon 2.0, the `PylonIncludes.h` header file has been added. This header file includes the pylon header files that are typically needed for building pylon based applications. In addition, the header file includes linker statements causing the linker to automatically link the import libraries needed to build pylon based applications. It is no longer necessary to explicitly link the pylon libraries like `PylonBase_MD_VC80.lib`.

It isn't mandatory to use the `PylonIncludes.h` header file, but it is convenient since the import libraries are linked automatically.

The following code snippet illustrates how to include the `PylonIncludes.h` file:

```
#include <pylon/PylonIncludes.h>
```

1.4.3 Compiler and Linker Settings

1.4.3.1 Include Path

All appearances of `GENICAM_ROOT` in the compiler's include path settings must be replaced by `GENICAM_ROOT_V1_1`.

Alternatively, you can replace `$(GENICAM_ROOT)` by `$(PYLON_ROOT)/../genicam` as shown in the Include Path section.

1.4.3.2 Linker Settings

In the linker's library path settings, `GENICAM_ROOT` must be replaced by GENICAM_ROOT_V1_1.

Alternatively, you could replace `$(GENICAM_ROOT)` by `$(PYLON_ROOT)/../genicam` as shown in the Library Path and Library Files section.

When using the `PylonIncludes.h` header file (see Convenience Header File), the pylon and GenICam import libraries (e.g. `GenApi_MD_VC80.lib`, `GCBase_MD_VC80.lib`, and `PylonBase_MD_VC80.lib`) can be removed from the 'Additional Dependencies' list in the linker settings property page.

1.4.4 Changes for the 'Native' Programming Approach

1.4.4.1 Device Creation

When using device specific camera classes as described in the The 'Native' Approach: Configuring Cameras Using a Camera Class section, some changes are necessary. For creating a camera class in pylon 1.0, it was necessary to downcast the IPylonDevice pointer returned by the Transport Layer object or by the Transport Layer factory to a camera class. The reason being that the objects returned by the `CreateDevice()` methods in the form of `IPylonDevice` pointers were instances of the `CBasler1394Camera` and `CBaslerGigECamera` classes, respectively. That is to say, the Transport Layer Factory or the Transport Layer objects were responsible for creating specific camera classes.

In pylon 2.0, the `CreateDevice()` methods are returning pure IPylonDevice pointers that cannot be cast to specific camera classes any more. Instead, a camera class behaves like a smart pointer, that is created outside of a Transport Layer and is attached to a IPylonDevice pointer. The Camera class takes the ownership of IPylonDevice pointers.

The following code snippets show how to use the new camera class:

```
pylon 1.0: creating a CBasler1394 camera object...
 Pylon::CBasler1394Camera &Camera
    = dynamic_cast<Pylon::CBasler1394Camera &> (*pTl->CreateDevice (devices[0]));

 // ... becomes

 Pylon::CBasler1394Camera Camera = pTl->CreateDevice (devices[0]);

 // or
 Pylon::CBasler1394Camera Camera;
 Camera.Attach (pTl->CreateDevice (devices[0]));

 // or

 Pylon::CBasler1394Camera *Camera;
 Camera = new Pylon::CBasler1394Camera (pTl->CreateDevice (devices[0]));


 // pylon 1.0: creating a CBaslerGigE camera object...
 Pylon::CBaslerGigECamera &Camera
  = dynamic_cast<Pylon::CBaslerGigECamera &> (*pTl->CreateDevice (devices[0]));

 // ... becomes

 Pylon::CBaslerGigECamera Camera = pTl->CreateDevice (devices[0]);
```

```
// or
Pylon::CBaslerGigECamera Camera;
Camera.Attach (pTl->CreateDevice (devices[0]));

// or

Pylon::CBaslerGigECamera *Camera;
Camera = new Pylon::CBaslerGigECamera (pTl->CreateDevice (devices[0]));
```

1.4.4.2 Stream Grabber Creation

Using a Stream Grabber class (e.g., the Pylon::CBasler1394Camera::StreamGrabber_t class) is a convenient way of accessing a Stream Grabber's parameters. In pylon 2.0, Stream Grabber classes behave very similarly to Camera classes. A Stream Grabber class takes the ownership of a Pylon::IStreamGrabber pointer.

In pylon 1.0 a Stream Grabber class was created as shown in the following example:

```
Pylon::CBasler1394Camera::StreamGrabber_t &StreamGrabber
 = dynamic_cast <Pylon::CBasler1394Camera::StreamGrabber_t&>(*Camera.GetStreamGrabber (0));
```

In pylon 2.0, this becomes one of the following variants:

```
Pylon::CBasler1394Camera::StreamGrabber_t StreamGrabber =
  Camera.GetStreamGrabber (0);

// or

Pylon::CBasler1394Camera::StreamGrabber_t StreamGrabber;
// ..
StreamGrabber.Attach( Camera.GetStreamGrabber(0) );

// or

Pylon::CBasler1394Camera::StreamGrabber_t * StreamGrabber;
StreamGrabber = new Pylon::CBasler1394Camera::StreamGrabber_t (m_pCamera->GetStreamGrabber (0));
```

1.4.4.3 Device Cleanup

In pylon 1.0, the Transport Layer Factory's or the Transport Layer object's `DestroyDevice()` method was used for deleting camera objects. In pylon 2.0, the camera class takes the ownership of the `IPylonDevice` pointer. The destructor of the camera class is responsible for cleaning up the camera object.

When migrating a pylon 1.0 application to pylon 2.0, Calls to the `DestroyDevice()` methods must be removed where `CBasler1394Camera` or `CBaslerGigECamera` pointers are used as arguments.

1.4.5 Changes for the 'Generic' Programming Approach

When using the 'generic' programming approach (see The 'Generic' Approach: Configuring Cameras Using Node Maps), there are no changes with respect to the use of `IPylonDevice` and `IStreamGrabber`. There are some minor differences when accessing GenApi nodes. These differences are explained below in the Node Access section.

1.4.6 Node Access

In pylon 1.0, the `GetNodeMap()` methods were returning objects of type `CNodeMapRef`. In pylon 2.0, GenApi::INodeMap pointers are returned. Therefore, the INodeMap::GetNode() method must be used for accessing GenApi nodes instead of the `CNodeMapRef::_GetNode()` method.

Example:

```
GenApi::CNodeMapRef *MyNodeMap = pDevice->GetNodeMap();

GenApi::INode* MyNode = MyNodeMap->_GetNode("GainRaw");
```

becomes

```
GenApi::INodeMap* MyNodeMap = pDevice->GetNodeMap();

GenApi::INode* MyNode = MyNodeMap->GetNode("GainRaw");
```

1.4.7 Transport Layer Object Cleanup

In pylon 1.0, the CTlFactory::DestroyTl() method was used for freeing a Transport Layer object. In pylon 2.0, use the ReleaseTl() method instead.

1.5 How to Control the Location of the Camera Description Files Used by pylon

When a Camera object is created, the GenApi Node Map used to access the camera parameters is generated from a camera description XML file. See the GenApi Node Maps section for a description of GenApi Node Maps.

For GigE cameras, per default this XML file is downloaded from the camera. For 1394 cameras, a description file installed by the pylon installer is used.

It is possible, to change this default behaviour. The location of the XML files is defined by a set of rules. The rules are stored in the <SDK ROOT>\pylon\ConfigFileRegistry\StandardRules.mml file.

The following example describes a rule matching to certain GigE scout cameras. For the matching cameras an XML file stored in the PC's filesystem is used instead of downloading the file from the camera.

```
<TransportLayer name="BaslerGigE">
   <Rule priority="10">
     <Manufacturer>Basler</Manufacturer>
     <Name>sca.*</Name>
     <SerialNumber>206102.*</SerialNumber>
     <Download>no</Download>
     <File>c:\myfiles\myConfigFile.xml</File>
   </Rule>
</TransportLayer>
```

The rule will be applied to cameras created by the BaslerGigE transport layer, i.e., the transport layer for GigE cameras. The rule matches all Basler scout cameras (scA....) having a Serial number starting

with '206102'. For the matching cameras the specified camera configuration file is to be used instead of downloading it from the camera.

Each rule has a priority. When several rules are matching to a device, the rule with the highest priority number is applied.

For the rule criteria `Manufacturer`, `Name`, and `Serial` Number regular expressions in Perl syntax can be specified.

It is possible to specifiy a `File` even when `Download` is set to 'yes'. In this case, the specified file is taken when the download from the camera has failed for some reason.

The above mentioned StandardRules.xml file must not be edited. Instead, create a new rules file in the \ConfigFileRegistry folder. The StandardRules.xml file contains a copy and paste template for such a custom rule file.

1.6 Software Licensing Information

The pylon API is based on the GenApi module of the GenICam (TM) reference implementation distributed under a modified BSD license and is copyright (c) 2005, Basler Vision Technologies. All rights reserved. Redistribution and use in source and binary forms, without modification, are permitted provided that the following conditions are met:

- Redistributions of the source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the GenICam standard group nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE HELD LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND BASED ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Chapter 2

Module Documentation

2.1 Public interface and definitions in Pylon

Files

- file PixelType.h

  Definition of types concerning image format and pixel type data.

- file PixelTypeFunctions.h

  Definition of types concerning image format and pixel type data.

- file Result.h

  Definition of a of types concerning grabbed data.

Classes

- class Pylon::AccessModeSet

  Collection of AccessMode bits.

- class Pylon::CDeviceInfo

  A property container for device information.

- class Pylon::WaitObject

  A wait object.

- class Pylon::AlertableWaitObject

  A wait object that the user may signal.

- class Pylon::WaitObjects

  A set of wait objects.

- class Pylon::GrabResult

  A grab result that combines the used image buffer and status information.

- class Pylon::EventResult

  An event result.

- class Pylon::CTlFactory

  the Transport Layer Factory

- interface Pylon::IChunkParser

  The interface for chunk parsers analyzing a data stream consisting of chunks.

- interface Pylon::IDevice

  The interface implemented by all device objects.

- interface Pylon::IPylonDevice

  Interface for camera objects.

- interface Pylon::IDeviceFactory

  Interface to be implemented by device factories used to create devices.

- interface Pylon::IEventAdapter

  IEventAdapter delivers event messages to ports.

- interface Pylon::IEventGrabber

  Interface of an object receiving asynchronous events.
  Asynchronous event messages are received from the camera. Internal Buffers are filled and stored in an output queue. While the output queue contains data the associated waitobject is signalled.

- interface Pylon::IStreamGrabber

  Interface to an (input) data stream.

Functions

- AccessModeSet Pylon::operator| (AccessMode lhs, AccessMode rhs)

  Creates a set containing lhs and rhs operands.

- AccessModeSet Pylon::operator| (const AccessModeSet &lhs, AccessMode rhs)

  Adds the operand rhs to the set lhs.

- template<class Function> DeviceCallbackHandle Pylon::RegisterRemovalCallback (IPylonDevice *pDevice, Function f)

  Register a C-function as a removal callback.

- template<class Client, class Member> DeviceCallbackHandle Pylon::RegisterRemovalCallback (IPylonDevice *pDevice, Client &c, Member m)

  Register a C++-member function as removal callback.

2.1.1 Function Documentation

2.1.1.1 AccessModeSet Pylon::operator| (const AccessModeSet & *lhs*, AccessMode *rhs*)

Adds the operand rhs to the set lhs.

Parameters:

lhs a set of bits.

rhs the additional bit

2.1.1.2 AccessModeSet Pylon::operator| (AccessMode *lhs*, AccessMode *rhs*)

Creates a set containing lhs and rhs operands.

Parameters:

lhs left operand

rhs right operand

2.1.1.3 template<class Client, class Member> DeviceCallbackHandle Pylon::RegisterRemovalCallback (IPylonDevice * *pDevice*, Client & *c*, Member *m*)

Register a C++-member function as removal callback.

See also:

Pylon::IPylonDevice::RegisterRemovalCallback()

Parameters:

pDevice Pointer to the device that generates callbacks

c The client object

m The member function to be called

2.1.1.4 template<class Function> DeviceCallbackHandle Pylon::RegisterRemovalCallback (IPylonDevice * *pDevice*, Function *f*)

Register a C-function as a removal callback.

See also:

Pylon::IPylonDevice::RegisterRemovalCallback()

Parameters:

pDevice Pointer to the device that generates callbacks

f The function to be called

2.2 The GigE specific definitions

Classes

- class Pylon::CBaslerGigEDeviceInfo

  Implementation of the GiGE specific device info object.

2.3 The IEEE1394 specific definitions

Classes

- class Pylon::CBasler1394DeviceInfo

  FireWire specific Device Info object. Does not support SerialNumber property.

- class Pylon::CBasler1394TI

  The FireWire specific implemenation of a TLFactory.

2.4 Abstract interface classes of the GenApi module

Files

- file IBase.h

  Definition of interface IBase.

- file IBoolean.h

  Definition of IBoolean interface.

- file ICategory.h

  Definition of interface ICategory and types FeatureList_t :.

- file ICommand.h

  Definition of ICommand interface.

- file IEnumEntry.h
- file IEnumeration.h

  Definition of interface IEnumeration.

- file IEnumerationT.h

  Definition of interface IEnumerationT.

- file IFloat.h

  Definition of the IFloat interface.

- file IInteger.h

  Definition of the interface IInteger.

- file INode.h

  Definition of interface INode and types NodeList_t and CallbackHandleType:.

- file INodeMap.h

  Definition of interface INodeMap.

- file IPort.h

  Definition of interface IPort.

- file IRegister.h

  Definition of RegisterList_t type and the interface IRegister.

- file IString.h

  Definition of interface IString.

- file IValue.h

  Definition of the interface IValue.

- file NodeMapRef.h

  Definition of CNodeMapRef.

Classes

- struct GenApi::IBase

  Base interface common to all nodes.

- struct GenApi::IBoolean

  Interface for Boolean properties.

- struct GenApi::ICategory

  Gives access to a category node.

- struct GenApi::ICommand
- struct GenApi::IEnumeration

  Interface for enumeration properties.

- struct GenApi::IEnumerationT< EnumT >

  Interface for enumeration properties.

- struct GenApi::IFloat

  Interface for float properties.

- struct GenApi::IInteger

  Interface for integer properties.

- struct GenApi::INode

  Interface common to all nodes.

- struct GenApi::INodeMap

  Interface to access the node map.

- struct GenApi::IPort

  Interface for ports.

- struct GenApi::IRegister

  Interface for registers.

- struct GenApi::IString

  Interface for string properties.

- struct GenApi::IValue

  Interface for value properties.

- class GenApi::CNodeMapRef

  Smartpointer for NodeMaps with create function.

Functions

- bool GenApi::IsReadable (EAccessMode AccessMode)

  Tests if readable.

- bool GenApi::IsReadable (const IBase *p)

  Checks if a node is readable.

- bool GenApi::IsReadable (const IBase &r)

  Checks if a node is readable.

- bool GenApi::IsWritable (EAccessMode AccessMode)

  Tests if writable.

- bool GenApi::IsWritable (const IBase *p)

  Checks if a node is writable.

- bool GenApi::IsWritable (const IBase &r)

  Checks if a node is writable.

- bool GenApi::IsImplemented (EAccessMode AccessMode)

  Tests if implemented.

- bool GenApi::IsImplemented (const IBase *p)

  Checks if a node is implemented.

- bool GenApi::IsImplemented (const IBase &r)

  Checks if a node is implemented.

- bool GenApi::IsAvailable (EAccessMode AccessMode)

  Tests if available.

- bool GenApi::IsAvailable (const IBase *p)

  Checks if a node is available.

- bool GenApi::IsAvailable (const IBase &r)

  Checks if a node is available.

- EAccessMode GenApi::Combine (EAccessMode Peter, EAccessMode Paul)

  Computes which access mode the two guards allow together.

- bool GenApi::IsVisible (EVisibility Visibility, EVisibility MaxVisiblity)

  Tests Visibility.

- EVisibility GenApi::Combine (EVisibility Peter, EVisibility Paul)

  Computes which visibility the two guards allow together.

2.4.1 Function Documentation

2.4.1.1 bool GenApi::IsVisible (EVisibility *Visibility*, EVisibility *MaxVisiblity*) [inline]

Tests Visibility.

CAVE : this relys on the EVisibility enum's coding

2.5 Public utilities of the GenApi module

Files

- file chunkadapter.h

  Declaration of the CChunkAdapter class.

- file Pointer.h

  Definition of template CPointer.

Classes

- struct GenApi::AttachStatistics_t
- class GenApi::CChunkAdapter

  Connects a chunked buffer to a node map.

- class GenApi::CPointer< T, B >

  Encapsulates a GenApi pointer dealing with the dynamic_cast automatically.

Typedefs

- typedef GenApi::AttachStatistics_t GenApi::AttachStatistics_t
- typedef CPointer< IBase > GenApi::CBasePtr

  SmartPointer for IBase interface pointer.

- typedef CPointer< INode > GenApi::CNodePtr

  SmartPointer for INode interface pointer.

- typedef CPointer< IValue > GenApi::CValuePtr

  SmartPointer for IValue interface pointer.

- typedef CPointer< ICategory > GenApi::CCategoryPtr

  SmartPointer for ICategory interface pointer.

- typedef CPointer< IInteger > GenApi::CIntegerPtr

  SmartPointer for IInteger interface pointer.

- typedef CPointer< IBoolean > GenApi::CBooleanPtr

  SmartPointer for IBoolean interface pointer.

- typedef CPointer< IFloat > GenApi::CFloatPtr

  SmartPointer for IFloat interface pointer.

- typedef CPointer< IString > GenApi::CStringPtr

  SmartPointer for IString interface pointer.

- typedef CPointer< IRegister > GenApi::CRegisterPtr

SmartPointer for IRegister interface pointer.

- typedef CPointer< IEnumeration > GenApi::CEnumerationPtr

  SmartPointer for IEnumeration interface pointer.

- typedef CPointer< IEnumEntry > GenApi::CEnumEntryPtr

  SmartPointer for IEnumEntry interface pointer.

- typedef CPointer< IPort > GenApi::CPortPtr

  SmartPointer for IPort interface pointer.

- typedef CPointer< IChunkPort > GenApi::CChunkPortPtr

  SmartPointer for IChunkPort interface pointer.

- typedef CPointer< INodeMap, INodeMap > GenApi::CNodeMapPtr

  SmartPointer for INodeMap interface pointer.

- typedef CPointer< INodeMapDyn, INodeMap > GenApi::CNodeMapDynPtr

  SmartPointer for INodeMapDyn interface pointer.

- typedef CPointer< IDeviceInfo, INodeMap > GenApi::CDeviceInfoPtr

  SmartPointer for IDeviceInfo interface pointer.

- typedef CPointer< ISelector > GenApi::CSelectorPtr

  SmartPointer for ISelector interface pointer.

- typedef CPointer< ICommand > GenApi::CCommandPtr

  SmartPointer for ICommand interface pointer.

Functions

- template<class Function> intptr_t GenApi::Register (INode *pNode, Function f)

  Register a C-function as a callback.

- template<class Client, class Member> intptr_t GenApi::Register (INode *pNode, Client &c, Member m)

  Register a C++-member function a callback.

2.5.1 Typedef Documentation

2.5.1.1 typedef struct GenApi::AttachStatistics_t GenApi::AttachStatistics_t

Delivers information about the attached chunks and nodes

2.6 Implementation parts of the GenApi module which must be public

Files

- file genicam/library/CPP/Include/genapi/Container.h

  Portable container classes for INode & IValue pointers.

- file NodeCallback.h

  Defines the NodeCallback class.

- file NodeCallbackImpl.h

  Implementation helpers for NodeCallback
  Defines a bunch of templates for creating callback-functors taking a INode pointer and returning void.

Classes

- struct GenApi::ISelector

  Interface for groups of features selected by a single one.

- class GenApi::NodeCallback

  Callback object for callback taking an INode pointer.

- class GenApi::NodeCallbackBase

  callback body instance for INode pointers

- class GenApi::Function_NodeCallbackBody< Function >

  Container for a function pointer.

- class GenApi::Member_NodeCallbackBody< Client, Member >

  Container for a member function pointer.

Typedefs

- typedef CBaseRefT< IBase > GenApi::CBaseRef
- typedef CBooleanRefT< IBoolean > GenApi::CBooleanRef
- typedef CCategoryRefT< ICategory > GenApi::CCategoryRef
- typedef CCommandRefT< ICommand > GenApi::CCommandRef
- typedef CEnumEntryRefT< IEnumEntry > GenApi::CEnumEntryRef
- typedef CFloatRefT< IFloat > GenApi::CFloatRef
- typedef CIntegerRefT< IInteger > GenApi::CIntegerRef
- typedef CPortRefT< IPort > GenApi::CPortRef
- typedef CRegisterRefT< IRegister > GenApi::CRegisterRef
- typedef CSelectorRefT< ISelector > GenApi::CSelectorRef
- typedef CStringRefT< IString > GenApi::CStringRef
- typedef CValueRefT< IValue > GenApi::CValueRef
- typedef CPointer< IPortConstruct > GenApi::CPortConstructPtr

  SmartPointer for IPortConstruct interface pointer.

Functions

- template<class Function> NodeCallback GenApi::make_NodeCallback (Function function)

  make a new callback object for C functions

- template<class Client, class Member> NodeCallback GenApi::make_NodeCallback (Client &client, Member member)

  make a new callback object for member functions

- GenICam::gcstring GenApi::GetInterfaceName (IBase *pBase)

  Returns the name of the main interface as string.

2.6.1 Typedef Documentation

2.6.1.1 typedef CBaseRefT<IBase> GenApi::CBaseRef

Reference to an IBase pointer

2.6.1.2 typedef CBooleanRefT<IBoolean> GenApi::CBooleanRef

Reference to an IBoolean pointer

2.6.1.3 typedef CCategoryRefT<ICategory> GenApi::CCategoryRef

Reference to an ICategory pointer

2.6.1.4 typedef CCommandRefT<ICommand> GenApi::CCommandRef

Reference to an ICommand pointer

2.6.1.5 typedef CEnumEntryRefT<IEnumEntry> GenApi::CEnumEntryRef

Reference to an IEnumEntry pointer

2.6.1.6 typedef CFloatRefT<IFloat> GenApi::CFloatRef

Reference to an IFloat pointer

2.6.1.7 typedef CIntegerRefT<IInteger> GenApi::CIntegerRef

Reference to an IInteger pointer

2.6.1.8 typedef CPortRefT<IPort> GenApi::CPortRef

Reference to an IEnumEntry pointer

2.6.1.9 typedef CRegisterRefT<IRegister> GenApi::CRegisterRef

Reference to an IRegister pointer

2.6.1.10 typedef CSelectorRefT<ISelector> GenApi::CSelectorRef

Reference to an ISelector pointer

2.6.1.11 typedef CStringRefT<IString> GenApi::CStringRef

Reference to an IString pointer

2.6.1.12 typedef CValueRefT<IValue> GenApi::CValueRef

Reference to an IValue pointer

2.7 Public utilities of the GenApi Log module

2.8 Public utilities of the GenICam Base module

Files

- file GCException.h
- file GCString.h

  Portable string implementation.

Classes

- class GenICam::GenericException

  GenICam's exception class.

- class GenICam::InvalidArgumentException

  Exception fired if an argument is invalid.

- class GenICam::OutOfRangeException

  Exception fired if an argument is out of range.

- class GenICam::PropertyException

  Exception fired if a property access fails.

- class GenICam::RuntimeException

  Runtime exception.

- class GenICam::LogicalErrorException

  Exception to be thrown to indicate logical errors in program flow.

- class GenICam::AccessException

  Exception to be thrown to indicate an access error.

- class GenICam::TimeoutException

  Exception to be thrown to indicate an access error.

- class GenICam::DynamicCastException

  Exception to be thrown to indicate the the result of a dynamic cast was zero.

- class GenICam::gcstring

  A string class which is a clone of std::string.

Defines

- #define DECLARE_EXCEPTION(name)

Functions

- std::ostream & operator<< (std::ostream &ostr, const GenICam::gcstring &str)
- std::istream & operator>> (std::istream &istr, GenICam::gcstring &str)

2.8.1 Define Documentation

2.8.1.1 #define DECLARE_EXCEPTION(name)

Value:

```
class GCBASE_API name : public GenICam::GenericException \
        { \
        public: \
            name(const char* description, const char *pSourceFileName, int SourceLine ); \
        }
```

Creates an exception with the same functionality as the GenericException but being of different type

2.8.2 Function Documentation

2.8.2.1 std::ostream& operator<< (std::ostream & *ostr*, const GenICam::gcstring & *str*) [inline]

STL operator out

2.8.2.2 std::istream& operator>> (std::istream & *istr*, GenICam::gcstring & *str*) [inline]

STL operator in

2.9 Implementation parts of the GenICam Base module

Classes

- class GenICam::ExceptionReporter< E >

  printf like creation of exceptions

Defines

- #define GENERIC_EXCEPTION GenICam::ExceptionReporter<GenICam::GenericException>(__FILE__, __LINE__).Report

  Fires an exception, e.g. throw EXCEPTION("%ld too large", Value);.

- #define INVALID_ARGUMENT_EXCEPTION GenICam::ExceptionReporter<GenICam::InvalidArgumentException>(__FILE__, __LINE__).Report

  Fires an invalid argument exception, e.g. throw INVALID_ARGUMENT("%ld too large", Value);.

- #define OUT_OF_RANGE_EXCEPTION GenICam::ExceptionReporter<GenICam::OutOfRangeException>(__FILE__, __LINE__).Report

  Fires an out of range exception, e.g. throw OUT_OF_RANGE_EXCEPTION("%ld too large", Value);.

- #define PROPERTY_EXCEPTION GenICam::ExceptionReporter<GenICam::PropertyException>(__FILE__, __LINE__).Report

  Fires an property exception, e.g. throw PROPERTY_EXCEPTION("%ld too large", Value);.

- #define RUNTIME_EXCEPTION GenICam::ExceptionReporter<GenICam::RuntimeException>(__FILE__, __LINE__).Report

  Fires a runtime exception, e.g. throw RUNTIME_EXCEPTION("buh!").

- #define LOGICAL_ERROR_EXCEPTION GenICam::ExceptionReporter<GenICam::LogicalErrorException>(__FILE__, __LINE__).Report

  Fires a logical error exception, e.g. throw LOGICAL_ERROR_EXCEPTION("Should never reach this point").

- #define ACCESS_EXCEPTION GenICam::ExceptionReporter<GenICam::AccessException>(__FILE__, __LINE__).Report

  Fires a access error exception, e.g. throw ACCESS_ERROR_EXCEPTION("Not ecverybody").

- #define TIMEOUT_EXCEPTION GenICam::ExceptionReporter<GenICam::AccessException>(__FILE__, __LINE__).Report

  Fires a timeout error exception, e.g. throw TIMEOUT_EXCEPTION("Not ecverybody").

- #define DYNAMICCAST_EXCEPTION GenICam::ExceptionReporter<GenICam::DynamicCastException>(__FILE__, __LINE__).Report

  Fires a dynamic cast exception, e.g. throw DYNAMICCAST_EXCEPTION("Not ecverybody").

- #define CHECK_RANGE_I64(_Value, _Min, _Max, _Inc)

  Range check for int64.

- #define CHECK_RANGE_FLT(_Value, _Min, _Max)

Range check for float.

- #define CHECK_DYNAMIC_CAST_POINTER(_Pointer)

  Checks if a dynamic_cast is possible.

Chapter 3

Namespace Documentation

3.1 Pylon Namespace Reference

Contains definitions of Pylon related types.

Classes

- class Base_Callback1Body

  abstract callback container

- class Callback1

  callback object with one parameter P1

- class Function_CallbackBody

  Container for a function pointer.

- class Member_CallbackBody

  Container for a member function pointer.

- interface IChunkParser

  The interface for chunk parsers analyzing a data stream consisting of chunks.

- class CChunkParser

  Base class for chunk parsers returned by camera objects.

- class DeviceInfoList

  STL std::vector like container for Pylon::CDeviceInfo objects.

- class TlInfoList

  STL std::vector like container for Pylon::CTlInfo objects.

- class AccessModeSet

  Collection of AccessMode bits.

- interface IDevice

The interface implemented by all device objects.

- interface IPylonDevice

  Interface for camera objects.

- interface IDeviceFactory

  Interface to be implemented by device factories used to create devices.

- class CDeviceInfo

  A property container for device information.

- interface IEventAdapter

  IEventAdapter delivers event messages to ports.

- interface IEventGrabber

  Interface of an object receiving asynchronous events.
  Asynchronous event messages are received from the camera. Internal Buffers are filled and stored in an output queue. While the output queue contains data the associated waitobject is signalled.

- class CEventGrabberProxy

  A Handle class (aka Envelope class) for the IEventGrabber interface.

- interface IProperties

  interface for a property container

- interface IInfo

  interface for an info object

- class CInfoBase

  Base implementation for PYLON info container.

- class CNodeMapProxy

  A Handle class (aka Envelope class) for the GenApi::INodeMap interface.

- class CPixelFormatConverter

  Abstract base class for Color Converter classes.

- class CPixelFormatConverterBayer

  Class for converting Bayer images to given output formats by using truncation.

- class CPixelFormatConverterGamma

  Class for converting Mono images to given output formats by using gamma correction.

- class CPixelFormatConverterGammaPacked

  Class for converting Mono packed images to given output formats by using gamma correction.

- class CPixelFormatConverterMonoPacked

  Abstract base class for converter classes converting Mono packed images to given output formats.

- class CPixelFormatConverterMonoXX

Abstract base class for converter classes converting Mono images to given output formats.

- class CPixelFormatConverterRGB

  Class for converting RGB and BGR images to given output formats by using truncation.

- class CPixelFormatConverterTruncate

  Class for converting Mono images to given output formats by using truncation.

- class CPixelFormatConverterTruncatePacked

  Class for converting Mono packed images to given output formats by using truncation.

- class CPixelFormatConverterYUV422

  Class for converting YUV422 UYVY images to given output formats.

- class CPixelFormatConverterYUV422YUYV

  Class for converting YUV422 YUYV images to given output formats.

- class WaitObject

  A wait object.

- class AlertableWaitObject

  A wait object that the user may signal.

- class WaitObjects

  A set of wait objects.

- class PylonAutoInitTerm

  helperclass to automagically call PylonInitialize and PylonTerminate in constructor and destructor

- class CPylonDeviceProxy

  A Handle class (aka Envelope class) for IPylonDevice implementations.

- class GrabResult

  A grab result that combines the used image buffer and status information.

- class EventResult

  An event result.

- interface IStreamGrabber

  Interface to an (input) data stream.

- class CStreamGrabberProxy

  A Handle class (aka Envelope class) for the IStreamGrabber interface.

- class CLock

  Implementation of lock protecting a critical section.

- class AutoLock

  a scopelevel lock class

- class LockableObject

  Base class the makes the Object lockable.

- class CTlFactory

  the Transport Layer Factory

- class SystemWaitObject
- class WaitObjectSet
- class CBasler1394Camera

  DRAFT: Basler generic GigEVision and IIDC camera interface.

- class CBasler1394DeviceInfo

  FireWire specific Device Info object. Does not support SerialNumber property.

- class CBasler1394EventGrabber

  Interface to the FireWire event parameters.

- class CBasler1394StreamGrabber

  Interface to the FireWire stream parameters.

- class CBasler1394Tl

  The FireWire specific implemenation of a TLFactory.

- class CBaslerGigECamera

  Basler GigEVision camera interface.

- class CBaslerGigEDeviceInfo

  Implementation of the GiGE specific device info object.

- class CBaslerGigEEventGrabber

  Interface to the GigE event parameters.

- class CBaslerGigEStreamGrabber

  Interface to the GigE specific stream parameters.

- class CBaslerGigETLParams

  Interface to the GigE transport layer specific parameters.

Namespaces

- namespace Key

  Contains keys used in info classes.

Typedefs

- typedef Pylon::Callback1< IPylonDevice * > DeviceCallback

  callback type with a IPylonDevice as parameter

- typedef void * DeviceCallbackHandle

  opaque device callback object

- typedef GenICam::gcstring String_t

  Pylon's string definition.

- typedef GenICam::gcstring_vector StringList_t

  Pylon's string list definition.

- typedef void * StreamBufferHandle

  This opaque type represents a registered buffer.

Enumerations

- enum AccessMode { Control = 0x1, Stream = 0x3, Event = 0x4, Exclusive = 0x5 }

  The available access modes when opening a camera object.

- enum GrabStatus { ,
  Idle, Queued, Grabbed, Canceled,
  Failed }

  Possible grab status values.

- enum PayloadType

  Possible payload type values.

- enum WaitResult

Functions

- template<class Function, class Callback, class P> Callback make_FunctionCallback (Function function)

  make a new callback object for C functions

- template<class Client, class Member, class Callback, class P> Callback make_MemberFunctionCallback (Client &client, Member member)

  make a new callback object for member functions

- AccessModeSet operator+ (AccessMode lhs, AccessMode rhs)

  Creates a set containing lhs and rhs operands.

- AccessModeSet operator| (AccessMode lhs, AccessMode rhs)

  Creates a set containing lhs and rhs operands.

- AccessModeSet operator+ (const AccessModeSet &lhs, AccessMode rhs)

  Adds the operand rhs to the set lhs.

- AccessModeSet operator| (const AccessModeSet &lhs, AccessMode rhs)

  Adds the operand rhs to the set lhs.

- template<class Function> DeviceCallbackHandle RegisterRemovalCallback (IPylonDevice *pDevice, Function f)

  Register a C-function as a removal callback.

- template<class Client, class Member> DeviceCallbackHandle RegisterRemovalCallback (IPylonDevice *pDevice, Client &c, Member m)

  Register a C++-member function as removal callback.

- int BitPerPixel (enum PixelType pixelType)
- int PixelSize (enum PixelType pixelType)
- enum PixelColorFilter GetPixelColorFilter (enum PixelType pixelType)
- template<> unsigned WaitForObject (const SystemWaitObject< HANDLE > &, unsigned)

  Wait for a single object to get signalled.

- void _cdecl PylonInitialize ()

  Initializes the pylon runtime system.

- void _cdecl PylonTerminate (bool ShutDownLogging=true)

  Frees resources allocated by the pylon runtime system.

- template<class UYVY_type> void ConvertToRGBA (struct Pylon::SRGBAPixel *pDest, const unsigned char *pSrc, const struct SImageFormat &imageFormat, int destLinePitch, bool swapImage)
- void ConvertToRGB (BYTE *pDest, const BYTE *pSrc, const struct SImageFormat &imageFormat, int destLinePitch, bool swapImage)

3.1.1 Detailed Description

Contains definitions of Pylon related types.

3.1.2 Enumeration Type Documentation

3.1.2.1 enum Pylon::AccessMode

The available access modes when opening a camera object.

Enumerator:

Control access the control and status registers

Stream access a streaming data channel

Event access the event data channel

Exclusive exclusive access to the device

3.1.2.2 enum Pylon::GrabStatus

Possible grab status values.

Enumerator:

Idle currently not used

Queued in the input queue

Grabbed filled with data

Canceled request was canceled

Failed request failed

3.1.2.3 enum Pylon::WaitResult

Predefined result of wait functions

3.1.3 Function Documentation

3.1.3.1 AccessModeSet Pylon::operator+ (AccessMode *lhs*, AccessMode *rhs*)

Creates a set containing lhs and rhs operands.

Parameters:

lhs left operand

rhs right operand

Returns:

returns an AccessModeSet containing both operands

3.1.3.2 AccessModeSet Pylon::operator+ (const AccessModeSet & *lhs*, AccessMode *rhs*)

Adds the operand rhs to the set lhs.

Parameters:

lhs a set of bits.

rhs the additional bit

3.1.3.3 static int Pylon::BitPerPixel (enum PixelType *pixelType*) `[inline]`

Returns the bits per pixel as used by the image data stream (transport layer) NOTICE: This is not the PixelSize (bits per pixel) as defined by the dynamic range of the camera. (Therefore no(!) plural BitsPer..)

3.1.3.4 static int Pylon::PixelSize (enum PixelType *pixelType*) `[inline]`

Returns the PixelSize bits per pixel as given by the maximal dynamic range of the camera. NOTICE: This is not the BitPerPixel of the transport layer.

3.1.3.5 static enum PixelColorFilter Pylon::GetPixelColorFilter (enum PixelType *pixelType*) `[inline]`

Returns the color filter type NOTICE: This is not the PixelSize (bits per pixel) as defined by the dynamic range of the camera. (Therefore no(!) plural BitsPer..)

3.1.3.6 void _cdecl Pylon::PylonInitialize ()

Initializes the pylon runtime system.

You must call PylonInitialize before calling any other pylon functions. When finished you must call PylonTerminate to free up all resources used by pylon.

You can use the helperclass PylonAutoInitTerm to let the compiler call PylonInitialze and PylonTerminate.

Just create a local object on the stack in your main function and the constructor and destructor will call the functions. See PylonAutoInitTerm for a sample.

For MFC applications we recommend to call PylonInitialize and PylonTerminate in the application's InitInstance() and ExitInstance() methods. This prevents the MFC runtime from reporting a huge number of pretended memory leaks.

3.1.3.7 void _cdecl Pylon::PylonTerminate (bool *ShutDownLogging* = `true`)

Frees resources allocated by the pylon runtime system.

Call this function before terminating the application. Don't use any pylon methods or pylon objects after having called PylonTerminate().

3.1.3.8 template<class UYVY_type> void Pylon::ConvertToRGBA (struct SRGBAPixel * *pDest*, const unsigned char * *pSrc*, const struct SImageFormat & *imageFormat*, int *destLinePitch*, bool *swapImage*)

Conversion of YUV422 pixel data into RGB (CCIR 601)

R = (Y - 16) * 1.164 + (V-128) * 1.596 G = (Y - 16) * 1.164 - (U-128) * 0.391 - (V-128) * 0.813 B = (Y - 16) * 1.164 + (U-128) * 2.018

The formulas are approximated by Look up Tables sc_LUT_* in file PixelFormatConverterYUV422_LUTs.cpp

R = (298 * (Y-16) + 409 * (V-128) + 128) >> 8 G = (298 * (Y-16) - 100 * (U-128) - 208 * (V-128) + 128) >> 8 B = (298 * (Y-16) + 516 * (U-128) + 128) >> 8

The template allows for implementation of Version UYVY and YUYV and more permutations

Parameters:

pDest Pointer to a buffer the RGBA data will be written to, OUT

pSrc Pointer to the pixel data to be converted, IN

imageFormat Size of image (in pixel)

destLinePitch destination buffer's line pitch in bytes

3.1.3.9 void Pylon::ConvertToRGB (BYTE ∗ *pDest*, const BYTE ∗ *pSrc*, const struct SImageFormat & *imageFormat*, int *destLinePitch*, bool *swapImage*)

Conversion of YUV422 pixel data into RGB (CCIR 601)

R = (Y - 16) ∗ 1.164 + (V-128) ∗ 1.596 G = (Y - 16) ∗ 1.164 - (U-128) ∗ 0.391 - (V-128) ∗ 0.813 B = (Y - 16) ∗ 1.164 + (U-128) ∗ 2.018

The formulas are approximated by Look up Table sc_LUT_∗

R = (298 ∗ (Y-16) + 409 ∗ (V-128) + 128) >> 8 G = (298 ∗ (Y-16) - 100 ∗ (U-128) - 208 ∗ (V-128) + 128) >> 8 B = (298 ∗ (Y-16) + 516 ∗ (U-128) + 128) >> 8

Parameters:

pDest Pointer to a buffer the RGB data will be written to, OUT

pSrc Pointer to the pixel data to be converted, IN

imageFormat Size of image (in pixel)

destLinePitch destination buffer's line pitch in bytes

3.2 Pylon::Key Namespace Reference

Contains keys used in info classes.

Variables

- const String_t InternalNameKey ("InternalName")

  key for driver specific device name

3.2.1 Detailed Description

Contains keys used in info classes.

Chapter 4

Class Documentation

4.1 GenApi::AttachStatistics_t Struct Reference

Public Attributes

- int NumChunkPorts

  Number of chunk ports found in the node map.

- int NumChunks

  Number of chunks found in the buffer.

- int NumAttachedChunks

  Number of chunks from the buffer attached to a chunk port.

4.1.1 Detailed Description

Delivers information about the attached chunks and nodes

4.2 GenApi::CChunkAdapter Class Reference

Connects a chunked buffer to a node map.

Inherited by GenApi::CChunkAdapterDcam, and GenApi::CChunkAdapterGEV.

Public Member Functions

- virtual ~CChunkAdapter ()

  Destructor.

- void AttachNodeMap (INodeMap *pNodeMap)

  Attaches to a node map and retrieves the chunk ports.

- void DetachNodeMap ()

  Detaches from the node map.

- virtual bool CheckBufferLayout (uint8_t *pBuffer, int64_t BufferLength)=0

  Checks if a buffer contains chunks in a known format.

- virtual void AttachBuffer (uint8_t *pBuffer, int64_t BufferLength, AttachStatistics_t *pAttachStatistics=NULL)=0

  Attaches a buffer to the matching ChunkPort.

- void DetachBuffer ()

  Detaches a buffer.

- void UpdateBuffer (uint8_t *pBaseAddress)

  Updates the base address of the buffer.

Protected Member Functions

- CChunkAdapter (INodeMap *pNodeMap=NULL)

  Serves as default constructor.

Protected Attributes

- std::vector< CChunkPort * > m_pChunkPorts

  An array of attached ChunkPorts.

4.2.1 Detailed Description

Connects a chunked buffer to a node map.

4.2.2 Member Function Documentation

4.2.2.1 virtual void GenApi::CChunkAdapter::AttachBuffer (uint8_t * *pBuffer*, int64_t *BufferLength*, AttachStatistics_t * *pAttachStatistics* = NULL) [pure virtual]

Attaches a buffer to the matching ChunkPort.

Implement that for a specific buffer layout

Implemented in GenApi::CChunkAdapterDcam, and GenApi::CChunkAdapterGEV.

4.2.2.2 virtual bool GenApi::CChunkAdapter::CheckBufferLayout (uint8_t * *pBuffer*, int64_t *BufferLength*) [pure virtual]

Checks if a buffer contains chunks in a known format.

Implement that for a specific buffer layout

Implemented in GenApi::CChunkAdapterDcam, and GenApi::CChunkAdapterGEV.

4.3 GenApi::CChunkAdapterDcam Class Reference

Connects a chunked DCAM buffer to a node map.

Inherits GenApi::CChunkAdapter.

Public Member Functions

- CChunkAdapterDcam (INodeMap ∗pNodeMap=NULL)

  Constructor.

- virtual ∼CChunkAdapterDcam ()

  Destructor.

- virtual bool CheckBufferLayout (uint8_t ∗pBuffer, int64_t BufferLength)

  Checks if a buffer contains chunks in a known format.

- virtual void AttachBuffer (uint8_t ∗pBuffer, int64_t BufferLength, AttachStatistics_t ∗pAttach-Statistics=NULL)

  Attaches a buffer to the matching ChunkPort.

- bool HasCRC (uint8_t ∗pBuffer, int64_t BufferLength)

  Checks if buffer has a CRC attached.

- bool CheckCRC (uint8_t ∗pBuffer, int64_t BufferLength)

  Checks CRC sum of buffer.

- void AttachNodeMap (INodeMap ∗pNodeMap)

  Attaches to a node map and retrieves the chunk ports.

- void DetachNodeMap ()

  Detaches from the node map.

- void DetachBuffer ()

  Detaches a buffer.

- void UpdateBuffer (uint8_t ∗pBaseAddress)

  Updates the base address of the buffer.

Protected Member Functions

- unsigned long CRC16 (const unsigned char ∗pData, unsigned long nbyLength)

  Checks the CRC of a buffer.

- bool CheckBuffer (const unsigned char ∗pData, unsigned long nbyLength)

  Checks the CRC of a buffer.

Protected Attributes

- std::vector< CChunkPort * > m_pChunkPorts

  An array of attached ChunkPorts.

4.3.1 Detailed Description

Connects a chunked DCAM buffer to a node map.

4.4 GenApi::CChunkAdapterGEV Class Reference

Connects a chunked DCAM buffer to a node map.

Inherits GenApi::CChunkAdapter.

Public Member Functions

- CChunkAdapterGEV (INodeMap *pNodeMap=NULL)

  Constructor.

- virtual ~CChunkAdapterGEV ()

  Destructor.

- virtual bool CheckBufferLayout (uint8_t *pBuffer, int64_t BufferLength)

  Checks if a buffer contains chunks in a known format.

- virtual void AttachBuffer (uint8_t *pBuffer, int64_t BufferLength, AttachStatistics_t *pAttachStatistics=NULL)

  Attaches a buffer to the matching ChunkPort.

- void AttachNodeMap (INodeMap *pNodeMap)

  Attaches to a node map and retrieves the chunk ports.

- void DetachNodeMap ()

  Detaches from the node map.

- void DetachBuffer ()

  Detaches a buffer.

- void UpdateBuffer (uint8_t *pBaseAddress)

  Updates the base address of the buffer.

Protected Attributes

- std::vector< CChunkPort * > m_pChunkPorts

  An array of attached ChunkPorts.

4.4.1 Detailed Description

Connects a chunked DCAM buffer to a node map.

4.5 GenApi::CNodeMapRef Class Reference

Smartpointer for NodeMaps with create function.

Public Member Functions

- CNodeMapRef (GenICam::gcstring DeviceName="Device")
  Constructor.
- virtual ~CNodeMapRef ()
  Destructor.
- void _LoadDLL (GenICam::gcstring VendorName, GenICam::gcstring ModelName)
  Creates the object from a DLL whose name is deduced from vendor anf model name.
- void _LoadDLL (GenICam::gcstring FileName)
  Creates the object from a DLL with given file name.
- void _LoadXMLFromFile (GenICam::gcstring VendorName, GenICam::gcstring ModelName)
  Creates the object from a XML file whose name is deduced from vendor anf model name.
- void _LoadXMLFromFile (GenICam::gcstring FileName)
  Creates the object from a XML file with given file name.
- void _LoadXMLFromString (const GenICam::gcstring &XMLData)
  Creates the object from XML data given in a string.
- virtual GenICam::gcstring _GetDeviceName ()
  Get device name.
- virtual void _Poll (int64_t ElapsedTime)
  Fires nodes which have a polling time.
- void _Destroy ()
  Destroys the node map.
- virtual void _GetNodes (NodeList_t &Nodes) const
  Retrieves all nodes in the node map.
- virtual INode * _GetNode (const GenICam::gcstring &key) const
  Retrieves the node from the central map by name.
- virtual void _InvalidateNodes () const
  Invalidates all nodes.
- virtual bool _Connect (IPort *pPort, const GenICam::gcstring &PortName) const
  Connects a port to a port node with given name.
- virtual bool _Connect (IPort *pPort) const
  Connects a port to the standard port "Device".

Static Public Member Functions

- static bool _ClearXMLCache ()

  Clears the cache of the camera description files.

Public Attributes

- CPointer< INodeMap, INodeMap > _Ptr

  Pointer to the NodeMap.

Protected Member Functions

- GenICam::gcstring _MakeDLLFileName (GenICam::gcstring VendorName, GenICam::gcstring ModelName)

  Creates the DLL filename.

- void _InternalLoadDLL (GenICam::gcstring FileName)

  Creates the object from a DLL with given file name without initializing.

- virtual void _Initialize ()

  Initializes the references.

4.5.1 Detailed Description

Smartpointer for NodeMaps with create function.

4.6 GenApi::CPointer< T, B > Class Template Reference

Encapsulates a GenApi pointer dealing with the dynamic_cast automatically.

Public Member Functions

- CPointer (void) throw ()

  Default constructor.

- CPointer (B *pB)

  Constructor from INode pointer type.

- void operator= (B *pB)

  Assign INode Pointer.

- operator T * (void) const

  Dereferencing.

- T & operator * (void) const

  Dereferencing.

- T & operator() (void) const

  Dereferencing.

- T * operator → (void) const

  Dereferencing.

- bool IsValid () const throw ()

  true if the pointer is valid

- operator bool (void) const throw ()

  true if the pointer is valid

- bool operator== (T *pT) const throw ()

  pointer equal

- bool operator== (int nMustBeNull) const

  pointer equal

Protected Attributes

- T * m_pT

  Underlying raw pointer.

4.6.1 Detailed Description

template<class T, class B = IBase> class GenApi::CPointer< T, B >

Encapsulates a GenApi pointer dealing with the dynamic_cast automatically.

4.7 GenApi::Function_NodeCallbackBody< Function > Class Template Reference

Container for a function pointer.

Inherits GenApi::NodeCallbackBase.

Public Member Functions

- Function_NodeCallbackBody (const Function &function)

  Constructor.

- void operator() (INode *pNode) const

  execute operation: call the function

- Function_NodeCallbackBody< Function > * clone () const

  virtual copy constructor

4.7.1 Detailed Description

template<class Function> class GenApi::Function_NodeCallbackBody< Function >

Container for a function pointer.

4.8 GenApi::GVCP_CHUNK_TRAILER Struct Reference

header of a GVCP request packet

4.8.1 Detailed Description

header of a GVCP request packet

4.9 GenApi::IBase Struct Reference

Base interface common to all nodes.

Inherited by GenApi::INode `[virtual]`, GenApi::IPort `[virtual]`, GenApi::ISelector `[virtual]`, and GenApi::IValue `[virtual]`.

Public Member Functions

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.9.1 Detailed Description

Base interface common to all nodes.

4.10 GenApi::IBoolean Struct Reference

Interface for Boolean properties.

Inherits GenApi::IValue.

Public Member Functions

- virtual void SetValue (bool Value, bool Verify=true)=0

  Set node value.

- virtual void operator= (bool Value)

  Set node value.

- virtual bool GetValue () const =0

  Get node value.

- virtual bool operator() () const

  Get node value.

- virtual INode * GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.10.1 Detailed Description

Interface for Boolean properties.

4.11 GenApi::ICategory Struct Reference

Gives access to a category node.

Inherits GenApi::IValue.

Public Member Functions

- virtual void GetFeatures (FeatureList_t &Features) const =0

  Get all features of the category (including sub-categories).

- virtual INode * GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.11.1 Detailed Description

Gives access to a category node.

4.12 GenApi::ICommand Struct Reference

Inherits GenApi::IValue.

Public Member Functions

- virtual void Execute ()=0

  Execute the command.

- virtual void operator() ()=0

  Execute the command.

- virtual bool IsDone () const =0

  Query whether the command is executed.

- virtual INode * GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.12.1 Detailed Description

Interface for command like properties

4.13 GenApi::IEnumEntry Struct Reference

Interface of single enum value.

Inherits GenApi::IValue.

Public Member Functions

- virtual int64_t GetValue ()=0

  Get numeric enum value.

- virtual GenICam::gcstring GetSymbolic () const =0

  Get symbolic enum value.

- virtual INode * GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.13.1 Detailed Description

Interface of single enum value.

Maps of Enum Values to symbolic values

4.14 GenApi::IEnumeration Struct Reference

Interface for enumeration properties.

Inherits GenApi::IValue.

Inherited by GenApi::IEnumerationT< EnumT >, GenApi::IEnumerationT< Basler_GigECameraParams::AcquisitionModeEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::BalanceRatioSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::BlackLevelSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::ChunkPixelFormatEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::ChunkSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::DeviceScanTypeEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::EventNotificationEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::EventSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::ExposureModeEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::FileCMDEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::GainSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::GevCCPEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::GevInterfaceSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::GevStreamChannelSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::LegacyBinningVerticalEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::LineFormatEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::LineModeEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::LineSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::LineSourceEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::LUTSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::ParameterSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::PixelCodingEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::PixelColorFilterEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::PixelFormatEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::PixelSizeEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::SpatialCorrectionStartingLineEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TemperatureSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TestImageSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TimerSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TimerTriggerActivationEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TimerTriggerSourceEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TriggerActivationEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TriggerModeEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TriggerSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::TriggerSourceEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::UserOutputSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::UserSetDefaultSelectorEnums >, GenApi::IEnumerationT< Basler_GigECameraParams::UserSetSelectorEnums >, GenApi::IEnumerationT< Basler_GigEStreamParams::StatusEnums >, GenApi::IEnumerationT< Basler_GigETLParams::RequestedTypeEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::AcquisitionModeEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::BalanceRatioSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::BlackLevelSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::ChunkSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::DeviceScanTypeEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::EventNotificationEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::EventSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::ExposureModeEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::FileCMDEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::GainSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LegacyBinningVerticalEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineFormatEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineModeEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineSourceEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LUTSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::ParameterSelectorEnums

>, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::PixelColorFilterEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::PixelFormatEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::PixelSizeEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TemperatureSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TestImageSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TimerSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TimerTriggerActivationEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TimerTriggerSourceEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerActivationEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerModeEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerSourceEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::UserOutputSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::UserSetDefaultSelectorEnums >, GenApi::IEnumerationT< Basler_IIDC1394CameraParams::UserSetSelectorEnums >, and GenApi::IEnumerationT< Basler_IIDC1394StreamParams::StatusEnums >.

Public Member Functions

- virtual void GetSymbolics (StringList_t &Symbolics)=0

  Get list of symbolic Values.

- virtual void GetEntries (NodeList_t &Entries)=0

  Get list of entry nodes.

- virtual IEnumeration & operator= (const GenICam::gcstring &ValueStr)=0

  Set string node value.

- virtual void SetIntValue (int64_t Value, bool Verify=true)=0

  Set integer node value.

- virtual GenICam::gcstring operator ∗ ()=0

  Get string node value.

- virtual int64_t GetIntValue (bool Verify=false)=0

  Get integer node value.

- virtual IEnumEntry ∗ GetEntryByName (const GenICam::gcstring &EntryName)=0

  Get an entry node by name.

- virtual INode ∗ GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.14.1 Detailed Description

Interface for enumeration properties.

4.15 GenApi::IEnumerationT< EnumT > Struct Template Reference

Interface for enumeration properties.

Inherits GenApi::IEnumeration.

Inherited by GenApi::CEnumerationTRef< EnumT >.

Public Member Functions

- virtual void SetValue (EnumT Value, bool Verify=true)=0

  Set node value.

- virtual IEnumeration & operator= (EnumT Value)=0

  Set node value.

- virtual EnumT GetValue (bool Verify=false)=0

  Get node value.

- virtual EnumT operator() ()=0

  Get node value.

- virtual IEnumeration & operator= (const GenICam::gcstring &ValueStr)=0

  Set node value.

- virtual void GetSymbolics (StringList_t &Symbolics)=0

  Get list of symbolic Values.

- virtual void GetEntries (NodeList_t &Entries)=0

  Get list of entry nodes.

- virtual void SetIntValue (int64_t Value, bool Verify=true)=0

  Set integer node value.

- virtual GenICam::gcstring operator * ()=0

  Get string node value.

- virtual int64_t GetIntValue (bool Verify=false)=0

  Get integer node value.

- virtual IEnumEntry * GetEntryByName (const GenICam::gcstring &EntryName)=0

  Get an entry node by name.

- virtual INode * GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.15.1 Detailed Description

template<typename EnumT> struct GenApi::IEnumerationT< EnumT >

Interface for enumeration properties.

4.15.2 Member Function Documentation

4.15.2.1 template<typename EnumT> virtual IEnumeration& GenApi::IEnumerationT< EnumT >::operator= (const GenICam::gcstring & *ValueStr*) `[pure virtual]`

Set node value.

Note : the operator= is not inherited thus the operator= versions from IEnumeration must be implemented again

Implements GenApi::IEnumeration.

4.16 GenApi::IFloat Struct Reference

Interface for float properties.

Inherits GenApi::IValue.

Public Member Functions

- virtual void SetValue (double Value, bool Verify=true)=0
  Set node value.
- virtual IFloat & operator= (double Value)=0
  Set node value.
- virtual double GetValue (bool Verify=false)=0
  Get node value.
- virtual double operator() ()=0
  Get node value.
- virtual double operator ∗ ()=0
  Get node value.
- virtual double GetMin ()=0
  Get minimum value allowed.
- virtual double GetMax ()=0
  Get maximum value allowed.
- virtual ERepresentation GetRepresentation ()=0
  Get recommended representation.
- virtual GenICam::gcstring GetUnit () const =0
  Get the physical unit name.
- virtual INode ∗ GetNode ()
  Get the INode interface of the node.
- virtual GenICam::gcstring ToString (bool Verify=false)=0
  Get content of the node as string.
- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0
  Set content of the node as string.
- virtual EAccessMode GetAccessMode () const =0
  Get the access mode of the node.

4.16.1 Detailed Description

Interface for float properties.

4.17 GenApi::IInteger Struct Reference

Interface for integer properties.

Inherits GenApi::IValue.

Public Member Functions

- virtual void SetValue (int64_t Value, bool Verify=true)=0
  Set node value.
- virtual IInteger & operator= (int64_t Value)=0
  Set node value.
- virtual int64_t GetValue (bool Verify=false)=0
  Get node value.
- virtual int64_t operator() ()=0
  Get node value.
- virtual int64_t operator ∗ ()=0
  Get node value.
- virtual int64_t GetMin ()=0
  Get minimum value allowed.
- virtual int64_t GetMax ()=0
  Get maximum value allowed.
- virtual int64_t GetInc ()=0
  Get increment.
- virtual ERepresentation GetRepresentation ()=0
  Get recommended representation.
- virtual INode ∗ GetNode ()
  Get the INode interface of the node.
- virtual GenICam::gcstring ToString (bool Verify=false)=0
  Get content of the node as string.
- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0
  Set content of the node as string.
- virtual EAccessMode GetAccessMode () const =0
  Get the access mode of the node.

4.17.1 Detailed Description

Interface for integer properties.

4.18 GenApi::INode Struct Reference

Interface common to all nodes.

Inherits GenApi::IBase.

Public Member Functions

- virtual GenICam::gcstring GetName (bool FullQualified=false) const =0

  Get node name.

- virtual GenApi::ENameSpace GetNameSpace () const =0

  Get name space.

- virtual EVisibility GetVisibility () const =0

  Get the recommended visibility of the node.

- virtual void InvalidateNode ()=0

  Set/reset changed status.

- virtual bool IsCachable () const =0

  Is the node value cachable.

- virtual ECachingMode GetCachingMode () const =0

  Get Caching Mode.

- virtual int64_t GetPollingTime () const =0

  recommended polling time (for not cachable nodes)

- virtual GenICam::gcstring GetToolTip () const =0

  Get a short description of the node.

- virtual GenICam::gcstring GetDescription () const =0

  Get a long description of the node.

- virtual GenICam::gcstring GetDisplayName () const =0

  Get a name string for display.

- virtual void GetChildren (NodeList_t &Children) const =0

  Get all children of the node.

- virtual CallbackHandleType RegisterCallback (NodeCallback &)=0

  Register change callback.

- virtual CallbackHandleType DeregisterCallback (CallbackHandleType)=0

  Deregister change callback.

- virtual INodeMap * GetNodeMap () const =0

  Retrieves the central node map.

- virtual GenICam::gcstring GetEventID () const =0

  Get the EventId of the node.

- virtual bool IsStreamable () const =0

  True if the node is streamable.

- virtual void GetPropertyNames (GenICam::gcstring_vector &PropertyNames) const =0

  Returns a list of the names all properties set during initialization.

- virtual bool GetProperty (const GenICam::gcstring &PropertyName, GenICam::gcstring &ValueStr, GenICam::gcstring &AttributeStr)=0

  Retrieves a property plus an additional attribute by name.

- virtual void ImposeAccessMode (EAccessMode ImposedAccessMode)=0

  Imposes an access mode to the natural access mode of the node.

- virtual void ImposeVisibility (EVisibility ImposedVisibility)=0

  Imposes a visibility to the natural visibility of the node.

- virtual INode * GetAlias () const =0

  Retrieves the a node which describes the same feature in a different way.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.18.1 Detailed Description

Interface common to all nodes.

4.18.2 Member Function Documentation

4.18.2.1 virtual bool GenApi::INode::GetProperty (const GenICam::gcstring & *PropertyName*, GenICam::gcstring & *ValueStr*, GenICam::gcstring & *AttributeStr*) `[pure virtual]`

Retrieves a property plus an additional attribute by name.

If a property has multiple values/attribute they come with Tabs as delimiters

4.19 GenApi::INodeMap Struct Reference

Interface to access the node map.

Public Member Functions

- virtual void GetNodes (NodeList_t &Nodes) const =0

  Retrieves all nodes in the node map.

- virtual INode ∗ GetNode (const GenICam::gcstring &Name) const =0

  Retrieves the node from the central map by Name.

- virtual void InvalidateNodes () const =0

  Invalidates all nodes.

- virtual bool Connect (IPort ∗pPort, const GenICam::gcstring &PortName) const =0

  Connects a port to a port node with given name.

- virtual bool Connect (IPort ∗pPort) const =0

  Connects a port to the standard port "Device".

- virtual GenICam::gcstring GetDeviceName ()=0

  Get device name.

- virtual void Poll (int64_t ElapsedTime)=0

  Fires nodes which have a polling time.

4.19.1 Detailed Description

Interface to access the node map.

4.19.2 Member Function Documentation

4.19.2.1 virtual GenICam::gcstring GenApi::INodeMap::GetDeviceName () [pure virtual]

Get device name.

The device name identifies a device instance, e.g. for debuggin purposes. The default ist "Device".

4.20 GenApi::IPort Struct Reference

Interface for ports.

Inherits GenApi::IBase.

Public Member Functions

- virtual void Read (void ∗pBuffer, int64_t Address, int64_t Length)=0

  Reads a chunk of bytes from the port.

- virtual void Write (const void ∗pBuffer, int64_t Address, int64_t Length)=0

  Writes a chunk of bytes to the port.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.20.1 Detailed Description

Interface for ports.

4.21 GenApi::IRegister Struct Reference

Interface for registers.

Inherits GenApi::IValue.

Public Member Functions

- virtual void Set (const uint8_t ∗pBuffer, int64_t Length)=0

  Set the register's contents.

- virtual void Get (uint8_t ∗pBuffer, int64_t Length)=0

  Fills a buffer with the register's contents.

- virtual int64_t GetLength ()=0

  Retrieves the Length of the register [Bytes].

- virtual int64_t GetAddress ()=0

  Retrieves the Address of the register.

- virtual INode ∗ GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.21.1 Detailed Description

Interface for registers.

4.22 GenApi::ISelector Struct Reference

Interface for groups of features selected by a single one.

Inherits GenApi::IBase.

Public Member Functions

- virtual bool IsSelector () const =0

  true iff this feature selects a group of features

- virtual void GetSelectedFeatures (FeatureList_t &) const =0

  retrieve the group of selected features

- virtual void GetSelectingFeatures (FeatureList_t &) const =0

  retrieve the group of features selecting this node

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.22.1 Detailed Description

Interface for groups of features selected by a single one.

4.23 GenApi::IString Struct Reference

Interface for string properties.

Inherits GenApi::IValue.

Public Member Functions

- virtual void SetValue (const GenICam::gcstring &Value, bool Verify=true)=0

  Set node value.

- virtual IString & operator= (const GenICam::gcstring &Value)=0

  Set node value.

- virtual GenICam::gcstring GetValue (bool Verify=false)=0

  Get node value.

- virtual GenICam::gcstring operator() ()=0

  Get node value.

- virtual GenICam::gcstring operator ∗ ()=0

  Get node value.

- virtual INode ∗ GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.23.1 Detailed Description

Interface for string properties.

4.24 GenApi::IValue Struct Reference

Interface for value properties.

Inherits GenApi::IBase.

Inherited by GenApi::IBoolean [virtual], GenApi::ICategory [virtual], GenApi::ICommand [virtual], GenApi::IEnumEntry [virtual], GenApi::IEnumeration [virtual], GenApi::IFloat [virtual], GenApi::IInteger [virtual], GenApi::IRegister [virtual], and GenApi::IString [virtual].

Public Member Functions

- virtual INode * GetNode ()

  Get the INode interface of the node.

- virtual GenICam::gcstring ToString (bool Verify=false)=0

  Get content of the node as string.

- virtual void FromString (const GenICam::gcstring &ValueStr, bool Verify=true)=0

  Set content of the node as string.

- virtual EAccessMode GetAccessMode () const =0

  Get the access mode of the node.

4.24.1 Detailed Description

Interface for value properties.

4.25 GenApi::Member_NodeCallbackBody< Client, Member > Class Template Reference

Container for a member function pointer.

Inherits GenApi::NodeCallbackBase.

Public Types

- typedef void(Client::*) PMEMBERFUNC (INode *)

  Member function type.

Public Member Functions

- Member_NodeCallbackBody (Client &client, Member member)

  Constructor.

- void operator() (INode *pNode) const

  execute operation

- Member_NodeCallbackBody< Client, Member > * clone () const

  virtual copy constructor

4.25.1 Detailed Description

template<class Client, class Member> class GenApi::Member_NodeCallbackBody< Client, Member >

Container for a member function pointer.

4.26 GenApi::NodeCallback Class Reference

Callback object for callback taking an INode pointer.

Inherits Callback1.

4.26.1 Detailed Description

Callback object for callback taking an INode pointer.

4.27 GenApi::NodeCallbackBase Class Reference

callback body instance for INode pointers

Inherits Base_Callback1Body.

Inherited by GenApi::Function_NodeCallbackBody< Function >, and GenApi::Member_NodeCallback-Body< Client, Member >.

4.27.1 Detailed Description

callback body instance for INode pointers

4.28 GenICam::AccessException Class Reference

Exception to be thrown to indicate an access error.

Inherits GenericException.

4.28.1 Detailed Description

Exception to be thrown to indicate an access error.

4.29 GenICam::DynamicCastException Class Reference

Exception to be thrown to indicate the the result of a dynamic cast was zero.

Inherits GenericException.

4.29.1 Detailed Description

Exception to be thrown to indicate the the result of a dynamic cast was zero.

4.30 GenICam::ExceptionReporter< E > Class Template Reference

printf like creation of exceptions

4.30.1 Detailed Description

template<class E> class GenICam::ExceptionReporter< E >

printf like creation of exceptions

4.31 GenICam::gcstring Class Reference

A string class which is a clone of std::string.

4.31.1 Detailed Description

A string class which is a clone of std::string.

4.32 GenICam::GenericException Class Reference

GenICam's exception class.

Inherits exception.

Public Member Functions

- GenericException (const char *description, const char *sourceFileName, unsigned int sourceLine)

  Constructor.

- const char * GetDescription () const throw ()

  Get error description.

- const char * GetSourceFileName () const throw ()

  Get filename in which the error occurred.

- unsigned int GetSourceLine () const throw ()

  Get line number at which the error occurred.

- virtual const char * what () const throw ()

  Get error description (overwrite from std:exception).

Protected Attributes

- unsigned int m_SourceLine

  Line number at which the error occurred.

- gcstring m_SourceFileName

  Filename in which the error occurred.

- gcstring m_Description

  Error description.

- gcstring m_What

  Complete error message, including file name and line number.

4.32.1 Detailed Description

GenICam's exception class.

4.33 GenICam::InvalidArgumentException Class Reference

Exception fired if an argument is invalid.

Inherits GenericException.

4.33.1 Detailed Description

Exception fired if an argument is invalid.

4.34 GenICam::LogicalErrorException Class Reference

Exception to be thrown to indicate logical errors in program flow.

Inherits GenericException.

4.34.1 Detailed Description

Exception to be thrown to indicate logical errors in program flow.

4.35 GenICam::OutOfRangeException Class Reference

Exception fired if an argument is out of range.

Inherits GenericException.

4.35.1 Detailed Description

Exception fired if an argument is out of range.

4.36 GenICam::PropertyException Class Reference

Exception fired if a property access fails.

Inherits GenericException.

4.36.1 Detailed Description

Exception fired if a property access fails.

4.37 GenICam::RuntimeException Class Reference

Runtime exception.

Inherits GenericException.

4.37.1 Detailed Description

Runtime exception.

4.38 GenICam::TimeoutException Class Reference

Exception to be thrown to indicate an access error.

Inherits GenericException.

4.38.1 Detailed Description

Exception to be thrown to indicate an access error.

4.39 Pylon::AccessModeSet Class Reference

Collection of AccessMode bits.

Inherits bitset.

Public Member Functions

- AccessModeSet ()

  Default constructor creates an empty set.

- AccessModeSet (AccessMode)

  Converts an access mode into a set.

- AccessModeSet (const AccessModeSet &s)

  Copy constructor.

4.39.1 Detailed Description

Collection of AccessMode bits.

This set is used when a device is opened. The combination of different access modes specifies how the device is opened. Not all combinations may be allowed because the device implementations have certain restrictions.

```
AccessModeSet a = Exclusive | Stream;
if (a.test( Exclusive ))
{
    // Exclusive implies Control access, so set it also
    a |= Control;
}
```

See also:

The method of IDevice::Open() uses it to define a default value.
The global operator |(AccessMode lhs, AccessMode rhs) allows to combine two modes to a set.

4.40 Pylon::AlertableWaitObject Class Reference

A wait object that the user may signal.

Inherits Pylon::WaitObject.

Public Member Functions

- AlertableWaitObject (BOOL manualReset=false, BOOL initiallySignalled=false)
  constructor
- AlertableWaitObject (HANDLE h)
  constructor using an existing object
- ~AlertableWaitObject ()
  destructor
- AlertableWaitObject (const AlertableWaitObject &)
  copy constructor
- AlertableWaitObject & operator= (const AlertableWaitObject &)
  assignment operator
- bool IsSignalled () const
  check whether the object is signalled
- void Reset ()
  reset the object to unsignalled state
- void Signal ()
  set the object to signalled state
- operator HANDLE () const
  conversion operator
- bool Wait (unsigned timeout)
  wait for the object to be signalled

4.40.1 Detailed Description

A wait object that the user may signal.

4.41 Pylon::AutoLock Class Reference

a scopelevel lock class

Public Member Functions

- AutoLock (CLock &lock)

  Constructor that acquires the lock.

- ~AutoLock ()

  Destructor that releases the lock.

4.41.1 Detailed Description

a scopelevel lock class

4.42 Pylon::Base_Callback1Body< P1 > Class Template Reference

abstract callback container

Public Member Functions

- virtual ~Base_Callback1Body ()

  destructor

- virtual void operator() (P1) const =0

  do operation with P1 as argument

- virtual Base_Callback1Body< P1 > * clone () const =0

  deep copy

4.42.1 Detailed Description

template<class P1> class Pylon::Base_Callback1Body< P1 >

abstract callback container

4.43 Pylon::Callback1< P1 > Class Template Reference

callback object with one parameter P1

Public Member Functions

- Callback1 (Base_Callback1Body< P1 > *pBody)
  constructor, taking lifetime control of body
- Callback1 (const Callback1< P1 > &callback)
  copy constructor doing deep copy
- ~Callback1 ()
  destructor, destroying body
- Callback1< P1 > & operator= (const Callback1< P1 > &callback)
  assignment operator
- void operator() (P1 p1)
  do operation defined in body

4.43.1 Detailed Description

template<class P1> class Pylon::Callback1< P1 >

callback object with one parameter P1

4.44 Pylon::CBasler1394Camera Class Reference

DRAFT: Basler generic GigEVision and IIDC camera interface.

Inherits Pylon::CPylonDeviceProxy.

Public Member Functions

- virtual ~CBasler1394Camera (void)

  Destructor.

Construction

- CBasler1394Camera ()

  Creates a camera object that is not attached to an pylon device. Use the Attach() method to attach the device.

- CBasler1394Camera (IPylonDevice *)

  Creates a camera object that takes the ownership over an pylon device.

Some smart pointer functionality

- void Attach (IPylonDevice *pDevice, bool takeOwnership=true)

  Attach a the camera object to a pylon device.

Static Public Member Functions

- static String_t DeviceClass ()

  The name of this device class, use this for enumeration.

Public Attributes

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::GainSelectorEnums > & GainSelector

  Selects the gain control to configure. Once a gain control has been selected, all changes to the gain settings will be applied to the selected control.

- GenApi::IInteger & GainRaw

  This is an integer value that sets the selected gain control in device specific units.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::BlackLevelSelectorEnums > & BlackLevelSelector

  Selcts a black level control to configure. Once a black level control has been selected, all changes to the black level settings will be applied to the selected control.

- GenApi::IInteger & BlackLevelRaw

  Sets the value of the selected black level control as an integer.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::BalanceRatioSelectorEnums > & BalanceRatioSelector

  Selects a balance ratio to configure. Once a balance ratio control has been selected, all changes to the balance ratio settings will be applied to the selected control.

- GenApi::IFloat & BalanceRatioAbs

  Sets the value of the selected balance ratio control as a float.

- GenApi::IInteger & BalanceRatioRaw

  Sets the value of the selected balance ratio control as an integer.

- GenApi::IBoolean & GammaEnable

  Enables the gamma correction.

- GenApi::IFloat & Gamma

  This feature is used to perform gamma correction of pixel intensity.

- GenApi::IInteger & DigitalShift

  Sets the value of the selected digital shift control.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::PixelFormatEnums > & PixelFormat

  Sets the format of the pixel data transmitted for acquired images.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::PixelSizeEnums > & PixelSize

  Indicates the depth of the pixel values in the image in bits per pixel.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::PixelColorFilterEnums > & PixelColor-Filter

  Indicates the alignment of the camera's Bayer filter to the pixels in the acquired images.

- GenApi::IInteger & PixelDynamicRangeMin

  Indicates the minimum possible pixel value that could be transferred from the camera.

- GenApi::IInteger & PixelDynamicRangeMax

  Indicates the maximum possible pixel value that could be transferred from the camera.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TestImageSelectorEnums > & Test-ImageSelector

  Selecting a test image from the list will enable the test image.

- GenApi::IInteger & Width

  Sets the width of the area of interest in pixels.

- GenApi::IInteger & Height

  Sets the height of the area of interest in pixels.

- GenApi::IInteger & OffsetX

  Sets the X offset (left offset) of the area of interest in pixels.

- GenApi::IInteger & OffsetY

  Sets the Y offset (top offset) for the area of interest in pixels.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LegacyBinningVerticalEnums > & LegacyBinningVertical

  Sets the vertical binning feature.

- GenApi::IInteger & BinningVertical

  Sets the number of adjacent vertical pixes to be summed.

- GenApi::IInteger & BinningHorizontal

  Sets the number of adjacent horizontal pixes to be summed.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::AcquisitionModeEnums > & AcquisitionMode

  Sets the image acquisition mode.

- GenApi::ICommand & AcquisitionStart

  Starts the acquisition of images.

- GenApi::ICommand & AcquisitionStop

  Stops the acquisition of images.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerSelectorEnums > & TriggerSelector

  Selects the trigger type to configure. Once a trigger type has been selected, all changes to the trigger settings will be applied to the selected trigger.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerModeEnums > & TriggerMode

  Sets the mode for the selected trigger.

- GenApi::ICommand & TriggerSoftware

  Generates a software trigger signal that is used when the trigger source is set to 'software'.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerSourceEnums > & TriggerSource

  Sets the signal source for the selected trigger.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TriggerActivationEnums > & TriggerActivation

  Sets the signal transition needed to activate the selected trigger.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::ExposureModeEnums > & ExposureMode

  Sets the exposure mode.

- GenApi::IFloat & ExposureTimeAbs

  Directly sets the camera's exposure time in microseconds.

- GenApi::IFloat & ExposureTimeBaseAbs

Sets the time base (in microseconds) that is used when the exposure time is set with the 'exposure time raw' setting.

- GenApi::IBoolean & ExposureTimeBaseAbsEnable

  Enables the use of the exposure time base.

- GenApi::IInteger & ExposureTimeRaw

  Sets the 'raw' exposure time. Actual exposure time = raw exposure setting x exposure time base abs setting.

- GenApi::IFloat & AcquisitionLineRateAbs

  Sets the camera's acquisition line rate in lines per second.

- GenApi::IFloat & ResultingLineRateAbs

  Indicates the maximum allowed line acquisition rate (in lines per second) given the current settings for the area of interest, exposure time, and bandwidth.

- GenApi::IBoolean & AcquisitionFrameRateEnable

  Enables setting the camera's acquisition frame rate to a specified value.

- GenApi::IFloat & AcquisitionFrameRateAbs

  If the acquisition frame rate feature is enabled, this value sets the camera's acquisition frame rate in frames per second.

- GenApi::IFloat & ResultingFrameRateAbs

  Indicates the maximum allowed frame acquisition rate (in frames per second) given the current settings for the area of interest, exposure time, and bandwidth.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineSelectorEnums > & LineSelector

  Selects the I/O line to configure. Once a line has been selected, all changes to the line settings will be applied to the selected line.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineModeEnums > & LineMode

  Sets the mode for the selected line.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineFormatEnums > & LineFormat

  Sets the electrical configuration of the selected line.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LineSourceEnums > & LineSource

  Sets the source signal for the selected line (if the selected line is an output).

- GenApi::IBoolean & LineInverter

  Enables the inverter function for the selected line.

- GenApi::IFloat & LineDebouncerTimeAbs

  Sets the absolute value of the selected line debouncer time in microseconds.

- GenApi::IInteger & LineDebouncerTimeRaw

  Sets the raw value of the selected line debouncer time.

- GenApi::IBoolean & LineStatus

  Indicates the current logical state for the selected line.

- GenApi::IInteger & LineStatusAll

  A single bitfield indicating the current logical state of all available line signals at time of polling.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::UserOutputSelectorEnums > & UserOutputSelector

  Selects the user settable output signal to configure. Once a user settable output signal has been selected, all changes to the user settable output signal settings will be applied to the selected user settable output signal.

- GenApi::IBoolean & UserOutputValue

  Sets the state of the selected user settable output signal.

- GenApi::IInteger & UserOutputValueAll

  A single bitfield that sets the state of all user settable output signals in one access.

- GenApi::IFloat & TimerDelayTimebaseAbs

  Sets the time base (in microseconds) that is used when a timer delay is set with the 'timer delay raw' setting.

- GenApi::IFloat & TimerDurationTimebaseAbs

  Sets the time base (in microseconds) that is used when a timer duration is set with the 'timer duration raw' setting.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TimerSelectorEnums > & TimerSelector

  Selects the timer to configure. Once a timer has been selected, all changes to the timer settings will be applied to the selected timer.

- GenApi::IFloat & TimerDelayAbs

  Directly sets the delay for the selected timer in microseconds.

- GenApi::IInteger & TimerDelayRaw

  Sets the 'raw' delay for the selected timer. Actual delay = raw timer delay setting x timer delay time base abs setting.

- GenApi::IFloat & TimerDurationAbs

  Directly sets the duration for the selected timer in microseconds.

- GenApi::IInteger & TimerDurationRaw

  Sets the 'raw' duration for the selected timer. Actual duration = raw timer duration setting x timer duration time base abs setting.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TimerTriggerSourceEnums > & TimerTriggerSource

  Sets the internal camera signal used to trigger the selected timer.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TimerTriggerActivationEnums > & TimerTriggerActivation

  Sets the type of signal transistion that will start the timer.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::LUTSelectorEnums > & LUTSelector

Selects the lookup table (LUT) to configure. Once a LUT has been selected, all changes to the LUT settings will be applied to the selected LUT.

- GenApi::IBoolean & LUTEnable

  Enables the selected LUT.

- GenApi::IInteger & LUTIndex

  Sets the LUT element to access.

- GenApi::IInteger & LUTValue

  Sets the value of the LUT element at the LUT index.

- GenApi::IRegister & LUTValueAll

  Accesses the entire content of the selected LUT in one chunk access.

- GenApi::IInteger & PayloadSize

  Size of the payload in bytes.

- GenApi::IInteger & PacketSize

  Sets the packet size in bytes.

- GenApi::IInteger & RecommendedPacketSize

  Indicates the recommended packet size in bytes. If a smaller packet size is used, the camera's maximum allowed frame rate will be reduced.

- GenApi::IInteger & TLParamsLocked

  Indicates whether a live grab is under way.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::UserSetSelectorEnums > & UserSet-Selector

  Selects the configuration set to load, save, or configure. Once a configuration set has been selected, all changes to the configuration set settings will be applied to the selected configuration set.

- GenApi::ICommand & UserSetLoad

  Loads the selected configuration into the camera's volatile memory and makes it the active configuration set. Once the selected set is loaded, the parameters in the selected set will control the camera.

- GenApi::ICommand & UserSetSave

  Saves the current active configuration set into the selected user set.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::UserSetDefaultSelectorEnums > & UserSetDefaultSelector

  Sets the configuration set to be used as the default startup set. The configuration set that has been selected as the default startup set will be loaded as the active set whenever the camera is powered on or reset.

- GenApi::IInteger & DeviceSerialNumber

  Indicates the serial number of the device.

- GenApi::IString & DeviceFirmwareVersion

  Indicates the version of the device's firmware/software.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::DeviceScanTypeEnums > & DeviceScanType

  Indicates the scan type of the device's sensor.

- GenApi::ICommand & DeviceReset

  Immediately resets and reboots the device.

- GenApi::IInteger & SensorWidth

  Indicates the width of the camera's sensor in pixels.

- GenApi::IInteger & SensorHeight

  Indicates the height of the camera's sensor in pixels.

- GenApi::IInteger & WidthMax

  Indicates the maximum allowed width of the image in pixels.

- GenApi::IInteger & HeightMax

  Indicates the maximum allowed height of the image in pixels.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::TemperatureSelectorEnums > & TemperatureSelector

  Lists the temperature sources available for readout.

- GenApi::IFloat & TemperatureAbs

  Shows the current temperature of the selected target in degrees celcius.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::ParameterSelectorEnums > & ParameterSelector

  Selects the parameter to configure. Once a parameter has been selected, all changes made using the Remove Limits feature will be applied to the selected parameter.

- GenApi::IBoolean & RemoveLimits

  Removes the factory-set limits of the selected parameter.

- GenApi::IBoolean & ChunkModeActive

  Enables the chunk mode.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::ChunkSelectorEnums > & ChunkSelector

  Selects the chunk to configure. Once a chunk has been selected, all changes to the chunk settings will be applied to the selected chunk.

- GenApi::IBoolean & ChunkEnable

  Enables the inclusion of the selected chunk in the payload data.

- GenApi::IInteger & ChunkTimestamp

  Indicates the value of the timestamp when the image was acquired.

- GenApi::IInteger & ChunkFramecounter

  Indicates the value of the frame counter when the image was acquired.

- GenApi::IInteger & ChunkLineStatusAll

  A bit field that indicates the status of all of the camera's input and output lines when the image was acquired.

- GenApi::IInteger & ChunkTriggerinputcounter

  Indicates the value of the trigger input counter when the image was acquired.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::EventSelectorEnums > & EventSelector

  Selects the type of event to configure. Once an event has been selected, all changes to the event settings will be applied to the selected event.

- GenApi::IEnumerationT< Basler_IIDC1394CameraParams::EventNotificationEnums > & EventNotification

  Sets the notification type that will be sent to the host application for the selected event.

- GenApi::IInteger & ExposureEndEventStreamChannelIndex

  Indicates the stream channel index for an exposure end event.

- GenApi::IInteger & ExposureEndEventFrameID

  Indicates the frame ID for an exposure end event.

- GenApi::IInteger & ExposureEndEventTimestamp

  Indicates the time stamp for an exposure end event.

- GenApi::IInteger & EventOverrunEventStreamChannelIndex

  Indicates the stream channel index for an event overrun event.

- GenApi::IInteger & EventOverrunEventFrameID

  Indicates the frame ID for an event overrun event.

- GenApi::IInteger & EventOverrunEventTimestamp

  Indicates the time stamp for an event overrun event.

4.44.1 Detailed Description

DRAFT: Basler generic GigEVision and IIDC camera interface.

4.44.2 Constructor & Destructor Documentation

4.44.2.1 Pylon::CBasler1394Camera::CBasler1394Camera (IPylonDevice *) [inline]

Creates a camera object that takes the ownership over an pylon device.

When having the ownership, the destructor of this camera object destroys the pylon device the camera object is attached to.

4.44.3 Member Function Documentation

4.44.3.1 void Pylon::CBasler1394Camera::Attach (IPylonDevice * *pDevice*, bool *takeOwnership* = `true`) `[inline]`

Attach a the camera object to a pylon device.

It is not allowed to call Attach when the camera object is already attache!

When having the ownership, the destructor of this camera object destroys the pylon device the camera object is attached to. Otherwise, the pylon device object remains valid when the camera object has been destroyed.

4.44.4 Member Data Documentation

4.44.4.1 GenApi::IFloat& Pylon::CBasler1394Camera::AcquisitionFrameRateAbs

If the acquisition frame rate feature is enabled, this value sets the camera's acquisition frame rate in frames per second.

Sets the 'absolute' value of the acquisition frame rate. The 'absolute' value is a float value that sets the acquisition frame rate in frames per second.

4.44.4.2 GenApi::IBoolean& Pylon::CBasler1394Camera::AcquisitionFrameRateEnable

Enables setting the camera's acquisition frame rate to a specified value.

This boolean value enables setting the camera's acquisition frame rate to a specified value.

4.44.4.3 GenApi::IFloat& Pylon::CBasler1394Camera::AcquisitionLineRateAbs

Sets the camera's acquisition line rate in lines per second.

Sets the 'absolute' value of the acquisition line rate. The 'absolute' value is a float value that sets the acquisition line rate in lines per second.

4.44.4.4 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::AcquisitionModeEnums >& Pylon::CBasler1394Camera::AcquisitionMode

Sets the image acquisition mode.

This enumeration sets the image acquisition mode.

4.44.4.5 GenApi::ICommand& Pylon::CBasler1394Camera::AcquisitionStart

Starts the acquisition of images.

This command starts the acquisition of images. If the camera is set for single frame acquisition, it will start acquisition of one frame. If the camera is set for continuous frame acquisition, it will start continuous acquisition of frames.

4.44.4.6 GenApi::ICommand& Pylon::CBasler1394Camera::AcquisitionStop

Stops the acquisition of images.

If the camera is set for continuous image acquisition and acquisition has been started, this command stops the acquisition of images.

4.44.4.7 GenApi::IFloat& Pylon::CBasler1394Camera::BalanceRatioAbs

Sets the value of the selected balance ratio control as a float.

This value sets the selected balance ratio control as a float value.

4.44.4.8 GenApi::IInteger& Pylon::CBasler1394Camera::BalanceRatioRaw

Sets the value of the selected balance ratio control as an integer.

This value sets the selected balance ratio control as an integer.

4.44.4.9 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::BalanceRatioSelectorEnums>& Pylon::CBasler1394Camera::BalanceRatioSelector

Selects a balance ratio to configure. Once a balance ratio control has been selected, all changes to the balance ratio settings will be applied to the selected control.

This enumeration selects a balance ratio control to configuration. Once a balance ratio control has been selected, all changes to the balance ratio settings will be applied to the selected control.

4.44.4.10 GenApi::IInteger& Pylon::CBasler1394Camera::BinningHorizontal

Sets the number of adjacent horizontal pixes to be summed.

Sets the number of binned adjacent horizontal pixels. Their charges will be summed and reported out of the camera as a single pixel.

4.44.4.11 GenApi::IInteger& Pylon::CBasler1394Camera::BinningVertical

Sets the number of adjacent vertical pixes to be summed.

Sets the number of binned adjacent vertical pixels. Their charges will be summed and reported out of the camera as a single pixel.

4.44.4.12 GenApi::IInteger& Pylon::CBasler1394Camera::BlackLevelRaw

Sets the value of the selected black level control as an integer.

This value sets the selected black level control as an integer.

4.44.4.13 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::BlackLevelSelectorEnums >& Pylon::CBasler1394Camera::BlackLevelSelector

Selcts a black level control to configure. Once a black level control has been selected, all changes to the black level settings will be applied to the selected control.

This enumeration selects the black level control to configure. Once a black level control has been selected, all changes to the black level settings will be applied to the selected control.

4.44.4.14 GenApi::IBoolean& Pylon::CBasler1394Camera::ChunkEnable

Enables the inclusion of the selected chunk in the payload data.

This boolean value enables the inclusion of the selected chunk in the payload data.

4.44.4.15 GenApi::IInteger& Pylon::CBasler1394Camera::ChunkFramecounter

Indicates the value of the frame counter when the image was acquired.

This integer indicates the value of the frame counter when the image was acquired.

4.44.4.16 GenApi::IInteger& Pylon::CBasler1394Camera::ChunkLineStatusAll

A bit field that indicates the status of all of the camera's input and output lines when the image was acquired.

This value is a bit field that indicates the status of all of the camera's input and output lines when the image was acquired.

4.44.4.17 GenApi::IBoolean& Pylon::CBasler1394Camera::ChunkModeActive

Enables the chunk mode.

This boolean value enables the camera's chunk mode.

4.44.4.18 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::ChunkSelectorEnums >& Pylon::CBasler1394Camera::ChunkSelector

Selects the chunk to configure. Once a chunk has been selected, all changes to the chunk settings will be applied to the selected chunk.

This enumeration selects the chunk to configure. Once a chunk has been selected, all changes to the chunk settings will be applied to the selected chunk.

4.44.4.19 GenApi::IInteger& Pylon::CBasler1394Camera::ChunkTimestamp

Indicates the value of the timestamp when the image was acquired.

This integer indicates the value of the timestamp when the image was acquired.

4.44.4.20 GenApi::IInteger& Pylon::CBasler1394Camera::ChunkTriggerinputcounter

Indicates the value of the trigger input counter when the image was acquired.

This integer indicates the value of the trigger input counter when the image was acquired.

4.44.4.21 GenApi::IString& Pylon::CBasler1394Camera::DeviceFirmwareVersion

Indicates the version of the device's firmware/software.

This is a read only element. It is a string that indicates the version of the device's firmware and software.

4.44.4.22 GenApi::ICommand& Pylon::CBasler1394Camera::DeviceReset

Immediately resets and reboots the device.

This is a command that immediately resets and reboots the device.

4.44.4.23 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::DeviceScanTypeEnums >& Pylon::CBasler1394Camera::DeviceScanType

Indicates the scan type of the device's sensor.

This enumeration lists the possible scan types for the sensor in the device.

4.44.4.24 GenApi::IInteger& Pylon::CBasler1394Camera::DeviceSerialNumber

Indicates the serial number of the device.

This is a read only element. It is an integer that indicates the serial number of the device.

4.44.4.25 GenApi::IInteger& Pylon::CBasler1394Camera::DigitalShift

Sets the value of the selected digital shift control.

This value sets the selected digital shift control

4.44.4.26 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::EventNotificationEnums >& Pylon::CBasler1394Camera::EventNotification

Sets the notification type that will be sent to the host application for the selected event.

This enumeration sets the notification type that will be sent to the host application for the selected event.

4.44.4.27 GenApi::IInteger& Pylon::CBasler1394Camera::EventOverrunEventFrameID

Indicates the frame ID for an event overrun event.

This enumeration value indicates the frame ID for an event overrun event

4.44.4.28 GenApi::IInteger& Pylon::CBasler1394Camera::EventOverrunEventStreamChannelIndex

Indicates the stream channel index for an event overrun event.

This enumeration value indicates the stream channel index for an event overrun event

4.44.4.29 GenApi::IInteger& Pylon::CBasler1394Camera::EventOverrunEventTimestamp

Indicates the time stamp for an event overrun event.

This enumeration value indicates the time stamp for an event overrun event

4.44.4.30 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::EventSelectorEnums >& Pylon::CBasler1394Camera::EventSelector

Selects the type of event to configure. Once an event has been selected, all changes to the event settings will be applied to the selected event.

This enumeration selects the type of event to configure. Once an event has been selected, all changes to the event settings will be applied to the selected event.

4.44.4.31 GenApi::IInteger& Pylon::CBasler1394Camera::ExposureEndEventFrameID

Indicates the frame ID for an exposure end event.

This enumeration value indicates the frame ID for an exposure end event.

4.44.4.32 GenApi::IInteger& Pylon::CBasler1394Camera::ExposureEndEventStreamChannelIndex

Indicates the stream channel index for an exposure end event.

This enumeration value indicates the stream channel index for an exposure end event.

4.44.4.33 GenApi::IInteger& Pylon::CBasler1394Camera::ExposureEndEventTimestamp

Indicates the time stamp for an exposure end event.

This enumeration value indicates the time stamp for an exposure end event.

4.44.4.34 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::ExposureModeEnums >& Pylon::CBasler1394Camera::ExposureMode

Sets the exposure mode.

This enumeration sets the exposure mode.

4.44.4.35 GenApi::IFloat& Pylon::CBasler1394Camera::ExposureTimeAbs

Directly sets the camera's exposure time in microseconds.

This float value sets the camera's exposure time in microseconds.

4.44.4.36 GenApi::IFloat& Pylon::CBasler1394Camera::ExposureTimeBaseAbs

Sets the time base (in microseconds) that is used when the exposure time is set with the 'exposure time raw' setting.

This float value sets the time base (in microseconds) that is used when the exposure time is set with the 'raw' setting.

4.44.4.37 GenApi::IBoolean& Pylon::CBasler1394Camera::ExposureTimeBaseAbsEnable

Enables the use of the exposure time base.

This value enables the use of the exposure time base.

4.44.4.38 GenApi::IInteger& Pylon::CBasler1394Camera::ExposureTimeRaw

Sets the 'raw' exposure time. Actual exposure time = raw exposure setting x exposure time base abs setting.

This value sets an integer that will be used as a multiplier for the exposure timebase. The actual exposure time equals the current exposure time raw setting times the current exposure time base abs setting.

4.44.4.39 GenApi::IInteger& Pylon::CBasler1394Camera::GainRaw

This is an integer value that sets the selected gain control in device specific units.

Sets the 'raw' value of the selected gain control. The 'raw' value is an integer value that sets the selected gain control in units specific to the camera.

4.44.4.40 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::GainSelectorEnums >& Pylon::CBasler1394Camera::GainSelector

Selects the gain control to configure. Once a gain control has been selected, all changes to the gain settings will be applied to the selected control.

This enumeration selects the gain control to configure. Once a gain control has been selected, all changes to the gain settings will be applied to the selected control.

4.44.4.41 GenApi::IFloat& Pylon::CBasler1394Camera::Gamma

This feature is used to perform gamma correction of pixel intensity.

This feature is used to perform gamma correction of pixel intensity. This is typically used to compensate for non-linearity of the display system (such as CRT).

4.44.4.42 GenApi::IBoolean& Pylon::CBasler1394Camera::GammaEnable

Enables the gamma correction.

This boolean value enables the gamma correction.

4.44.4.43 GenApi::IInteger& Pylon::CBasler1394Camera::Height

Sets the height of the area of interest in pixels.

This value sets the height of the area of interest in pixels.

4.44.4.44 GenApi::IInteger& Pylon::CBasler1394Camera::HeightMax

Indicates the maximum allowed height of the image in pixels.

This is a read only element. It is an integer that indicates maximum allowed height of the image in pixels taking into account any function that may limit the allowed height.

4.44.4.45 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::LegacyBinningVerticalEnums >& Pylon::CBasler1394Camera::LegacyBinningVertical

Sets the vertical binning feature.

This enumeration sets the vertical binning feature.

4.44.4.46 GenApi::IFloat& Pylon::CBasler1394Camera::LineDebouncerTimeAbs

Sets the absolute value of the selected line debouncer time in microseconds.

Sets the absolute value of the selected line debouncer time in microseconds

4.44.4.47 GenApi::IInteger& Pylon::CBasler1394Camera::LineDebouncerTimeRaw

Sets the raw value of the selected line debouncer time.

Sets the raw value of the selected line debouncer time

4.44.4.48 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::LineFormatEnums >& Pylon::CBasler1394Camera::LineFormat

Sets the electrical configuration of the selected line.

This feature controls the current electrical format of the selected physical input or output Line. Line Format can take any of the following values: No Connect: The Line is not connected. Tri-state: The Line is currently in Tri-state mode (Not driven). TTL: The Line is currently accepting or sending TTL level signals. LVDS: The Line is currently accepting or sending LVDS level signals. RS-422: The Line is currently accepting or sending RS-422 level signals. Opto-coupled: The Line is Opto-coupled.

4.44.4.49 GenApi::IBoolean& Pylon::CBasler1394Camera::LineInverter

Enables the inverter function for the selected line.

This boolean value enables the inverter function for the selected line.

4.44.4.50 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::LineModeEnums >& Pylon::CBasler1394Camera::LineMode

Sets the mode for the selected line.

This feature controls whether the physical Line is used to Input or Output a signal. When a Line supports input and output mode, the default state is Input to avoid possible electrical contention. Line Mode can take any of the following values: Input: The selected physical line is used to input an electrical signal. Output: The selected physical line is used to output an electrical signal.

4.44.4.51 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::LineSelectorEnums >& Pylon::CBasler1394Camera::LineSelector

Selects the I/O line to configure. Once a line has been selected, all changes to the line settings will be applied to the selected line.

This enumeration selects the I/O line to configure. Once a line has been selected, all changes to the line settings will be applied to the selected line.

4.44.4.52 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::LineSourceEnums >& Pylon::CBasler1394Camera::LineSource

Sets the source signal for the selected line (if the selected line is an output).

This enumeration selects the internally generated camera signal (source signal) for the selected line when the selected line is an output.

4.44.4.53 GenApi::IBoolean& Pylon::CBasler1394Camera::LineStatus

Indicates the current logical state for the selected line.

This boolean value indicates the current logical state for the selected line at the time of polling.

4.44.4.54 GenApi::IInteger& Pylon::CBasler1394Camera::LineStatusAll

A single bitfield indicating the current logical state of all available line signals at time of polling.

This integer value is a single bitfield that indicates the current logical state of all available lines at time of polling.

4.44.4.55 GenApi::IBoolean& Pylon::CBasler1394Camera::LUTEnable

Enables the selected LUT.

This boolean value enables the selected LUT.

4.44.4.56 GenApi::IInteger& Pylon::CBasler1394Camera::LUTIndex

Sets the LUT element to access.

This value sets the LUT element to access. This value is used to index into a LUT array.

4.44.4.57 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::LUTSelectorEnums >& Pylon::CBasler1394Camera::LUTSelector

Selects the lookup table (LUT) to configure. Once a LUT has been selected, all changes to the LUT settings will be applied to the selected LUT.

This enumeration the lookup table (LUT) to configure. Once a LUT has been selected, all changes to the LUT settings will be applied to the selected LUT.

4.44.4.58 GenApi::IInteger& Pylon::CBasler1394Camera::LUTValue

Sets the value of the LUT element at the LUT index.

This value sets the value of the LUT element at the LUT index.

4.44.4.59 GenApi::IInteger& Pylon::CBasler1394Camera::OffsetX

Sets the X offset (left offset) of the area of interest in pixels.

This value sets the X offset (left offset) for the area of interest in pixels, i.e., the distance in pixels between the left side of the sensor and the left side of the image area.

4.44.4.60 GenApi::IInteger& Pylon::CBasler1394Camera::OffsetY

Sets the Y offset (top offset) for the area of interest in pixels.

This value sets the Y offset (top offset) for the area of interest, i.e., the distance in pixels between the top of the sensor and the top of the image area.

4.44.4.61 GenApi::IInteger& Pylon::CBasler1394Camera::PacketSize

Sets the packet size in bytes.

This value sets the packet size in bytes.

4.44.4.62 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::ParameterSelectorEnums >& Pylon::CBasler1394Camera::ParameterSelector

Selects the parameter to configure. Once a parameter has been selected, all changes made using the Remove Limits feature will be applied to the selected parameter.

This enumeration selects the parameter to configure. Selects the parameter to configure. Once a parameter has been selected, all changes made using the Remove Limits feature will be applied to the selected parameter

4.44.4.63 GenApi::IInteger& Pylon::CBasler1394Camera::PayloadSize

Size of the payload in bytes.

Size of the payload in bytes. This is the total number of bytes sent in the payload. Image data + chunk data if present. No packet headers.

4.44.4.64 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::PixelColorFilterEnums >& Pylon::CBasler1394Camera::PixelColorFilter

Indicates the alignment of the camera's Bayer filter to the pixels in the acquired images.

This is a read only feature. This enumeration provides a list of values that indicate the alignment of the camera's Bayer filter to the pixels in the acquired images.

4.44.4.65 GenApi::IInteger& Pylon::CBasler1394Camera::PixelDynamicRangeMax

Indicates the maximum possible pixel value that could be transferred from the camera.

This a read only feature. It indicates the maximum possible pixel value that could be transferred from the camera.

4.44.4.66 GenApi::IInteger& Pylon::CBasler1394Camera::PixelDynamicRangeMin

Indicates the minimum possible pixel value that could be transferred from the camera.

This a read only feature. It indicates the minimum possible pixel value that could be transferred from the camera.

4.44.4.67 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::PixelFormatEnums >& Pylon::CBasler1394Camera::PixelFormat

Sets the format of the pixel data transmitted for acquired images.

This enumeration sets the format of the pixel data transmitted for acquired images.

4.44.4.68 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::PixelSizeEnums >& Pylon::CBasler1394Camera::PixelSize

Indicates the depth of the pixel values in the image in bits per pixel.

This is a read only feature. This enumeration provides a list of values that indicate the depth of the pixel values in the acquired images in bits per pixel. This value will always be coherent with the pixel format setting.

4.44.4.69 GenApi::IInteger& Pylon::CBasler1394Camera::RecommendedPacketSize

Indicates the recommended packet size in bytes. If a smaller packet size is used, the camera's maximum allowed frame rate will be reduced.

This value indicates the recommended packet size in bytes. If a smaller packet size is used, the camera's maximum allowed frame rate will be reduced.

4.44.4.70 GenApi::IBoolean& Pylon::CBasler1394Camera::RemoveLimits

Removes the factory-set limits of the selected parameter.

Removes the factory-set limits of the selected parameter. Having removed the factory-set limits you may set the parameter within extended limits. These are only defined by technical restrictions. Note: Inferior image quality may result.

4.44.4.71 GenApi::IFloat& Pylon::CBasler1394Camera::ResultingFrameRateAbs

Indicates the maximum allowed frame acquisition rate (in frames per second) given the current settings for the area of interest, exposure time, and bandwidth.

Indicates the 'absolute' value of the maximum allowed acquisition frame rate. The 'absolute' value is a float value that indicates the maximum allowed acquisition frame rate in frames per second given the current settings for the area of interest, exposure time, and bandwidth.

4.44.4.72 GenApi::IFloat& Pylon::CBasler1394Camera::ResultingLineRateAbs

Indicates the maximum allowed line acquisition rate (in lines per second) given the current settings for the area of interest, exposure time, and bandwidth.

Indicates the 'absolute' value of the maximum allowed acquisition line rate. The 'absolute' value is a float value that indicates the maximum allowed acquisition line rate in lines per second given the current settings for the area of interest, exposure time, and bandwidth.

4.44.4.73 GenApi::IInteger& Pylon::CBasler1394Camera::SensorHeight

Indicates the height of the camera's sensor in pixels.

This is a read only element. It is an integer that indicates the actual height of the camera's sensor in pixels.

4.44.4.74 GenApi::IInteger& Pylon::CBasler1394Camera::SensorWidth

Indicates the width of the camera's sensor in pixels.

This is a read only element. It is an integer that indicates the actual width of the camera's sensor in pixels.

4.44.4.75 GenApi::IFloat& Pylon::CBasler1394Camera::TemperatureAbs

Shows the current temperature of the selected target in degrees celcius.

Shows the current temperature of the selected target in degrees celcius

4.44.4.76 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TemperatureSelectorEnums >& Pylon::CBasler1394Camera::TemperatureSelector

Lists the temperature sources available for readout.

Lists the temperature sources available for readout

4.44.4.77 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TestImageSelectorEnums >& Pylon::CBasler1394Camera::TestImageSelector

Selecting a test image from the list will enable the test image.

This enumeration provides a list of the available test images. Selecting a test image from the list will enable the test image.

4.44.4.78 GenApi::IFloat& Pylon::CBasler1394Camera::TimerDelayAbs

Directly sets the delay for the selected timer in microseconds.

This float value sets the delay for the selected timer in microseconds.

4.44.4.79 GenApi::IInteger& Pylon::CBasler1394Camera::TimerDelayRaw

Sets the 'raw' delay for the selected timer. Actual delay = raw timer delay setting x timer delay time base abs setting.

This value sets an integer that will be used as a multiplier for the timer delay timebase. The actual delay time equals the current timer delay raw setting times the current timer delay time base abs setting.

4.44.4.80 GenApi::IFloat& Pylon::CBasler1394Camera::TimerDelayTimebaseAbs

Sets the time base (in microseconds) that is used when a timer delay is set with the 'timer delay raw' setting.

This float value sets the time base (in microseconds) that is used when a timer delay is set with the 'raw' setting.

4.44.4.81 GenApi::IFloat& Pylon::CBasler1394Camera::TimerDurationAbs

Directly sets the duration for the selected timer in microseconds.

This float value sets the duration for the selected timer in microseconds.

4.44.4.82 GenApi::IInteger& Pylon::CBasler1394Camera::TimerDurationRaw

Sets the 'raw' duration for the selected timer. Actual duration = raw timer duration setting x timer duration time base abs setting.

This value sets an integer that will be used as a multiplier for the timer duration timebase. The actual duration time equals the current timer duration raw setting times the current timer duration time base abs setting.

4.44.4.83 GenApi::IFloat& Pylon::CBasler1394Camera::TimerDurationTimebaseAbs

Sets the time base (in microseconds) that is used when a timer duration is set with the 'timer duration raw' setting.

This float value sets the time base (in microseconds) that is used when a timer duration is set with the 'raw' setting.

4.44.4.84 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TimerSelectorEnums >& Pylon::CBasler1394Camera::TimerSelector

Selects the timer to configure. Once a timer has been selected, all changes to the timer settings will be applied to the selected timer.

This enumeration selects the timer to configure. . Once a timer has been selected, all changes to the timer settings will be applied to the selected timer.

4.44.4.85 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TimerTriggerActivationEnums >& Pylon::CBasler1394Camera::TimerTriggerActivation

Sets the type of signal transistion that will start the timer.

This enumeration sets the type of signal transistion that will start the timer.

4.44.4.86 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TimerTriggerSourceEnums >& Pylon::CBasler1394Camera::TimerTriggerSource

Sets the internal camera signal used to trigger the selected timer.

This enumeration sets the internal camera signal used to trigger the selected timer.

4.44.4.87 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TriggerActivationEnums >& Pylon::CBasler1394Camera::TriggerActivation

Sets the signal transition needed to activate the selected trigger.

This enumeration sets the signal transition needed to activate the selected trigger.

4.44.4.88 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TriggerModeEnums >& Pylon::CBasler1394Camera::TriggerMode

Sets the mode for the selected trigger.

This enumeration sets the trigger mode for the selected trigger.

4.44.4.89 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TriggerSelectorEnums >& Pylon::CBasler1394Camera::TriggerSelector

Selects the trigger type to configure. Once a trigger type has been selected, all changes to the trigger settings will be applied to the selected trigger.

This enumeration selects the trigger type to configure. Once a trigger type has been selected, all changes to the trigger settings will be applied to the selected trigger.

4.44.4.90 GenApi::ICommand& Pylon::CBasler1394Camera::TriggerSoftware

Generates a software trigger signal that is used when the trigger source is set to 'software'.

This command generates a software trigger signal. The software trigger signal will be used if the trigger source is set to 'software'.

4.44.4.91 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::TriggerSourceEnums >& Pylon::CBasler1394Camera::TriggerSource

Sets the signal source for the selected trigger.

This enumeration sets the signal source for the selected trigger.

4.44.4.92 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::UserOutputSelectorEnums >& Pylon::CBasler1394Camera::UserOutputSelector

Selects the user settable output signal to configure. Once a user settable output signal has been selected, all changes to the user settable output signal settings will be applied to the selected user settable output signal.

This enumeration selects the user settable output signal to configure. Once a user settable output signal has been selected, all changes to the user settable output signal settings will be applied to the selected user settable output signal.

4.44.4.93 GenApi::IBoolean& Pylon::CBasler1394Camera::UserOutputValue

Sets the state of the selected user settable output signal.

This boolean value sets the state of the selected user settable output signal.

4.44.4.94 GenApi::IInteger& Pylon::CBasler1394Camera::UserOutputValueAll

A single bitfield that sets the state of all user settable output signals in one access.

This integer value is a single bitfield that sets the state of all user settable output signals in one access.

4.44.4.95 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::UserSetDefaultSelectorEnums >& Pylon::CBasler1394Camera::UserSetDefaultSelector

Sets the configuration set to be used as the default startup set. The configuration set that has been selected as the default startup set will be loaded as the active set whenever the camera is powered on or reset.

This enumeration sets the configuration set to be used as the default startup set. The configuration set that has been selected as the default startup set will be loaded as the active set whenever the camera is powered on or reset.

4.44.4.96 GenApi::ICommand& Pylon::CBasler1394Camera::UserSetLoad

Loads the selected configuration into the camera's volatile memory and makes it the active configuration set. Once the selected set is loaded, the parameters in the selected set will control the camera.

This command loads the selected configuration set from the non-volatile memory in the camera to the volatile memory and makes the selected set the active configuration set. Once the selected set is loaded, the parameters in the selected set will control the camera.

4.44.4.97 GenApi::ICommand& Pylon::CBasler1394Camera::UserSetSave

Saves the current active configuration set into the selected user set.

This command copies the parameters in the current active configuration set into the selected user set in the camera's non-volatile memory.

4.44.4.98 GenApi::IEnumerationT<Basler_IIDC1394CameraParams::UserSetSelectorEnums >& Pylon::CBasler1394Camera::UserSetSelector

Selects the configuration set to load, save, or configure. Once a configuration set has been selected, all changes to the configuration set settings will be applied to the selected configuration set.

This enumeration selects the configuration set to load, save or configure. Possible values for the User Set Selector are: Default: Selects a configuration set that contains factory settings. User Set 1: Selects the first user set. When the Default configuration set is selected and loaded using User Set Load, the device

must be in default factory settings state and must make sure the mandatory continuous acquisition use case works directly. Default User Set is read-only and cannot be modified.

4.44.4.99 GenApi::IInteger& Pylon::CBasler1394Camera::Width

Sets the width of the area of interest in pixels.

This value sets the width of the area of interest in pixels.

4.44.4.100 GenApi::IInteger& Pylon::CBasler1394Camera::WidthMax

Indicates the maximum allowed width of the image in pixels.

This is a read only element. It is an integer that indicates maximum allowed width of the image in pixels taking into account any function that may limit the allowed width.

4.45 Pylon::CBasler1394DeviceInfo Class Reference

FireWire specific Device Info object. Does not support SerialNumber property.

Inherits Pylon::CDeviceInfo.

Public Member Functions

- String_t GetInternalName () const

  Retrieves the device name as used by the Basler pylon 1394 Driver.

- String_t GetSerialNumber () const

  Retrieves the serial number if it supported by the underlying implementation, or "N/A".

- String_t GetUserDefinedName () const

  Retrieves the user-defined name if present.

- String_t GetModelName () const

  Retrieves the model name of the device.

- String_t GetDeviceVersion () const

  Retrieves the version string of the device.

- String_t GetDeviceFactory () const

  Retrieves the device factory able to create this device.

- bool operator== (const CInfoBase &rhs)

  compare two info objects

- String_t GetFriendlyName () const

  Retrieves a human readable name.

- String_t GetFullName () const

  Retrieves the unique device name.

- String_t GetVendorName () const

  Retrieves the device vendor name.

- String_t GetDeviceClass () const

  Retrieves the device class.

- virtual int GetPropertyNames (StringList_t &) const

  Retrieves a list of property names.

- virtual bool GetPropertyValue (const String_t &Name, String_t &Value) const

  Retrieves a property value.

- virtual void SetPropertyValue (const String_t &Name, const String_t &Value)

  Modifies a property value.

4.45.1 Detailed Description

FireWire specific Device Info object. Does not support SerialNumber property.

4.46 Pylon::CBasler1394EventGrabber Class Reference

Interface to the FireWire event parameters.

Inherits Pylon::CEventGrabberProxy.

Public Member Functions

- virtual ~CBasler1394EventGrabber (void)
  Destructor.

Construction

Some smart pointer functionality

- void Attach (Pylon::IEventGrabber *pInterface)
  Attach an interface pointer.

Public Attributes

- GenApi::IInteger & NumBuffer
  Number of buffers to be used.

4.46.1 Detailed Description

Interface to the FireWire event parameters.

4.46.2 Member Function Documentation

4.46.2.1 void Pylon::CBasler1394EventGrabber::Attach (Pylon::IEventGrabber * *pInterface*) [inline]

Attach an interface pointer.

It is not allowed to call Attach() when there is already an interface pointer attached!

Reimplemented from Pylon::CEventGrabberProxy.

4.46.3 Member Data Documentation

4.46.3.1 GenApi::IInteger& Pylon::CBasler1394EventGrabber::NumBuffer

Number of buffers to be used.

Number of Buffers that are going to be used receiving events. .

4.47 Pylon::CBasler1394StreamGrabber Class Reference

Interface to the FireWire stream parameters.

Inherits Pylon::CStreamGrabberProxy.

Public Member Functions

- virtual ~CBasler1394StreamGrabber (void)
  Destructor.

Construction

Some smart pointer functionality

- void Attach (Pylon::IStreamGrabber *pInterface)
  Attach an interface pointer.

Public Attributes

- GenApi::IInteger & MaxNumBuffer
  Number of buffers to be used.
- GenApi::IInteger & MaxBufferSize
  Buffer size in bytes.
- GenApi::IInteger & ThreadTimeout
  Timeout value in milliseconds for the output thread.
- GenApi::IInteger & ThreadPriority
  Priority value of the output thread (default=0).
- GenApi::IEnumerationT< Basler_IIDC1394StreamParams::StatusEnums > & Status
  Status of the stream grabber.

4.47.1 Detailed Description

Interface to the FireWire stream parameters.

4.47.2 Member Function Documentation

4.47.2.1 void Pylon::CBasler1394StreamGrabber::Attach (Pylon::IStreamGrabber * *pInterface*) [inline]

Attach an interface pointer.

It is not allowed to call Attach() when there is already an interface pointer attached!

Reimplemented from Pylon::CStreamGrabberProxy.

4.47.3 Member Data Documentation

4.47.3.1 GenApi::IInteger& Pylon::CBasler1394StreamGrabber::MaxBufferSize

Buffer size in bytes.

Size of the greatest buffer to be used. Unit is bytes. Raising this value may cause higher resource demands on a subsequent PrepareGrab call.

4.47.3.2 GenApi::IInteger& Pylon::CBasler1394StreamGrabber::MaxNumBuffer

Number of buffers to be used.

Maximum number of Buffers that are going to be used simultaneously. Raising this value may cause higher resource demands on a subsequent PrepareGrab call.

4.47.3.3 GenApi::IEnumerationT<Basler_IIDC1394StreamParams::StatusEnums >& Pylon::CBasler1394StreamGrabber::Status

Status of the stream grabber.

The stream grabber is initially in the status Closed. In this state it can only be opened, other operations are not allowed. It switches to Open when the stream grabber is opened successfully. A call to PrepareGrab locks the stream grabber and the camera. It is synchronized with the transport layer specific parameter of the camera. Both are released by calling FinishGrab and the status switches back to Open. On call of the Close method the status returns to Closed. In case of unrecoverable errors status turn invalid.

4.47.3.4 GenApi::IInteger& Pylon::CBasler1394StreamGrabber::ThreadPriority

Priority value of the output thread (default=0).

When the stream grabber is opened the output thread is started. Non-zero values are used to set the thread priority. It is not changed until the stream grabber is re-opened again.

4.48 Pylon::CBasler1394TI Class Reference

The FireWire specific implemenation of a TLFactory.

Inherits CTransportLayerBase.

4.48.1 Detailed Description

The FireWire specific implemenation of a TLFactory.

4.49 Pylon::CBaslerGigECamera Class Reference

Basler GigEVision camera interface.

Inherits Pylon::CPylonGigEDeviceProxy.

Public Member Functions

- virtual ~CBaslerGigECamera (void)

  Destructor.

Implementation of the IDevice

Implementaion of IPylonDevice

Construction

- CBaslerGigECamera ()

  Creates a camera object that is not attached to an pylon device. Use the Attach() method to attach the device.

- CBaslerGigECamera (IPylonDevice *)

  Creates a camera object that takes the ownership over an pylon device.

Some smart pointer functionality

- void Attach (IPylonDevice *pDevice, bool takeOwnership=true)

  Attach a the camera object to a pylon device.

Implementation of the IDevice

Implementaion of IPylonDevice

Implementation of the IDevice

Implementaion of IPylonDevice

Implementation of the IDevice

Implementaion of IPylonDevice

Static Public Member Functions

- static String_t DeviceClass ()

  The name of this device class, use this for enumeration.

Public Attributes

- GenApi::IEnumerationT< Basler_GigECameraParams::GainSelectorEnums > & GainSelector

  Selects the gain control to configure. Once a gain control has been selected, all changes to the gain settings will be applied to the selected control.

- GenApi::IInteger & GainRaw

  This is an integer value that sets the selected gain control in device specific units.

- GenApi::IFloat & GainAbs

  This is a float value that sets the selected gain control in dB.

- GenApi::IEnumerationT< Basler_GigECameraParams::BlackLevelSelectorEnums > & BlackLevel-Selector

  Selcts a black level control to configure. Once a black level control has been selected, all changes to the black level settings will be applied to the selected control.

- GenApi::IInteger & BlackLevelRaw

  Sets the value of the selected black level control as an integer.

- GenApi::IFloat & BlackLevelAbs

  Sets the value of the selected black level control as a float.

- GenApi::IEnumerationT< Basler_GigECameraParams::BalanceRatioSelectorEnums > & Balance-RatioSelector

  Selects a balance ratio to configure. Once a balance ratio control has been selected, all changes to the balance ratio settings will be applied to the selected control.

- GenApi::IFloat & BalanceRatioAbs

  Sets the value of the selected balance ratio control as a float.

- GenApi::IInteger & BalanceRatioRaw

  Sets the value of the selected balance ratio control as an integer.

- GenApi::IBoolean & GammaEnable

  Enables the gamma correction.

- GenApi::IFloat & Gamma

  This feature is used to perform gamma correction of pixel intensity.

- GenApi::IInteger & DigitalShift

  Sets the value of the selected digital shift control.

- GenApi::IEnumerationT< Basler_GigECameraParams::PixelFormatEnums > & PixelFormat

  Sets the format of the pixel data transmitted for acquired images.

- GenApi::IEnumerationT< Basler_GigECameraParams::PixelCodingEnums > & PixelCoding

  Sets the color coding of the pixels in the acquired images.

- GenApi::IEnumerationT< Basler_GigECameraParams::PixelSizeEnums > & PixelSize

  Indicates the depth of the pixel values in the image in bits per pixel.

- GenApi::IEnumerationT< Basler_GigECameraParams::PixelColorFilterEnums > & PixelColorFilter

  Indicates the alignment of the camera's Bayer filter to the pixels in the acquired images.

- GenApi::IInteger & PixelDynamicRangeMin

  Indicates the minimum possible pixel value that could be transferred from the camera.

- GenApi::IInteger & PixelDynamicRangeMax

  Indicates the maximum possible pixel value that could be transferred from the camera.

- GenApi::IInteger & SpatialCorrection

  Specifies the direction of imaging and the separation (consecutive numbers) of related line captures.

- GenApi::IInteger & SpatialCorrectionAmount

  TODO:.

- GenApi::IEnumerationT< Basler_GigECameraParams::SpatialCorrectionStartingLineEnums > & SpatialCorrectionStartingLine

  TODO:.

- GenApi::IEnumerationT< Basler_GigECameraParams::TestImageSelectorEnums > & TestImage-Selector

  Selecting a test image from the list will enable the test image.

- GenApi::IInteger & Width

  Sets the width of the area of interest in pixels.

- GenApi::IInteger & Height

  Sets the height of the area of interest in pixels.

- GenApi::IInteger & OffsetX

  Sets the X offset (left offset) of the area of interest in pixels.

- GenApi::IInteger & OffsetY

  Sets the Y offset (top offset) for the area of interest in pixels.

- GenApi::IEnumerationT< Basler_GigECameraParams::LegacyBinningVerticalEnums > & Legacy-BinningVertical

  Sets the vertical binning feature.

- GenApi::IInteger & BinningVertical

  Sets the number of adjacent vertical pixes to be summed.

- GenApi::IInteger & BinningHorizontal

  Sets the number of adjacent horizontal pixes to be summed.

- GenApi::IEnumerationT< Basler_GigECameraParams::AcquisitionModeEnums > & Acquisition-Mode

  Sets the image acquisition mode.

- GenApi::ICommand & AcquisitionStart

  Starts the acquisition of images.

- GenApi::ICommand & AcquisitionStop

  Stops the acquisition of images.

- GenApi::ICommand & AcquisitionAbort

  Immediately aborts the acquisition of images.

- GenApi::IEnumerationT< Basler_GigECameraParams::TriggerSelectorEnums > & TriggerSelector

  Selects the trigger type to configure. Once a trigger type has been selected, all changes to the trigger settings will be applied to the selected trigger.

- GenApi::IEnumerationT< Basler_GigECameraParams::TriggerModeEnums > & TriggerMode

  Sets the mode for the selected trigger.

- GenApi::ICommand & TriggerSoftware

  Generates a software trigger signal that is used when the trigger source is set to "software".

- GenApi::IEnumerationT< Basler_GigECameraParams::TriggerSourceEnums > & TriggerSource

  Sets the signal source for the selected trigger.

- GenApi::IEnumerationT< Basler_GigECameraParams::TriggerActivationEnums > & TriggerActivation

  Sets the signal transition needed to activate the selected trigger.

- GenApi::IEnumerationT< Basler_GigECameraParams::ExposureModeEnums > & ExposureMode

  Sets the exposure mode.

- GenApi::IFloat & ExposureTimeAbs

  Directly sets the camera's exposure time in microseconds.

- GenApi::IFloat & ExposureTimeBaseAbs

  Sets the time base (in microseconds) that is used when the exposure time is set with the "exposure time raw" setting.

- GenApi::IBoolean & ExposureTimeBaseAbsEnable

  Enables the use of the exposure time base.

- GenApi::IInteger & ExposureTimeRaw

  Sets the "raw" exposure time. Actual exposure time = raw exposure setting x exposure time base abs setting.

- GenApi::IFloat & AcquisitionLineRateAbs

  Sets the camera's acquisition line rate in lines per second.

- GenApi::IFloat & ResultingLineRateAbs

  Indicates the maximum allowed line acquisition rate (in lines per second) given the current settings for the area of interest, exposure time, and bandwidth.

- GenApi::IBoolean & AcquisitionFrameRateEnable

Enables setting the camera's acquisition frame rate to a specified value.

- GenApi::IFloat & AcquisitionFrameRateAbs

  If the acquisition frame rate feature is enabled, this value sets the camera's acquisition frame rate in frames per second.

- GenApi::IFloat & ResultingFrameRateAbs

  Indicates the maximum allowed frame acquisition rate (in frames per second) given the current settings for the area of interest, exposure time, and bandwidth.

- GenApi::IEnumerationT< Basler_GigECameraParams::LineSelectorEnums > & LineSelector

  Selects the I/O line to configure. Once a line has been selected, all changes to the line settings will be applied to the selected line.

- GenApi::IEnumerationT< Basler_GigECameraParams::LineModeEnums > & LineMode

  Sets the mode for the selected line.

- GenApi::IEnumerationT< Basler_GigECameraParams::LineFormatEnums > & LineFormat

  Sets the electrical configuration of the selected line.

- GenApi::IEnumerationT< Basler_GigECameraParams::LineSourceEnums > & LineSource

  Sets the source signal for the selected line (if the selected line is an output).

- GenApi::IBoolean & LineInverter

  Enables the inverter function for the selected line.

- GenApi::IBoolean & LineTermination

  TODO: Enables the termination function for the selected line.

- GenApi::IFloat & LineDebouncerTimeAbs

  Sets the absolute value of the selected line debouncer time in microseconds.

- GenApi::IInteger & LineDebouncerTimeRaw

  Sets the raw value of the selected line debouncer time.

- GenApi::IBoolean & LineStatus

  Indicates the current logical state for the selected line.

- GenApi::IInteger & LineStatusAll

  A single bitfield indicating the current logical state of all available line signals at time of polling.

- GenApi::IEnumerationT< Basler_GigECameraParams::UserOutputSelectorEnums > & UserOutput-Selector

  Selects the user settable output signal to configure. Once a user settable output signal has been selected, all changes to the user settable output signal settings will be applied to the selected user settable output signal.

- GenApi::IBoolean & UserOutputValue

  Sets the state of the selected user settable output signal.

- GenApi::IInteger & UserOutputValueAll

  A single bitfield that sets the state of all user settable output signals in one access.

- GenApi::IInteger & UserOutputValueAllMask

  Defines a mask that is used when the User Output Value All setting is used to set all of the user settable output signals in one access.

- GenApi::IFloat & TimerDelayTimebaseAbs

  Sets the time base (in microseconds) that is used when a timer delay is set with the "timer delay raw" setting.

- GenApi::IFloat & TimerDurationTimebaseAbs

  Sets the time base (in microseconds) that is used when a timer duration is set with the "timer duration raw" setting.

- GenApi::IEnumerationT< Basler_GigECameraParams::TimerSelectorEnums > & TimerSelector

  Selects the timer to configure. Once a timer has been selected, all changes to the timer settings will be applied to the selected timer.

- GenApi::IFloat & TimerDelayAbs

  Directly sets the delay for the selected timer in microseconds.

- GenApi::IInteger & TimerDelayRaw

  Sets the "raw" delay for the selected timer. Actual delay = raw timer delay setting x timer delay time base abs setting.

- GenApi::IFloat & TimerDurationAbs

  Directly sets the duration for the selected timer in microseconds.

- GenApi::IInteger & TimerDurationRaw

  Sets the "raw" duration for the selected timer. Actual duration = raw timer duration setting x timer duration time base abs setting.

- GenApi::IEnumerationT< Basler_GigECameraParams::TimerTriggerSourceEnums > & TimerTriggerSource

  Sets the internal camera signal used to trigger the selected timer.

- GenApi::IEnumerationT< Basler_GigECameraParams::TimerTriggerActivationEnums > & TimerTriggerActivation

  Sets the type of signal transistion that will start the timer.

- GenApi::IEnumerationT< Basler_GigECameraParams::LUTSelectorEnums > & LUTSelector

  Selects the lookup table (LUT) to configure. Once a LUT has been selected, all changes to the LUT settings will be applied to the selected LUT.

- GenApi::IBoolean & LUTEnable

  Enables the selected LUT.

- GenApi::IInteger & LUTIndex

  Sets the LUT element to access.

- GenApi::IInteger & LUTValue

  Sets the value of the LUT element at the LUT index.

- GenApi::IRegister & LUTValueAll

  Accesses the entire content of the selected LUT in one chunk access.

- GenApi::IEnumerationT< Basler_GigECameraParams::UserSetSelectorEnums > & UserSetSelector

  Selects the configuration set to load, save, or configure. Once a configuration set has been selected, all changes to the configuration set settings will be applied to the selected configuration set.

- GenApi::ICommand & UserSetLoad

  Loads the selected configuration into the camera's volatile memory and makes it the active configuration set. Once the selected set is loaded, the parameters in the selected set will control the camera.

- GenApi::ICommand & UserSetSave

  Saves the current active configuration set into the selected user set.

- GenApi::IEnumerationT< Basler_GigECameraParams::UserSetDefaultSelectorEnums > & UserSetDefaultSelector

  Sets the configuration set to be used as the default startup set. The configuration set that has been selected as the default startup set will be loaded as the active set whenever the camera is powered on or reset.

- GenApi::IString & DeviceVendorName

  Indicates the name of the device's vendor.

- GenApi::IString & DeviceModelName

  Indicates the model name of the device.

- GenApi::IString & DeviceManufacturerInfo

  Provides additional information from the vendor about the device.

- GenApi::IString & DeviceVersion

  Indicates the version of the device.

- GenApi::IString & DeviceFirmwareVersion

  Indicates the version of the device's firmware and software.

- GenApi::IString & DeviceID

  A unique identifier for the device such as a serial number or a GUID.

- GenApi::IString & DeviceUserID

  A device ID that is user programmable.

- GenApi::IEnumerationT< Basler_GigECameraParams::DeviceScanTypeEnums > & DeviceScanType

  Indicates the scan type of the device's sensor.

- GenApi::ICommand & DeviceReset

  Immediately resets and reboots the device.

- GenApi::IInteger & SensorWidth

  Indicates the width of the camera's sensor in pixels.

- GenApi::IInteger & SensorHeight

  Indicates the height of the camera's sensor in pixels.

- GenApi::IInteger & WidthMax

  Indicates the maximum allowed width of the image in pixels.

- GenApi::IInteger & HeightMax

  Indicates the maximum allowed height of the image in pixels.

- GenApi::IEnumerationT< Basler_GigECameraParams::TemperatureSelectorEnums > & TemperatureSelector

  TODO: Lists the temperature sources available for readout.

- GenApi::IFloat & TemperatureAbs

  TODO: Shows the current temperature of the selected target in degrees celcius.

- GenApi::IEnumerationT< Basler_GigECameraParams::ParameterSelectorEnums > & Parameter-Selector

  Selects the parameter to configure. Once a parameter has been selected, all changes made using the Remove Limits feature will be applied to the selected parameter.

- GenApi::IBoolean & RemoveLimits

  Removes the factory-set limits of the selected parameter.

- GenApi::IBoolean & ChunkModeActive

  Enables the chunk mode.

- GenApi::IEnumerationT< Basler_GigECameraParams::ChunkSelectorEnums > & ChunkSelector

  Selects the chunk to configure. Once a chunk has been selected, all changes to the chunk settings will be applied to the selected chunk.

- GenApi::IBoolean & ChunkEnable

  Enables the inclusion of the selected chunk in the payload data.

- GenApi::IInteger & ChunkStride

  Indicates the number of bytes of data between the beginning of one line in the acquired image and the beginning of the next line in the acquired image.

- GenApi::IInteger & ChunkOffsetX

  Indicates the X offset of the area of interest represented in the acquired image.

- GenApi::IInteger & ChunkOffsetY

  Indicates the Y offset of the area of interest represented in the acquired image.

- GenApi::IInteger & ChunkWidth

  Indicates the widtth of the area of interest represented in the acquired image.

- GenApi::IInteger & ChunkHeight

  Indicates the height of the area of interest represented in the acquired image.

- GenApi::IInteger & ChunkDynamicRangeMin

  Indicates the minimum possible pixel value in the acquired image.

- GenApi::IInteger & ChunkDynamicRangeMax

  Indicates the maximum possible pixel value in the acquired image.

- GenApi::IEnumerationT< Basler_GigECameraParams::ChunkPixelFormatEnums > & ChunkPixelFormat

  Indicates the format of the pixel data in the acquired image.

- GenApi::IInteger & ChunkTimestamp

  Indicates the value of the timestamp when the image was acquired.

- GenApi::IInteger & ChunkFramecounter

  Indicates the value of the frame counter when the image was acquired.

- GenApi::IInteger & ChunkLineStatusAll

  A bit field that indicates the status of all of the camera's input and output lines when the image was acquired.

- GenApi::IInteger & ChunkTriggerinputcounter

  Indicates the value of the trigger input counter when the image was acquired.

- GenApi::IInteger & ChunkPayloadCRC16

  Indicates the value of CRC checksum.

- GenApi::IEnumerationT< Basler_GigECameraParams::EventSelectorEnums > & EventSelector

  Selects the type of event to configure. Once an event has been selected, all changes to the event settings will be applied to the selected event.

- GenApi::IEnumerationT< Basler_GigECameraParams::EventNotificationEnums > & EventNotification

  Sets the notification type that will be sent to the host application for the selected event.

- GenApi::IInteger & ExposureEndEventStreamChannelIndex

  Indicates the stream channel index for an exposure end event.

- GenApi::IInteger & ExposureEndEventFrameID

  Indicates the frame ID for an exposure end event.

- GenApi::IInteger & ExposureEndEventTimestamp

  Indicates the time stamp for an exposure end event.

- GenApi::IInteger & EventOverrunEventStreamChannelIndex

  Indicates the stream channel index for an event overrun event.

- GenApi::IInteger & EventOverrunEventFrameID

  Indicates the frame ID for an event overrun event.

- GenApi::IInteger & EventOverrunEventTimestamp

Indicates the time stamp for an event overrun event.

- GenApi::IInteger & PayloadSize

  Size of the payload in bytes.

- GenApi::IInteger & GevVersionMajor

  Indicates the major version number of the GigE Vision specification supported by this device.

- GenApi::IInteger & GevVersionMinor

  Indicates the minor version number of the GigE Vision specification supported by this device.

- GenApi::IBoolean & GevDeviceModeIsBigEndian

  Indicates the endianess of the bootstrap registers.

- GenApi::IInteger & GevDeviceModeCharacterSet

  Indictes the character set.

- GenApi::IEnumerationT< Basler_GigECameraParams::GevInterfaceSelectorEnums > & GevInterfaceSelector

  Selects the physical network interface to configure. Once a network interface has been selected, all changes to the network interface settings will be applied to the selected interface.

- GenApi::IInteger & GevMACAddress

  Indicates the MAC address for the selected network interface.

- GenApi::IBoolean & GevSupportedIPConfigurationLLA

  Indicates whether the selected network interface supports auto IP addressing (also known as LLA).

- GenApi::IBoolean & GevSupportedIPConfigurationDHCP

  Indicates whether the selected network interface supports DHCP IP addressing.

- GenApi::IBoolean & GevSupportedIPConfigurationPersistentIP

  Indicates whether the selected network interface supports fixed IP addressing (also known as persistent IP addressing).

- GenApi::IInteger & GevCurrentIPConfiguration

  Sets the current IP configuration of the selected network interface.

- GenApi::IInteger & GevCurrentIPAddress

  Indicates the current IP address for the selected network interface.

- GenApi::IInteger & GevCurrentSubnetMask

  Indicates the current subnet mask for the selected network interface.

- GenApi::IInteger & GevCurrentDefaultGateway

  Indicates the current default gateway for the selected network interface.

- GenApi::IInteger & GevPersistentIPAddress

  If fixed (persistent) IP addressing is supported by the device and enabled, sets the fixed IP address for the selected network interface.

- GenApi::IInteger & GevPersistentSubnetMask

  If fixed (persistent) IP addressing is supported by the device and enabled, sets the fixed subnet mask for the selected network interface.

- GenApi::IInteger & GevPersistentDefaultGateway

  If fixed (persistent) IP addressing is supported by the device and enabled, sets the fixed default gateway for the selected network interface.

- GenApi::IInteger & GevLinkSpeed

  Indicates the connection speed in Mbps for the selected network interface.

- GenApi::IString & GevFirstURL

  Indicates the first URL to the XML device description file.

- GenApi::IString & GevSecondURL

  Indicates the second URL to the XML device description file.

- GenApi::IInteger & GevNumberOfInterfaces

  Indicates the number of network interfaces on the device.

- GenApi::IInteger & GevMessageChannelCount

  Indicates the number of message channels supported by the device.

- GenApi::IInteger & GevStreamChannelCount

  Indicates the number of stream channels supported by the device.

- GenApi::IBoolean & GevSupportedOptionalCommandsEVENTDATA

  Indicates whether EVENTDATA_CMD and EVENTDATA_ACK are supported.

- GenApi::IBoolean & GevSupportedOptionalCommandsEVENT

  Indicates whether EVENT_CMD and EVENT_ACK are supported.

- GenApi::IBoolean & GevSupportedOptionalCommandsPACKETRESEND

  Indicates whether PACKETRESEND_CMD is supported.

- GenApi::IBoolean & GevSupportedOptionalCommandsWRITEMEM

  Indicates whether WRITEMEM_CMD and WRITEMEM_ACK are supported.

- GenApi::IBoolean & GevSupportedOptionalCommandsConcatenation

  Indicates whether multiple operations in a single message are supported.

- GenApi::IInteger & GevHeartbeatTimeout

  Sets the heartbeat timeout in milliseconds.

- GenApi::IInteger & GevTimestampTickFrequency

  Indicates the number of timestamp clock ticks in 1 second.

- GenApi::ICommand & GevTimestampControlLatch

  Latches the current timestamp value of the device.

- GenApi::ICommand & GevTimestampControlReset

  Resets the timestamp value for the device.

- GenApi::ICommand & GevTimestampControlLatchReset

  Resets the timestamp control latch.

- GenApi::IInteger & GevTimestampValue

  Indicates the latched value of the timestamp. (The timestamp must first be latched using the Timestamp Control Latch command.).

- GenApi::IEnumerationT< Basler_GigECameraParams::GevCCPEnums > & GevCCP

  Sets the control channel privilege feature.

- GenApi::IEnumerationT< Basler_GigECameraParams::GevStreamChannelSelectorEnums > & GevStreamChannelSelector

  Selects the stream channel to configure. Once a stream channel has been selected, all changes to the stream channel settings will be applied to the selected stream channel.

- GenApi::IInteger & GevSCPInterfaceIndex

  Sets the index of the network interface to use.

- GenApi::IInteger & GevSCDA

  Sets the stream channel destination IPv4 address for the selected stream channel.

- GenApi::IInteger & GevSCPHostPort

  Sets the port to which the device must send data streams.

- GenApi::IInteger & GevSCPSPacketSize

  Sets the packet size in bytes for the selected stream channel.

- GenApi::IInteger & GevSCPD

  Sets the inter-packet delay (in ticks) for the selected stream channel.

- GenApi::IInteger & GevSCFTD

  Sets the frame transfer start delay (in ticks) for the selected stream channel.

- GenApi::IInteger & GevSCBWR

  Sets a percentage of the Ethernet bandwidth assigned to the camera to be held in reserve. The reserve is used for packet resends and control data transmissions.

- GenApi::IInteger & GevSCBWRA

  Sets a multiplier for the Bandwidth Reserve parameter. The multiplier is used to establish an extra pool of reserved bandwidth that can be used if an unusually large burst of packet resends is needed.

- GenApi::IInteger & GevSCBWA

  Indicates the bandwidth (in bytes per second) that will be used by the camera to transmit image and chunk feature data and to handle resends and control data transmissions.

- GenApi::IInteger & GevSCDMT

  Indicates the maximum amount of data (in bytes per second) that the camera could generate given its current settings and ideal conditions, i.e., unlimited bandwidth and no packet resends.

- GenApi::IInteger & GevSCDCT

  Indicates the actual bandwidth (in bytes per second) that the camera will use to transmit image data and chunk data given the current AOI settings, chunk feature settings, and the pixel format setting.

- GenApi::IInteger & GevSCFJM

  Indicates the maximum time (in ticks) that the next frame transmission could be delayed due to a burst of resends.

- GenApi::IInteger & TLParamsLocked

  Indicates whether a live grab is under way.

4.49.1 Detailed Description

Basler GigEVision camera interface.

4.49.2 Constructor & Destructor Documentation

4.49.2.1 Pylon::CBaslerGigECamera::CBaslerGigECamera (IPylonDevice ∗) [inline]

Creates a camera object that takes the ownership over an pylon device.

When having the ownership, the destructor of this camera object destroys the pylon device the camera object is attached to.

4.49.3 Member Function Documentation

4.49.3.1 void Pylon::CBaslerGigECamera::Attach (IPylonDevice ∗ *pDevice*, bool *takeOwnership* = true) [inline]

Attach a the camera object to a pylon device.

It is not allowed to call Attach when the camera object is already attache!

When having the ownership, the destructor of this camera object destroys the pylon device the camera object is attached to. Otherwise, the pylon device object remains valid when the camera object has been destroyed.

4.49.4 Member Data Documentation

4.49.4.1 GenApi::ICommand& Pylon::CBaslerGigECamera::AcquisitionAbort

Immediately aborts the acquisition of images.

This command will immediately abort any image acquisition process that is currently in progress.

4.49.4.2 GenApi::IFloat& Pylon::CBaslerGigECamera::AcquisitionFrameRateAbs

If the acquisition frame rate feature is enabled, this value sets the camera's acquisition frame rate in frames per second.

Sets the "absolute" value of the acquisition frame rate. The "absolute" value is a float value that sets the acquisition frame rate in frames per second.

4.49.4.3 GenApi::IBoolean& Pylon::CBaslerGigECamera::AcquisitionFrameRateEnable

Enables setting the camera's acquisition frame rate to a specified value.

This boolean value enables setting the camera's acquisition frame rate to a specified value.

4.49.4.4 GenApi::IFloat& Pylon::CBaslerGigECamera::AcquisitionLineRateAbs

Sets the camera's acquisition line rate in lines per second.

Sets the "absolute" value of the acquisition line rate. The "absolute" value is a float value that sets the acquisition line rate in lines per second.

4.49.4.5 GenApi::IEnumerationT<Basler_GigECameraParams::AcquisitionModeEnums >& Pylon::CBaslerGigECamera::AcquisitionMode

Sets the image acquisition mode.

This enumeration sets the image acquisition mode.

4.49.4.6 GenApi::ICommand& Pylon::CBaslerGigECamera::AcquisitionStart

Starts the acquisition of images.

This command starts the acquisition of images. If the camera is set for single frame acquisition, it will start acquisition of one frame. If the camera is set for continuous frame acquisition, it will start continuous acquisition of frames.

4.49.4.7 GenApi::ICommand& Pylon::CBaslerGigECamera::AcquisitionStop

Stops the acquisition of images.

If the camera is set for continuous image acquisition and acquisition has been started, this command stops the acquisition of images.

4.49.4.8 GenApi::IFloat& Pylon::CBaslerGigECamera::BalanceRatioAbs

Sets the value of the selected balance ratio control as a float.

This value sets the selected balance ratio control as a float value.

4.49.4.9 GenApi::IInteger& Pylon::CBaslerGigECamera::BalanceRatioRaw

Sets the value of the selected balance ratio control as an integer.

This value sets the selected balance ratio control as an integer.

4.49.4.10 GenApi::IEnumerationT<Basler_GigECameraParams::BalanceRatioSelectorEnums >& Pylon::CBaslerGigECamera::BalanceRatioSelector

Selects a balance ratio to configure. Once a balance ratio control has been selected, all changes to the balance ratio settings will be applied to the selected control.

This enumeration selects a balance ratio control to configuration. Once a balance ratio control has been selected, all changes to the balance ratio settings will be applied to the selected control.

4.49.4.11 GenApi::IInteger& Pylon::CBaslerGigECamera::BinningHorizontal

Sets the number of adjacent horizontal pixes to be summed.

Sets the number of binned adjacent horizontal pixels. Their charges will be summed and reported out of the camera as a single pixel.

4.49.4.12 GenApi::IInteger& Pylon::CBaslerGigECamera::BinningVertical

Sets the number of adjacent vertical pixes to be summed.

Sets the number of binned adjacent vertical pixels. Their charges will be summed and reported out of the camera as a single pixel.

4.49.4.13 GenApi::IFloat& Pylon::CBaslerGigECamera::BlackLevelAbs

Sets the value of the selected black level control as a float.

This value sets the selected black level control as a float value.

4.49.4.14 GenApi::IInteger& Pylon::CBaslerGigECamera::BlackLevelRaw

Sets the value of the selected black level control as an integer.

This value sets the selected black level control as an integer.

4.49.4.15 GenApi::IEnumerationT<Basler_GigECameraParams::BlackLevelSelectorEnums >& Pylon::CBaslerGigECamera::BlackLevelSelector

Selcts a black level control to configure. Once a black level control has been selected, all changes to the black level settings will be applied to the selected control.

This enumeration selects the black level control to configure. Once a black level control has been selected, all changes to the black level settings will be applied to the selected control.

4.49.4.16 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkDynamicRangeMax

Indicates the maximum possible pixel value in the acquired image.

This value indicates indicates the maximum possible pixel value acquired in the image

4.49.4.17 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkDynamicRangeMin

Indicates the minimum possible pixel value in the acquired image.

This value indicates the minimum possible pixel value in the acquired image.

4.49.4.18 GenApi::IBoolean& Pylon::CBaslerGigECamera::ChunkEnable

Enables the inclusion of the selected chunk in the payload data.

This boolean value enables the inclusion of the selected chunk in the payload data.

4.49.4.19 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkFramecounter

Indicates the value of the frame counter when the image was acquired.

This integer indicates the value of the frame counter when the image was acquired.

4.49.4.20 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkHeight

Indicates the height of the area of interest represented in the acquired image.

This value Indicates the height of the area of interest represented in the acquired image.

4.49.4.21 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkLineStatusAll

A bit field that indicates the status of all of the camera's input and output lines when the image was acquired.

This value is a bit field that indicates the status of all of the camera's input and output lines when the image was acquired.

4.49.4.22 GenApi::IBoolean& Pylon::CBaslerGigECamera::ChunkModeActive

Enables the chunk mode.

This boolean value enables the camera's chunk mode.

4.49.4.23 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkOffsetX

Indicates the X offset of the area of interest represented in the acquired image.

This value Indicates the X offset of the area of interest represented in the acquired image.

4.49.4.24 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkOffsetY

Indicates the Y offset of the area of interest represented in the acquired image.

This value Indicates the Y offset of the area of interest represented in the acquired image.

4.49.4.25 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkPayloadCRC16

Indicates the value of CRC checksum.

This integer indicates the value of CRC checksum.

4.49.4.26 GenApi::IEnumerationT<Basler_GigECameraParams::ChunkPixelFormatEnums >& Pylon::CBaslerGigECamera::ChunkPixelFormat

Indicates the format of the pixel data in the acquired image.

This enumeration lists the pixel formats that can be indicated by the pixel format chunk.

4.49.4.27 GenApi::IEnumerationT<Basler_GigECameraParams::ChunkSelectorEnums >& Pylon::CBaslerGigECamera::ChunkSelector

Selects the chunk to configure. Once a chunk has been selected, all changes to the chunk settings will be applied to the selected chunk.

This enumeration selects the chunk to configure. Once a chunk has been selected, all changes to the chunk settings will be applied to the selected chunk.

4.49.4.28 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkStride

Indicates the number of bytes of data between the beginning of one line in the acquired image and the beginning of the next line in the acquired image.

This value indicates the number of bytes of data between the beginning of one line in the acquired image and the beginning of the next line in the acquired image.

4.49.4.29 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkTimestamp

Indicates the value of the timestamp when the image was acquired.

This integer indicates the value of the timestamp when the image was acquired.

4.49.4.30 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkTriggerinputcounter

Indicates the value of the trigger input counter when the image was acquired.

This integer indicates the value of the trigger input counter when the image was acquired.

4.49.4.31 GenApi::IInteger& Pylon::CBaslerGigECamera::ChunkWidth

Indicates the widtth of the area of interest represented in the acquired image.

This value Indicates the width of the area of interest represented in the acquired image.

4.49.4.32 GenApi::IString& Pylon::CBaslerGigECamera::DeviceFirmwareVersion

Indicates the version of the device's firmware and software.

This is a read only element. It is a string that indicates the version of the device's firmware and software.

4.49.4.33 GenApi::IString& Pylon::CBaslerGigECamera::DeviceID

A unique identifier for the device such as a serial number or a GUID.

This is a read only element. It is a string that provides a unique identifier for the device such as a serial number or a GUID.

4.49.4.34 GenApi::IString& Pylon::CBaslerGigECamera::DeviceManufacturerInfo

Provides additional information from the vendor about the device.

This is a read only element. It is a string that provides additional information from the vendor about the camera.

4.49.4.35 GenApi::IString& Pylon::CBaslerGigECamera::DeviceModelName

Indicates the model name of the device.

This is a read only element. It is a text description that indicates the model name of the device.

4.49.4.36 GenApi::ICommand& Pylon::CBaslerGigECamera::DeviceReset

Immediately resets and reboots the device.

This is a command that immediately resets and reboots the device.

4.49.4.37 GenApi::IEnumerationT<Basler_GigECameraParams::DeviceScanTypeEnums >& Pylon::CBaslerGigECamera::DeviceScanType

Indicates the scan type of the device's sensor.

This enumeration lists the possible scan types for the sensor in the device.

4.49.4.38 GenApi::IString& Pylon::CBaslerGigECamera::DeviceUserID

A device ID that is user programmable.

This is a read only element. It is a user programmable string.

4.49.4.39 GenApi::IString& Pylon::CBaslerGigECamera::DeviceVendorName

Indicates the name of the device's vendor.

This is a read only element. It is a text description that indicates the name of the device's vendor.

4.49.4.40 GenApi::IString& Pylon::CBaslerGigECamera::DeviceVersion

Indicates the version of the device.

This is a read only element. It is a string that indicates the version of the device.

4.49.4.41 GenApi::IInteger& Pylon::CBaslerGigECamera::DigitalShift

Sets the value of the selected digital shift control.

This value sets the selected digital shift control

4.49.4.42 GenApi::IEnumerationT<Basler_GigECameraParams::EventNotificationEnums >& Pylon::CBaslerGigECamera::EventNotification

Sets the notification type that will be sent to the host application for the selected event.

This enumeration sets the notification type that will be sent to the host application for the selected event.

4.49.4.43 GenApi::IInteger& Pylon::CBaslerGigECamera::EventOverrunEventFrameID

Indicates the frame ID for an event overrun event.

This enumeration value indicates the frame ID for an event overrun event

4.49.4.44 GenApi::IInteger& Pylon::CBaslerGigECamera::EventOverrunEventStreamChannelIndex

Indicates the stream channel index for an event overrun event.

This enumeration value indicates the stream channel index for an event overrun event

4.49.4.45 GenApi::IInteger& Pylon::CBaslerGigECamera::EventOverrunEventTimestamp

Indicates the time stamp for an event overrun event.

This enumeration value indicates the time stamp for an event overrun event

4.49.4.46 GenApi::IEnumerationT<Basler_GigECameraParams::EventSelectorEnums >& Pylon::CBaslerGigECamera::EventSelector

Selects the type of event to configure. Once an event has been selected, all changes to the event settings will be applied to the selected event.

This enumeration selects the type of event to configure. Once an event has been selected, all changes to the event settings will be applied to the selected event.

4.49.4.47 GenApi::IInteger& Pylon::CBaslerGigECamera::ExposureEndEventFrameID

Indicates the frame ID for an exposure end event.

This enumeration value indicates the frame ID for an exposure end event.

4.49.4.48 GenApi::IInteger& Pylon::CBaslerGigECamera::ExposureEndEventStreamChannelIndex

Indicates the stream channel index for an exposure end event.

This enumeration value indicates the stream channel index for an exposure end event.

4.49.4.49 GenApi::IInteger& Pylon::CBaslerGigECamera::ExposureEndEventTimestamp

Indicates the time stamp for an exposure end event.

This enumeration value indicates the time stamp for an exposure end event.

4.49.4.50 GenApi::IEnumerationT<Basler_GigECameraParams::ExposureModeEnums >& Pylon::CBaslerGigECamera::ExposureMode

Sets the exposure mode.

This enumeration sets the exposure mode.

4.49.4.51 GenApi::IFloat& Pylon::CBaslerGigECamera::ExposureTimeAbs

Directly sets the camera's exposure time in microseconds.

This float value sets the camera's exposure time in microseconds.

4.49.4.52 GenApi::IFloat& Pylon::CBaslerGigECamera::ExposureTimeBaseAbs

Sets the time base (in microseconds) that is used when the exposure time is set with the "exposure time raw" setting.

This float value sets the time base (in microseconds) that is used when the exposure time is set with the "raw" setting.

4.49.4.53 GenApi::IBoolean& Pylon::CBaslerGigECamera::ExposureTimeBaseAbsEnable

Enables the use of the exposure time base.

This value enables the use of the exposure time base.

4.49.4.54 GenApi::IInteger& Pylon::CBaslerGigECamera::ExposureTimeRaw

Sets the "raw" exposure time. Actual exposure time = raw exposure setting x exposure time base abs setting.

This value sets an integer that will be used as a multiplier for the exposure timebase. The actual exposure time equals the current exposure time raw setting times the current exposure time base abs setting.

4.49.4.55 GenApi::IFloat& Pylon::CBaslerGigECamera::GainAbs

This is a float value that sets the selected gain control in dB.

Sets the "absolute" value of the selected gain control. The "absolute" value is a float value that sets the selected gain control in dB.

4.49.4.56 GenApi::IInteger& Pylon::CBaslerGigECamera::GainRaw

This is an integer value that sets the selected gain control in device specific units.

Sets the "raw" value of the selected gain control. The "raw" value is an integer value that sets the selected gain control in units specific to the camera.

4.49.4.57 GenApi::IEnumerationT<Basler_GigECameraParams::GainSelectorEnums >& Pylon::CBaslerGigECamera::GainSelector

Selects the gain control to configure. Once a gain control has been selected, all changes to the gain settings will be applied to the selected control.

This enumeration selects the gain control to configure. Once a gain control has been selected, all changes to the gain settings will be applied to the selected control.

4.49.4.58 GenApi::IFloat& Pylon::CBaslerGigECamera::Gamma

This feature is used to perform gamma correction of pixel intensity.

This feature is used to perform gamma correction of pixel intensity. This is typically used to compensate for non-linearity of the display system (such as CRT).

4.49.4.59 GenApi::IBoolean& Pylon::CBaslerGigECamera::GammaEnable

Enables the gamma correction.

This boolean value enables the gamma correction.

4.49.4.60 GenApi::IEnumerationT<Basler_GigECameraParams::GevCCPEnums >& Pylon::CBaslerGigECamera::GevCCP

Sets the control channel privilege feature.

This enumeration sets the control channel privilege feature.

4.49.4.61 GenApi::IInteger& Pylon::CBaslerGigECamera::GevCurrentDefaultGateway

Indicates the current default gateway for the selected network interface.

This is a read only element. It indicates the current default gateway for the selected network interface.

4.49.4.62 GenApi::IInteger& Pylon::CBaslerGigECamera::GevCurrentIPAddress

Indicates the current IP address for the selected network interface.

This is a read only element. It indicates the current IP address for the selected network interface.

4.49.4.63 GenApi::IInteger& Pylon::CBaslerGigECamera::GevCurrentIPConfiguration

Sets the current IP configuration of the selected network interface.

This value sets the IP configuration of the selected network interface, i.e., fixed IP, DHCP, auto IP.

4.49.4.64 GenApi::IInteger& Pylon::CBaslerGigECamera::GevCurrentSubnetMask

Indicates the current subnet mask for the selected network interface.

This is a read only element. It indicates the current subnet mask for the selected network interface.

4.49.4.65 GenApi::IInteger& Pylon::CBaslerGigECamera::GevDeviceModeCharacterSet

Indictes the character set.

This is a read only element. Its value indicates the character set. 1 = UTF8

4.49.4.66 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevDeviceModeIsBigEndian

Indicates the endianess of the bootstrap registers.

This is a read only element. It indicates the endianess of the bootstrap registers. True = big endian.

4.49.4.67 GenApi::IString& Pylon::CBaslerGigECamera::GevFirstURL

Indicates the first URL to the XML device description file.

This is a read only element. It indicates the first URL to the XML device description file.

4.49.4.68 GenApi::IInteger& Pylon::CBaslerGigECamera::GevHeartbeatTimeout

Sets the heartbeat timeout in milliseconds.

This value sets the heartbeat timeout in milliseconds.

4.49.4.69 GenApi::IEnumerationT<Basler_GigECameraParams::GevInterfaceSelectorEnums >& Pylon::CBaslerGigECamera::GevInterfaceSelector

Selects the physical network interface to configure. Once a network interface has been selected, all changes to the network interface settings will be applied to the selected interface.

This selects the physical network interface to configure. Once a network interface has been selected, all changes to the network interface settings will be applied to the selected interface.

4.49.4.70 GenApi::IInteger& Pylon::CBaslerGigECamera::GevLinkSpeed

Indicates the connection speed in Mbps for the selected network interface.

This is a read only element. It indicates the connection speed in Mbps for the selected network interface.

4.49.4.71 GenApi::IInteger& Pylon::CBaslerGigECamera::GevMACAddress

Indicates the MAC address for the selected network interface.

This is a read only element. It indicates the MAC address for the selected network interface.

4.49.4.72 GenApi::IInteger& Pylon::CBaslerGigECamera::GevMessageChannelCount

Indicates the number of message channels supported by the device.

This is a read only element. It indicates the number of message channels supported by the device.

4.49.4.73 GenApi::IInteger& Pylon::CBaslerGigECamera::GevNumberOfInterfaces

Indicates the number of network interfaces on the device.

This is a read only element. It indicates the number of network interfaces on the device.

4.49.4.74 GenApi::IInteger& Pylon::CBaslerGigECamera::GevPersistentDefaultGateway

If fixed (persistent) IP addressing is supported by the device and enabled, sets the fixed default gateway for the selected network interface.

This value sets the fixed default gateway for the selected network interface (if fixed IP addressing is supported by the device and enabled).

4.49.4.75 GenApi::IInteger& Pylon::CBaslerGigECamera::GevPersistentIPAddress

If fixed (persistent) IP addressing is supported by the device and enabled, sets the fixed IP address for the selected network interface.

This value sets the fixed IP address for the selected network interface (if fixed IP addressing is supported by the device and enabled).

4.49.4.76 GenApi::IInteger& Pylon::CBaslerGigECamera::GevPersistentSubnetMask

If fixed (persistent) IP addressing is supported by the device and enabled, sets the fixed subnet mask for the selected network interface.

This value sets the fixed subnet mask for the selected network interface (if fixed IP addressing is supported by the device and enabled).

4.49.4.77 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCBWA

Indicates the bandwidth (in bytes per second) that will be used by the camera to transmit image and chunk feature data and to handle resends and control data transmissions.

This value indicates the base bandwidth in bytes per second that will be used by the camera to transmit image and chunk feature data and to handle resends and control data transmissions. This parameter represents a combination of the packet size and the inter-packet delay.

4.49.4.78 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCBWR

Sets a percentage of the Ethernet bandwidth assigned to the camera to be held in reserve. The reserve is used for packet resends and control data transmissions.

This value reserves a portion of Ethernet bandwidth assigned to the camera for packet resends and for the transmission of control data between the camera and the host PC. The setting is expressed as a percentage

of the bandwidth assigned parameter. For example, if the Bandwidth Assigned parameter indicates that 30 MBytes/s have been assigned to the camera and the Bandwidth Reserve parameter is set to 5%, then the bandwidth reserve will be 1.5 MBytes/s.

4.49.4.79 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCBWRA

Sets a multiplier for the Bandwidth Reserve parameter. The multiplier is used to establish an extra pool of reserved bandwidth that can be used if an unusually large burst of packet resends is needed.

This value sets a multiplier for the Bandwidth Reserve parameter. The multiplier is used to establish an extra pool of reserved bandwidth that can be used if an unusually large burst of packet resends is needed.

4.49.4.80 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCDA

Sets the stream channel destination IPv4 address for the selected stream channel.

This value sets the stream channel destination IPv4 address for the selected stream channel. The destination can be a unicast or a multicast.

4.49.4.81 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCDCT

Indicates the actual bandwidth (in bytes per second) that the camera will use to transmit image data and chunk data given the current AOI settings, chunk feature settings, and the pixel format setting.

This value indicates the actual bandwidth (in bytes per second) that the camera will use to transmit image data and chunk data given the current AOI settings, chunk feature settings, and the pixel format setting.

4.49.4.82 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCDMT

Indicates the maximum amount of data (in bytes per second) that the camera could generate given its current settings and ideal conditions, i.e., unlimited bandwidth and no packet resends.

This value indicates the maximum amount of data (in bytes per second) that the camera could generate given its current settings and ideal conditions, i.e., unlimited bandwidth and no packet resends.

4.49.4.83 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCFJM

Indicates the maximum time (in ticks) that the next frame transmission could be delayed due to a burst of resends.

If the Bandwidth Reserve Accumulation parameter is set to a high value, the camera can experience periods where there is a large burst of data resends. This burst of resends will delay the start of transmission of the next acquired image. The Frame Max Jitter parameter indicates the maximum time in ticks that the next frame transmission could be delayed due to a burst of resends.

4.49.4.84 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCFTD

Sets the frame transfer start delay (in ticks) for the selected stream channel.

This value sets the frame transfer delay for the selected stream channel. This value sets a delay betweem when the camera would normally begin transmitted an acquired image (frame) and when it actually begins transmitting the acquired image.

4.49.4.85 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCPD

Sets the inter-packet delay (in ticks) for the selected stream channel.

This value sets a delay between the transmission of each packet for the selected stream channel. The delay is measured in ticks.

4.49.4.86 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCPHostPort

Sets the port to which the device must send data streams.

This value sets the port to which the device must send data streams.

4.49.4.87 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCPInterfaceIndex

Sets the index of the network interface to use.

This value sets the index of the network interface to use.

4.49.4.88 GenApi::IInteger& Pylon::CBaslerGigECamera::GevSCPSPacketSize

Sets the packet size in bytes for the selected stream channel.

This value sets the packet size in bytes for the selected stream channel. Excludes data leader and data trailer. (The last packet may be smaller because the packet size is not necessarily a multiple of the block size for the stream channel.)

4.49.4.89 GenApi::IString& Pylon::CBaslerGigECamera::GevSecondURL

Indicates the second URL to the XML device description file.

This is a read only element. It indicates the second URL to the XML device description file.

4.49.4.90 GenApi::IInteger& Pylon::CBaslerGigECamera::GevStreamChannelCount

Indicates the number of stream channels supported by the device.

This is a read only element. It indicates the number of stream channels supported by the device.

4.49.4.91 GenApi::IEnumerationT<Basler_GigECameraParams::GevStreamChannelSelector-Enums >& Pylon::CBaslerGigECamera::GevStreamChannelSelector

Selects the stream channel to configure. Once a stream channel has been selected, all changes to the stream channel settings will be applied to the selected stream channel.

This enumeration selects the stream channels to configure. Once a stream channel has been selected, all changes to the stream channel settings will be applied to the selected stream channel.

4.49.4.92 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedIPConfigurationDHCP

Indicates whether the selected network interface supports DHCP IP addressing.

This is a read only element. It indicates whether the selected network interface supports DHCP IP addressing.

4.49.4.93 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedIPConfigurationLLA

Indicates whether the selected network interface supports auto IP addressing (also known as LLA).

This is a read only element. It indicates whether the selected network interface supports auto IP addressing (also known as LLA).

4.49.4.94 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedIPConfigurationPersistentIP

Indicates whether the selected network interface supports fixed IP addressing (also known as persistent IP addressing).

This is a read only element. It indicates whether the selected network interface supports fixed IP addressing (also known as persistent IP addressing).

4.49.4.95 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedOptionalCommandsConcatenation

Indicates whether multiple operations in a single message are supported.

This is a read only element. It indicates whether multiple operations in a single message are supported.

4.49.4.96 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedOptionalCommandsEVENT

Indicates whether EVENT_CMD and EVENT_ACK are supported.

This is a read only element. It indicates whether EVENT_CMD and EVENT_ACK are supported.

4.49.4.97 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedOptionalCommandsEVENTDATA

Indicates whether EVENTDATA_CMD and EVENTDATA_ACK are supported.

This is a read only element. It indicates whether EVENTDATA_CMD and EVENTDATA_ACK are supported.

4.49.4.98 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedOptionalCommandsPACKETRESEND

Indicates whether PACKETRESEND_CMD is supported.

This is a read only element. It indicates whether PACKETRESEND_CMD is supported.

4.49.4.99 GenApi::IBoolean& Pylon::CBaslerGigECamera::GevSupportedOptionalCommandsWRITEMEM

Indicates whether WRITEMEM_CMD and WRITEMEM_ACK are supported.

This is a read only element. It indicates whether WRITEMEM_CMD and WRITEMEM_ACK are supported

4.49.4.100 GenApi::ICommand& Pylon::CBaslerGigECamera::GevTimestampControlLatch

Latches the current timestamp value of the device.

This command latches the current timestamp value of the device.

4.49.4.101 GenApi::ICommand& Pylon::CBaslerGigECamera::GevTimestampControlLatchReset

Resets the timestamp control latch.

This command resets the timestamp control latch.

4.49.4.102 GenApi::ICommand& Pylon::CBaslerGigECamera::GevTimestampControlReset

Resets the timestamp value for the device.

This command resets the timestamp value for the device

4.49.4.103 GenApi::IInteger& Pylon::CBaslerGigECamera::GevTimestampTickFrequency

Indicates the number of timestamp clock ticks in 1 second.

This is a read only element. It indicates the number of timestamp clock ticks in 1 second.

4.49.4.104 GenApi::IInteger& Pylon::CBaslerGigECamera::GevTimestampValue

Indicates the latched value of the timestamp. (The timestamp must first be latched using the Timestamp Control Latch command.).

This is a read only element. It indicates the latched value of the timestamp. (The timestamp must first be latched using the Timestamp Control Latch command.)

4.49.4.105 GenApi::IInteger& Pylon::CBaslerGigECamera::GevVersionMajor

Indicates the major version number of the GigE Vision specification supported by this device.

This is a read only element. It indicates the major version number of the GigE Vision specification supported by this device.

4.49.4.106 GenApi::IInteger& Pylon::CBaslerGigECamera::GevVersionMinor

Indicates the minor version number of the GigE Vision specification supported by this device.

This is a read only element. It indicates the minor version number of the GigE Vision specification supported by this device.

4.49.4.107 GenApi::IInteger& Pylon::CBaslerGigECamera::Height

Sets the height of the area of interest in pixels.

This value sets the height of the area of interest in pixels.

4.49.4.108 GenApi::IInteger& Pylon::CBaslerGigECamera::HeightMax

Indicates the maximum allowed height of the image in pixels.

This is a read only element. It is an integer that indicates maximum allowed height of the image in pixels taking into account any function that may limit the allowed height.

4.49.4.109 GenApi::IEnumerationT<Basler_GigECameraParams::LegacyBinningVerticalEnums >& Pylon::CBaslerGigECamera::LegacyBinningVertical

Sets the vertical binning feature.

This enumeration sets the vertical binning feature.

4.49.4.110 GenApi::IFloat& Pylon::CBaslerGigECamera::LineDebouncerTimeAbs

Sets the absolute value of the selected line debouncer time in microseconds.

Sets the absolute value of the selected line debouncer time in microseconds

4.49.4.111 GenApi::IInteger& Pylon::CBaslerGigECamera::LineDebouncerTimeRaw

Sets the raw value of the selected line debouncer time.

Sets the raw value of the selected line debouncer time

4.49.4.112 GenApi::IEnumerationT<Basler_GigECameraParams::LineFormatEnums >& Pylon::CBaslerGigECamera::LineFormat

Sets the electrical configuration of the selected line.

This feature controls the current electrical format of the selected physical input or output Line. Line Format can take any of the following values: No Connect: The Line is not connected. Tri-state: The Line is currently in Tri-state mode (Not driven). TTL: The Line is currently accepting or sending TTL level signals. LVDS: The Line is currently accepting or sending LVDS level signals. RS-422: The Line is currently accepting or sending RS-422 level signals. Opto-coupled: The Line is Opto-coupled.

4.49.4.113 GenApi::IBoolean& Pylon::CBaslerGigECamera::LineInverter

Enables the inverter function for the selected line.

This boolean value enables the inverter function for the selected line.

4.49.4.114 GenApi::IEnumerationT<Basler_GigECameraParams::LineModeEnums >& Pylon::CBaslerGigECamera::LineMode

Sets the mode for the selected line.

This feature controls whether the physical Line is used to Input or Output a signal. When a Line supports input and output mode, the default state is Input to avoid possible electrical contention. Line Mode can take

any of the following values: Input: The selected physical line is used to input an electrical signal. Output: The selected physical line is used to output an electrical signal.

4.49.4.115 GenApi::IEnumerationT<Basler_GigECameraParams::LineSelectorEnums >& Pylon::CBaslerGigECamera::LineSelector

Selects the I/O line to configure. Once a line has been selected, all changes to the line settings will be applied to the selected line.

This enumeration selects the I/O line to configure. Once a line has been selected, all changes to the line settings will be applied to the selected line.

4.49.4.116 GenApi::IEnumerationT<Basler_GigECameraParams::LineSourceEnums >& Pylon::CBaslerGigECamera::LineSource

Sets the source signal for the selected line (if the selected line is an output).

This enumeration selects the internally generated camera signal (source signal) for the selected line when the selected line is an output.

4.49.4.117 GenApi::IBoolean& Pylon::CBaslerGigECamera::LineStatus

Indicates the current logical state for the selected line.

This boolean value indicates the current logical state for the selected line at the time of polling.

4.49.4.118 GenApi::IInteger& Pylon::CBaslerGigECamera::LineStatusAll

A single bitfield indicating the current logical state of all available line signals at time of polling.

This integer value is a single bitfield that indicates the current logical state of all available lines at time of polling.

4.49.4.119 GenApi::IBoolean& Pylon::CBaslerGigECamera::LineTermination

TODO: Enables the termination function for the selected line.

TODO:This boolean value enables the termination function for the selected line.

4.49.4.120 GenApi::IBoolean& Pylon::CBaslerGigECamera::LUTEnable

Enables the selected LUT.

This boolean value enables the selected LUT.

4.49.4.121 GenApi::IInteger& Pylon::CBaslerGigECamera::LUTIndex

Sets the LUT element to access.

This value sets the LUT element to access. This value is used to index into a LUT array.

4.49.4.122 GenApi::IEnumerationT<Basler_GigECameraParams::LUTSelectorEnums >& Pylon::CBaslerGigECamera::LUTSelector

Selects the lookup table (LUT) to configure. Once a LUT has been selected, all changes to the LUT settings will be applied to the selected LUT.

This enumeration the lookup table (LUT) to configure. Once a LUT has been selected, all changes to the LUT settings will be applied to the selected LUT.

4.49.4.123 GenApi::IInteger& Pylon::CBaslerGigECamera::LUTValue

Sets the value of the LUT element at the LUT index.

This value sets the value of the LUT element at the LUT index.

4.49.4.124 GenApi::IInteger& Pylon::CBaslerGigECamera::OffsetX

Sets the X offset (left offset) of the area of interest in pixels.

This value sets the X offset (left offset) for the area of interest in pixels, i.e., the distance in pixels between the left side of the sensor and the left side of the image area.

4.49.4.125 GenApi::IInteger& Pylon::CBaslerGigECamera::OffsetY

Sets the Y offset (top offset) for the area of interest in pixels.

This value sets the Y offset (top offset) for the area of interest, i.e., the distance in pixels between the top of the sensor and the top of the image area.

4.49.4.126 GenApi::IEnumerationT<Basler_GigECameraParams::ParameterSelectorEnums >& Pylon::CBaslerGigECamera::ParameterSelector

Selects the parameter to configure. Once a parameter has been selected, all changes made using the Remove Limits feature will be applied to the selected parameter.

This enumeration selects the parameter to configure. Selects the parameter to configure. Once a parameter has been selected, all changes made using the Remove Limits feature will be applied to the selected parameter

4.49.4.127 GenApi::IInteger& Pylon::CBaslerGigECamera::PayloadSize

Size of the payload in bytes.

Size of the payload in bytes. This is the total number of bytes sent in the payload. Image data + chunk data if present. No packet headers.

4.49.4.128 GenApi::IEnumerationT<Basler_GigECameraParams::PixelColorFilterEnums >& Pylon::CBaslerGigECamera::PixelColorFilter

Indicates the alignment of the camera's Bayer filter to the pixels in the acquired images.

This is a read only feature. This enumeration provides a list of values that indicate the alignment of the camera's Bayer filter to the pixels in the acquired images.

4.49.4.129 GenApi::IInteger& Pylon::CBaslerGigECamera::PixelDynamicRangeMax

Indicates the maximum possible pixel value that could be transferred from the camera.

This a read only feature. It indicates the maximum possible pixel value that could be transferred from the camera.

4.49.4.130 GenApi::IInteger& Pylon::CBaslerGigECamera::PixelDynamicRangeMin

Indicates the minimum possible pixel value that could be transferred from the camera.

This a read only feature. It indicates the minimum possible pixel value that could be transferred from the camera.

4.49.4.131 GenApi::IEnumerationT<Basler_GigECameraParams::PixelFormatEnums >& Pylon::CBaslerGigECamera::PixelFormat

Sets the format of the pixel data transmitted for acquired images.

This enumeration sets the format of the pixel data transmitted for acquired images.

4.49.4.132 GenApi::IEnumerationT<Basler_GigECameraParams::PixelSizeEnums >& Pylon::CBaslerGigECamera::PixelSize

Indicates the depth of the pixel values in the image in bits per pixel.

This is a read only feature. This enumeration provides a list of values that indicate the depth of the pixel values in the acquired images in bits per pixel. This value will always be coherent with the pixel format setting.

4.49.4.133 GenApi::IBoolean& Pylon::CBaslerGigECamera::RemoveLimits

Removes the factory-set limits of the selected parameter.

Removes the factory-set limits of the selected parameter. Having removed the factory-set limits you may set the parameter within extended limits. These are only defined by technical restrictions. Note: Inferior image quality may result.

4.49.4.134 GenApi::IFloat& Pylon::CBaslerGigECamera::ResultingFrameRateAbs

Indicates the maximum allowed frame acquisition rate (in frames per second) given the current settings for the area of interest, exposure time, and bandwidth.

Indicates the "absolute" value of the maximum allowed acquisition frame rate. The "absolute" value is a float value that indicates the maximum allowed acquisition frame rate in frames per second given the current settings for the area of interest, exposure time, and bandwidth.

4.49.4.135 GenApi::IFloat& Pylon::CBaslerGigECamera::ResultingLineRateAbs

Indicates the maximum allowed line acquisition rate (in lines per second) given the current settings for the area of interest, exposure time, and bandwidth.

Indicates the "absolute" value of the maximum allowed acquisition line rate. The "absolute" value is a float value that indicates the maximum allowed acquisition line rate in lines per second given the current settings for the area of interest, exposure time, and bandwidth.

4.49.4.136 GenApi::IInteger& Pylon::CBaslerGigECamera::SensorHeight

Indicates the height of the camera's sensor in pixels.

This is a read only element. It is an integer that indicates the actual height of the camera's sensor in pixels.

4.49.4.137 GenApi::IInteger& Pylon::CBaslerGigECamera::SensorWidth

Indicates the width of the camera's sensor in pixels.

This is a read only element. It is an integer that indicates the actual width of the camera's sensor in pixels.

4.49.4.138 GenApi::IInteger& Pylon::CBaslerGigECamera::SpatialCorrection

Specifies the direction of imaging and the separation (consecutive numbers) of related line captures.

Specifies the direction of imaging and the separation (consecutive numbers) of related line captures. Related line captures will be combined.

Positive integer: The object will pass the top sensor line first.

Negative integer: The object will pass the bottom sensor line first.

In color cameras, the top sensor line is the green line, and the bottom sensor line is the blue line.

4.49.4.139 GenApi::IInteger& Pylon::CBaslerGigECamera::SpatialCorrectionAmount

TODO:.

TODO:

4.49.4.140 GenApi::IEnumerationT<Basler_GigECameraParams::SpatialCorrectionStartingLineEnums >& Pylon::CBaslerGigECamera::SpatialCorrectionStartingLine

TODO:.

TODO:

4.49.4.141 GenApi::IFloat& Pylon::CBaslerGigECamera::TemperatureAbs

TODO: Shows the current temperature of the selected target in degrees celcius.

TODO: Shows the current temperature of the selected target in degrees celcius

4.49.4.142 GenApi::IEnumerationT<Basler_GigECameraParams::TemperatureSelectorEnums >& Pylon::CBaslerGigECamera::TemperatureSelector

TODO: Lists the temperature sources available for readout.

TODO: Lists the temperature sources available for readout

4.49.4.143 GenApi::IEnumerationT<Basler_GigECameraParams::TestImageSelectorEnums >& Pylon::CBaslerGigECamera::TestImageSelector

Selecting a test image from the list will enable the test image.

This enumeration provides a list of the available test images. Selecting a test image from the list will enable the test image.

4.49.4.144 GenApi::IFloat& Pylon::CBaslerGigECamera::TimerDelayAbs

Directly sets the delay for the selected timer in microseconds.

This float value sets the delay for the selected timer in microseconds.

4.49.4.145 GenApi::IInteger& Pylon::CBaslerGigECamera::TimerDelayRaw

Sets the "raw" delay for the selected timer. Actual delay = raw timer delay setting x timer delay time base abs setting.

This value sets an integer that will be used as a multiplier for the timer delay timebase. The actual delay time equals the current timer delay raw setting times the current timer delay time base abs setting.

4.49.4.146 GenApi::IFloat& Pylon::CBaslerGigECamera::TimerDelayTimebaseAbs

Sets the time base (in microseconds) that is used when a timer delay is set with the "timer delay raw" setting.

This float value sets the time base (in microseconds) that is used when a timer delay is set with the "raw" setting.

4.49.4.147 GenApi::IFloat& Pylon::CBaslerGigECamera::TimerDurationAbs

Directly sets the duration for the selected timer in microseconds.

This float value sets the duration for the selected timer in microseconds.

4.49.4.148 GenApi::IInteger& Pylon::CBaslerGigECamera::TimerDurationRaw

Sets the "raw" duration for the selected timer. Actual duration = raw timer duration setting x timer duration time base abs setting.

This value sets an integer that will be used as a multiplier for the timer duration timebase. The actual duration time equals the current timer duration raw setting times the current timer duration time base abs setting.

4.49.4.149 GenApi::IFloat& Pylon::CBaslerGigECamera::TimerDurationTimebaseAbs

Sets the time base (in microseconds) that is used when a timer duration is set with the "timer duration raw" setting.

This float value sets the time base (in microseconds) that is used when a timer duration is set with the "raw" setting.

4.49.4.150 GenApi::IEnumerationT<Basler_GigECameraParams::TimerSelectorEnums >& Pylon::CBaslerGigECamera::TimerSelector

Selects the timer to configure. Once a timer has been selected, all changes to the timer settings will be applied to the selected timer.

This enumeration selects the timer to configure. . Once a timer has been selected, all changes to the timer settings will be applied to the selected timer.

4.49.4.151 GenApi::IEnumerationT<Basler_GigECameraParams::TimerTriggerActivationEnums >& Pylon::CBaslerGigECamera::TimerTriggerActivation

Sets the type of signal transistion that will start the timer.

This enumeration sets the type of signal transistion that will start the timer.

4.49.4.152 GenApi::IEnumerationT<Basler_GigECameraParams::TimerTriggerSourceEnums >& Pylon::CBaslerGigECamera::TimerTriggerSource

Sets the internal camera signal used to trigger the selected timer.

This enumeration sets the internal camera signal used to trigger the selected timer.

4.49.4.153 GenApi::IEnumerationT<Basler_GigECameraParams::TriggerActivationEnums >& Pylon::CBaslerGigECamera::TriggerActivation

Sets the signal transition needed to activate the selected trigger.

This enumeration sets the signal transition needed to activate the selected trigger.

4.49.4.154 GenApi::IEnumerationT<Basler_GigECameraParams::TriggerModeEnums >& Pylon::CBaslerGigECamera::TriggerMode

Sets the mode for the selected trigger.

This enumeration sets the trigger mode for the selected trigger.

4.49.4.155 GenApi::IEnumerationT<Basler_GigECameraParams::TriggerSelectorEnums >& Pylon::CBaslerGigECamera::TriggerSelector

Selects the trigger type to configure. Once a trigger type has been selected, all changes to the trigger settings will be applied to the selected trigger.

This enumeration selects the trigger type to configure. Once a trigger type has been selected, all changes to the trigger settings will be applied to the selected trigger.

4.49.4.156 GenApi::ICommand& Pylon::CBaslerGigECamera::TriggerSoftware

Generates a software trigger signal that is used when the trigger source is set to "software".

This command generates a software trigger signal. The software trigger signal will be used if the trigger source is set to "software".

4.49.4.157 GenApi::IEnumerationT<Basler_GigECameraParams::TriggerSourceEnums >& Pylon::CBaslerGigECamera::TriggerSource

Sets the signal source for the selected trigger.

This enumeration sets the signal source for the selected trigger.

4.49.4.158 GenApi::IEnumerationT<Basler_GigECameraParams::UserOutputSelectorEnums >& Pylon::CBaslerGigECamera::UserOutputSelector

Selects the user settable output signal to configure. Once a user settable output signal has been selected, all changes to the user settable output signal settings will be applied to the selected user settable output signal.

This enumeration selects the user settable output signal to configure. Once a user settable output signal has been selected, all changes to the user settable output signal settings will be applied to the selected user settable output signal.

4.49.4.159 GenApi::IBoolean& Pylon::CBaslerGigECamera::UserOutputValue

Sets the state of the selected user settable output signal.

This boolean value sets the state of the selected user settable output signal.

4.49.4.160 GenApi::IInteger& Pylon::CBaslerGigECamera::UserOutputValueAll

A single bitfield that sets the state of all user settable output signals in one access.

This integer value is a single bitfield that sets the state of all user settable output signals in one access.

4.49.4.161 GenApi::IInteger& Pylon::CBaslerGigECamera::UserOutputValueAllMask

Defines a mask that is used when the User Output Value All setting is used to set all of the user settable output signals in one access.

This integer value defines a mask that is used when the User Output Value All setting is used to set all of the user settable output signals in one access.

4.49.4.162 GenApi::IEnumerationT<Basler_GigECameraParams::UserSetDefaultSelectorEnums >& Pylon::CBaslerGigECamera::UserSetDefaultSelector

Sets the configuration set to be used as the default startup set. The configuration set that has been selected as the default startup set will be loaded as the active set whenever the camera is powered on or reset.

This enumeration sets the configuration set to be used as the default startup set. The configuration set that has been selected as the default startup set will be loaded as the active set whenever the camera is powered on or reset.

4.49.4.163 GenApi::ICommand& Pylon::CBaslerGigECamera::UserSetLoad

Loads the selected configuration into the camera's volatile memory and makes it the active configuration set. Once the selected set is loaded, the parameters in the selected set will control the camera.

This command loads the selected configuration set from the non-volatile memory in the camera to the volatile memory and makes the selected set the active configuration set. Once the selected set is loaded, the parameters in the selected set will control the camera.

4.49.4.164 GenApi::ICommand& Pylon::CBaslerGigECamera::UserSetSave

Saves the current active configuration set into the selected user set.

This command copies the parameters in the current active configuration set into the selected user set in the camera's non-volatile memory.

4.49.4.165 GenApi::IEnumerationT<Basler_GigECameraParams::UserSetSelectorEnums >& Pylon::CBaslerGigECamera::UserSetSelector

Selects the configuration set to load, save, or configure. Once a configuration set has been selected, all changes to the configuration set settings will be applied to the selected configuration set.

This enumeration selects the configuration set to load, save or configure. Possible values for the User Set Selector are: Default: Selects a configuration set that contains factory settings. User Set 1: Selects the first user set. When the Default configuration set is selected and loaded using User Set Load, the device must be in default factory settings state and must make sure the mandatory continuous acquisition use case works directly. Default User Set is read-only and cannot be modified.

4.49.4.166 GenApi::IInteger& Pylon::CBaslerGigECamera::Width

Sets the width of the area of interest in pixels.

This value sets the width of the area of interest in pixels.

4.49.4.167 GenApi::IInteger& Pylon::CBaslerGigECamera::WidthMax

Indicates the maximum allowed width of the image in pixels.

This is a read only element. It is an integer that indicates maximum allowed width of the image in pixels taking into account any function that may limit the allowed width.

4.50 Pylon::CBaslerGigEDeviceInfo Class Reference

Implementation of the GiGE specific device info object.

Inherits Pylon::CDeviceInfo.

Public Member Functions

- String_t GetAddress () const

  Retrieves the device IP address in a human-readable representation.

- String_t GetIpAddress () const

  Retrieves the device IP address.

- String_t GetDefaultGateway () const

  Retrieves the device's gateway.

- String_t GetSubnetMask () const

  Retrieves the device subnet mask in a human-readable representation.

- String_t GetMacAddress () const

  Retrieves the device MAC address in a human-readable representation.

- String_t GetInterface () const

  Retrieves the interface address in a human-readable representation.

- bool IsPersistentIpActive () const

  Returns true when the device is configured for a persitent IP address.

- bool IsDhcpActive () const

  Returns true when the device is configured for using DHCP.

- bool IsAutoIpActive () const

  Returns true when the device is configured for using Auto IP (aka LLA).

- bool IsPersistentIpSupported () const

  Returns true when the device supports configuring a persitent IP address.

- bool IsDhcpSupported () const

  Returns true when the device supports DHCP.

- bool IsAutoIpSupported () const

  Returns true when the device supports Auto IP (aka LLA).

- String_t GetSerialNumber () const

  Retrieves the serial number if it supported by the underlying implementation, or "N/A".

- String_t GetUserDefinedName () const

  Retrieves the user-defined name if present.

- String_t GetModelName () const

  Retrieves the model name of the device.

- String_t GetDeviceVersion () const

  Retrieves the version string of the device.

- String_t GetDeviceFactory () const

  Retrieves the device factory able to create this device.

- bool operator== (const CInfoBase &rhs)

  compare two info objects

- String_t GetFriendlyName () const

  Retrieves a human readable name.

- String_t GetFullName () const

  Retrieves the unique device name.

- String_t GetVendorName () const

  Retrieves the device vendor name.

- String_t GetDeviceClass () const

  Retrieves the device class.

- virtual int GetPropertyNames (StringList_t &) const

  Retrieves a list of property names.

- virtual bool GetPropertyValue (const String_t &Name, String_t &Value) const

  Retrieves a property value.

- virtual void SetPropertyValue (const String_t &Name, const String_t &Value)

  Modifies a property value.

4.50.1 Detailed Description

Implementation of the GiGE specific device info object.

Enhances the general CDeviceInfo by the attributes Address and Interface. Address is the device IP number, Interface is the IP number of the connected interface.

4.51 Pylon::CBaslerGigEEventGrabber Class Reference

Interface to the GigE event parameters.

Inherits Pylon::CEventGrabberProxy.

Public Member Functions

- virtual ~CBaslerGigEEventGrabber (void)
  Destructor.

Implementation of the IEventGrabber interface

Construction

Some smart pointer functionality

- void Attach (Pylon::IEventGrabber *pInterface)
  Attach an interface pointer.

Implementation of the IEventGrabber interface

Implementation of the IEventGrabber interface

Public Attributes

- GenApi::IInteger & NumBuffer
  Number of buffers to be used.
- GenApi::IInteger & Timeout
  Acknowledge timeout in milliseconds.
- GenApi::IInteger & RetryCount
  Maximum number of retries.

4.51.1 Detailed Description

Interface to the GigE event parameters.

4.51.2 Member Function Documentation

4.51.2.1 void Pylon::CBaslerGigEEventGrabber::Attach (Pylon::IEventGrabber * *pInterface*) [inline]

Attach an interface pointer.

It is not allowed to call Attach() when there is already an interface pointer attached!

Reimplemented from Pylon::CEventGrabberProxy.

4.51.3 Member Data Documentation

4.51.3.1 GenApi::IInteger& Pylon::CBaslerGigEEventGrabber::NumBuffer

Number of buffers to be used.

Number of Buffers that are going to be used receiving events. .

4.51.3.2 GenApi::IInteger& Pylon::CBaslerGigEEventGrabber::RetryCount

Maximum number of retries.

Number retry attempts by the camera to get an acknowledge for a sent event message.

4.51.3.3 GenApi::IInteger& Pylon::CBaslerGigEEventGrabber::Timeout

Acknowledge timeout in milliseconds.

Time to wait by the camera if an acknowledge request is configured (RetryCount != 0) to wait until the acknowledge arrives before resending the event message on its own.

4.52 Pylon::CBaslerGigEStreamGrabber Class Reference

Interface to the GigE specific stream parameters.

Inherits Pylon::CStreamGrabberProxy.

Public Member Functions

- virtual ~CBaslerGigEStreamGrabber (void)

  Destructor.

Implementation of the IStreamGrabber interface

Construction

Some smart pointer functionality

- void Attach (Pylon::IStreamGrabber *pInterface)

  Attach an interface pointer.

Implementation of the IStreamGrabber interface

Implementation of the IStreamGrabber interface

Public Attributes

- GenApi::IInteger & MaxNumBuffer

  Maximum number of buffers that are used simultaneous.

- GenApi::IInteger & MaxBufferSize

  Maximum buffer size in bytes.

- GenApi::IInteger & ReceiveWindowSize

  Size of the sliding receive window in number of frames.

- GenApi::IBoolean & EnableResend

  Enables / disables the packet resend mechanism.

- GenApi::IInteger & PacketTimeout

  Timeout period in milliseconds between two packets within one frame.

- GenApi::IInteger & FrameRetention

  Maximum time in milliseconds to receive all packets of an individual frame.

- GenApi::IInteger & ResendRequestThreshold

  Resend threshold as percentage of receive window.

- GenApi::IInteger & ResendRequestBatching

  Additional resend batching as percentage of threshold.

- GenApi::IInteger & ResendTimeout

  Time in milliseconds to wait until a resend request is issued.

- GenApi::IInteger & ResendRequestResponseTimeout

  Timeout in milliseconds for missing resend responses.

- GenApi::IInteger & MaximumNumberResendRequests

  Maximum number of resend requests per packet.

- GenApi::IInteger & Type

  Current stream grabbers type.

4.52.1 Detailed Description

Interface to the GigE specific stream parameters.

4.52.2 Member Function Documentation

4.52.2.1 void Pylon::CBaslerGigEStreamGrabber::Attach (Pylon::IStreamGrabber * *pInterface*) [inline]

Attach an interface pointer.

It is not allowed to call Attach() when there is already an interface pointer attached!

Reimplemented from Pylon::CStreamGrabberProxy.

4.52.3 Member Data Documentation

4.52.3.1 GenApi::IInteger& Pylon::CBaslerGigEStreamGrabber::Type

Current stream grabbers type.

The concrete type of the stream grabber in use.

4.53 Pylon::CBaslerGigETLParams Class Reference

Interface to the GigE transport layer specific parameters.

Inherits Pylon::CNodeMapProxy.

Public Member Functions

- virtual ~CBaslerGigETLParams (void)
  Destructor.

Construction

Some smart pointer functionality

- void Attach (GenApi::INodeMap *pInterface)
  Attach an interface pointer.

Partial implementation of the INodeMap interface

Partial implementation of the INodeMap interface

Public Attributes

- GenApi::IInteger & ReadTimeout
  Read access timeout value in milliseconds.
- GenApi::IInteger & WriteTimeout
  Write access timeout value in milliseconds.
- GenApi::IInteger & HeartbeatTimeout
  Heartbeat timeout value in milliseconds.
- GenApi::IEnumerationT< Basler_GigETLParams::RequestedTypeEnums > & RequestedType
  Requested type of GVSP stream grabber.

4.53.1 Detailed Description

Interface to the GigE transport layer specific parameters.

4.53.2 Member Function Documentation

4.53.2.1 void Pylon::CBaslerGigETLParams::Attach (GenApi::INodeMap * *pInterface*) [inline]

Attach an interface pointer.

It is not allowed to call Attach() when there is already an interface pointer attached!

Reimplemented from Pylon::CNodeMapProxy.

4.54 Pylon::CChunkParser Class Reference

Base class for chunk parsers returned by camera objects.

Inherits Pylon::IChunkParser.

Public Member Functions

- virtual bool HasCRC ()=0

  Checks if buffer has a CRC attached.

- virtual bool CheckCRC ()=0

  Checks CRC sum of buffer.

- void AttachBuffer (void *pBuffer, int64_t BufferLength, GenApi::AttachStatistics_t *pAttachStatistics=NULL)

  Pass in a buffer and let the chunk parser analyze it.

- void DetachBuffer ()

  Detaches a buffer from the chunk parser. The buffer will no longer accessed by the chunk parser.

- void UpdateBuffer (void *pBaseAddress)

  Pass in a buffer and let the chunk parser update the camera object's parameters.

Protected Member Functions

- CChunkParser (GenApi::CChunkAdapter *pParser)

  default constructor - make it protected to prevent this class from instantiation

- virtual ~CChunkParser ()

  destructor

Protected Attributes

- GenApi::CChunkAdapter * m_pParser

  Pointer to the GenApi workhorse.

4.54.1 Detailed Description

Base class for chunk parsers returned by camera objects.

Part implementation of chunk parser of common functionality.

4.54.2 Member Function Documentation

4.54.2.1 void Pylon::CChunkParser::AttachBuffer (void * *pBuffer*, int64_t *BufferLength*, GenApi::AttachStatistics_t * *pAttachStatistics* = NULL) [virtual]

Pass in a buffer and let the chunk parser analyze it.

Corresponding parameters of the camera object reflecting the chunked data will be updated.

Parameters:

pBuffer Pointer to the new buffer

BufferLength Size of the new buffer in bytes

pAttachStatistics (optional) Pointer to a record taking statistic data of the analyzed buffer

Implements Pylon::IChunkParser.

4.54.2.2 virtual bool Pylon::IChunkParser::CheckCRC () [pure virtual, inherited]

Checks CRC sum of buffer.

Parameters:

pBuffer a pointer to the new buffer

BufferLength size of the new buffer in bytes

Returns:

true if the contained CRC equals the computed value.

4.54.2.3 void Pylon::CChunkParser::DetachBuffer () [virtual]

Detaches a buffer from the chunk parser. The buffer will no longer accessed by the chunk parser.

An attached buffer must be detached before freeing it. When attaching a new buffer, the previous one gets detached automatically.

Implements Pylon::IChunkParser.

4.54.2.4 virtual bool Pylon::IChunkParser::HasCRC () [pure virtual, inherited]

Checks if buffer has a CRC attached.

Parameters:

pBuffer a pointer to the new buffer

BufferLength size of the new buffer in bytes

Returns:

true if the buffer contains CRC value.

4.54.2.5 void Pylon::CChunkParser::UpdateBuffer (void * *pBaseAddress*) [virtual]

Pass in a buffer and let the chunk parser update the camera object's parameters.

This method can be used when the layout of the chunk data hasn't changed since a previous buffer has been attached to the chunk parser. In this case UpdateBuffer is slightly faster then AttachBuffer, because the buffer's layout is remembered.

Parameters:

pBaseAddress Pointer to the new buffer

Implements Pylon::IChunkParser.

4.55 Pylon::CDeviceInfo Class Reference

A property container for device information.

Inherits Pylon::CInfoBase.

Inherited by Pylon::CBasler1394DeviceInfo, and Pylon::CBaslerGigEDeviceInfo.

Public Member Functions

- String_t GetSerialNumber () const

  Retrieves the serial number if it supported by the underlying implementation, or "N/A".

- String_t GetUserDefinedName () const

  Retrieves the user-defined name if present.

- String_t GetModelName () const

  Retrieves the model name of the device.

- String_t GetDeviceVersion () const

  Retrieves the version string of the device.

- String_t GetDeviceFactory () const

  Retrieves the device factory able to create this device.

- bool operator== (const CInfoBase &rhs)

  compare two info objects

- String_t GetFriendlyName () const

  Retrieves a human readable name.

- String_t GetFullName () const

  Retrieves the unique device name.

- String_t GetVendorName () const

  Retrieves the device vendor name.

- String_t GetDeviceClass () const

  Retrieves the device class.

- virtual int GetPropertyNames (StringList_t &) const

  Retrieves a list of property names.

- virtual bool GetPropertyValue (const String_t &Name, String_t &Value) const

  Retrieves a property value.

- virtual void SetPropertyValue (const String_t &Name, const String_t &Value)

  Modifies a property value.

4.55.1 Detailed Description

A property container for device information.

This container is created by the device enumeration process and is used to create a device.

4.56 Pylon::CEventGrabberProxy Class Reference

A Handle class (aka Envelope class) for the IEventGrabber interface.

Inherited by Pylon::CBasler1394EventGrabber, and Pylon::CBaslerGigEEventGrabber.

4.56.1 Detailed Description

A Handle class (aka Envelope class) for the IEventGrabber interface.

4.57 Pylon::CInfoBase Class Reference

Base implementation for PYLON info container.

Inherits Pylon::IInfo.

Inherited by Pylon::CDeviceInfo, and Pylon::CTlInfo.

Public Member Functions

- bool operator== (const CInfoBase &rhs)

  compare two info objects

- String_t GetFriendlyName () const

  Retrieves a human readable name.

- String_t GetFullName () const

  Retrieves the unique device name.

- String_t GetVendorName () const

  Retrieves the device vendor name.

- String_t GetDeviceClass () const

  Retrieves the device class.

- virtual int GetPropertyNames (StringList_t &) const

  Retrieves a list of property names.

- virtual bool GetPropertyValue (const String_t &Name, String_t &Value) const

  Retrieves a property value.

- virtual void SetPropertyValue (const String_t &Name, const String_t &Value)

  Modifies a property value.

4.57.1 Detailed Description

Base implementation for PYLON info container.

Info container allow a generic access to implemented properties. All Properties and their values can be accessed without knowing them in advance. It is possible to enumerate all properties available and corresponding values. Properties and values are represented as String_t. The normal usage is to have enumerators that create the info objects and clients that read only.

If the type of the info object is known before client can use specific accessor function to retrieve the property values

4.58 Pylon::CLock Class Reference

Implementation of lock protecting a critical section.

Public Member Functions

- CLock ()

  constructor

- ~CLock ()

  destructor

- bool TryLock ()

  tries to acquire the lock

- void Lock ()

  wait to acquire the lock

- void Unlock ()

  release the lock

4.58.1 Detailed Description

Implementation of lock protecting a critical section.

4.59 Pylon::CNodeMapProxy Class Reference

A Handle class (aka Envelope class) for the GenApi::INodeMap interface.

Inherited by Pylon::CBaslerGigETLParams.

4.59.1 Detailed Description

A Handle class (aka Envelope class) for the GenApi::INodeMap interface.

4.60 Pylon::CPixelFormatConverter Class Reference

Abstract base class for Color Converter classes.

Inherited by Pylon::CPixelFormatConverterBayer, Pylon::CPixelFormatConverterMonoPacked, Pylon::CPixelFormatConverterMonoXX, Pylon::CPixelFormatConverterRGB, Pylon::CPixelFormat-ConverterYUV422, and Pylon::CPixelFormatConverterYUV422YUYV.

Public Member Functions

- CPixelFormatConverter (void)

  Default constructor.

- virtual ~CPixelFormatConverter (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- bool IsInitialized () const

  Check initialisation.

- virtual void Convert (void *pDest, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const =0

  Converts input image to output image.

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImage-Format &imageFormat, unsigned int sizeRaw) const =0

  Check consitstency of input parameters.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.60.1 Detailed Description

Abstract base class for Color Converter classes.

4.60.2 Member Function Documentation

4.60.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.60.2.2 virtual void Pylon::CPixelFormatConverter::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, pure virtual]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization It throws exceptions of type const GenICam::GenericException & when input errors are detected. This base class has no implemenation. (pure virtual)

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implemented in Pylon::CPixelFormatConverterBayer, Pylon::CPixelFormatConverterMonoPacked, Pylon::CPixelFormatConverterMonoXX, and Pylon::CPixelFormatConverterRGB.

4.60.2.3 virtual void Pylon::CPixelFormatConverter::Convert (void * *pDest*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [pure virtual]

Converts input image to output image.

Parameters:

pDest Pointer to a buffer the image data will be written to, OUT Output format may be unsigned char* or SRGBAPixel* In case of RGBA, the value of A is not changed (same as input)

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input image, IN

OutputImageFormat Image format of the output image, IN

Implemented in Pylon::CPixelFormatConverterBayer, Pylon::CPixelFormatConverterMonoPacked, Pylon::CPixelFormatConverterMonoXX, Pylon::CPixelFormatConverterRGB, Pylon::CPixelFormat-ConverterYUV422, and Pylon::CPixelFormatConverterYUV422YUYV.

4.60.2.4 void Pylon::CPixelFormatConverter::Init (const struct SImageFormat & *imageFormat*) `[virtual]`

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented in Pylon::CPixelFormatConverterGamma, Pylon::CPixelFormatConverterGammaPacked, Pylon::CPixelFormatConverterMonoPacked, Pylon::CPixelFormatConverterMonoXX, Pylon::CPixel-FormatConverterTruncate, Pylon::CPixelFormatConverterTruncatePacked, Pylon::CPixelFormatConverter-YUV422, and Pylon::CPixelFormatConverterYUV422YUYV.

4.61 Pylon::CPixelFormatConverterBayer Class Reference

Class for converting Bayer images to given output formats by using truncation.

Inherits Pylon::CPixelFormatConverter.

Public Member Functions

- CPixelFormatConverterBayer (void)

  Default constructor.

- virtual ~CPixelFormatConverterBayer (void)

  Destructor, virtual.

- virtual void Convert (void *pDestRGBA, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- bool IsInitialized () const

  Check initialisation.

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImageFormat &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.61.1 Detailed Description

Class for converting Bayer images to given output formats by using truncation.

Supported input format: Bayer 8, Bayer 10, Bayer 10 packed,Bayer 12, Bayer 12 packed, Bayer 16.

Supported output format: RGBA (8-Bit per color).

Source strides (line pitch) are only supported for unpacked formats.

For packed formats source line pitch must be 0.

4.61.2 Member Function Documentation

4.61.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.61.2.2 void Pylon::CPixelFormatConverterBayer::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.61.2.3 void Pylon::CPixelFormatConverterBayer::Convert (void * *pDestRGBA*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer *pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDestRGBA Pointer to a buffer the RGBA data will be written to, OUT value of A is not changed (same as input)

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.61.2.4 void Pylon::CPixelFormatConverter::Init (const struct SImageFormat & *imageFormat*) [virtual, inherited]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented in Pylon::CPixelFormatConverterGamma, Pylon::CPixelFormatConverterGammaPacked, Pylon::CPixelFormatConverterMonoPacked, Pylon::CPixelFormatConverterMonoXX, Pylon::CPixelFormatConverterTruncate, Pylon::CPixelFormatConverterTruncatePacked, Pylon::CPixelFormatConverterYUV422, and Pylon::CPixelFormatConverterYUV422YUYV.

4.62 Pylon::CPixelFormatConverterGamma Class Reference

Class for converting Mono images to given output formats by using gamma correction.

Inherits Pylon::CPixelFormatConverterMonoXX.

Public Member Functions

- CPixelFormatConverterGamma (void)

  Default constructor.

- virtual ~CPixelFormatConverterGamma (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- double GetGamma () const

  Returns the value of the Gamma correction.

- void SetGamma (double gamma)
- virtual void Convert (void *pDest, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Protected Member Functions

- virtual void BuildLUTMono8 ()

  Build lookup table for conversion MonoXX to Mono 8.

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImageFormat &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- unsigned char * m_pLUTMono

  Lookup table needed in derived classes.

- unsigned int m_LUTMonoSize

  Size of Lookup table.

- int m_PixelSize

  Imaging PixelSize in bits as given by camera.

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.62.1 Detailed Description

Class for converting Mono images to given output formats by using gamma correction.

Supported input format: Mono 10, Mono 12, Mono 16.

Supported output format: Mono 8.

4.62.2 Member Function Documentation

4.62.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void ∗ *pDestRGB*, unsigned int *sizeDest*, const void ∗ *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.62.2.2 void Pylon::CPixelFormatConverterMonoXX::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual, inherited]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.62.2.3 void Pylon::CPixelFormatConverterMonoXX::Convert (void ∗ *pDest*, unsigned int *sizeDest*, const void ∗ *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual, inherited]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer ∗pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDest Pointer to a buffer the image data will be written to, OUT Output format is unsigned char∗

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.62.2.4 void Pylon::CPixelFormatConverterGamma::Init (const struct SImageFormat & *imageFormat*) [virtual]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented from Pylon::CPixelFormatConverterMonoXX.

4.62.2.5 void Pylon::CPixelFormatConverterGamma::SetGamma (double *gamma*)

Sets the gamma value, this is a slow function, Internal LUTS are created

Parameters:

gamma Gamma value IN

4.63 Pylon::CPixelFormatConverterGammaPacked Class Reference

Class for converting Mono packed images to given output formats by using gamma correction.

Inherits Pylon::CPixelFormatConverterMonoPacked.

Public Member Functions

- CPixelFormatConverterGammaPacked (void)

  Default constructor.

- virtual ~CPixelFormatConverterGammaPacked (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- double GetGamma () const

  Returns the value of the Gamma correction.

- void SetGamma (double gamma)
- virtual void Convert (void *pDest, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Static Public Member Functions

- static void Unpack12 (unsigned short *pDest16, const unsigned char *pSrc, int nPixels)

  Unpacks data from Mono12packed to Mono16 (short).

- static void Unpack10 (unsigned short *pDest16, const unsigned char *pSrc, int nPixels)

  Unpacks data from Mono10packed to Mono16 (short).

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImageFormat &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- Pylon::CLUTMono * m_pLUTMono

  Lookup table needed in derived classes.

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.63.1 Detailed Description

Class for converting Mono packed images to given output formats by using gamma correction.

Supported input format: Mono 10 packed, Mono 12 packed.

Supported output format: Mono 8.

4.63.2 Member Function Documentation

4.63.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.63.2.2 void Pylon::CPixelFormatConverterMonoPacked::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual, inherited]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.63.2.3 void Pylon::CPixelFormatConverterMonoPacked::Convert (void ∗ *pDest*, unsigned int *sizeDest*, const void ∗ *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const `[virtual, inherited]`

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer ∗pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDest Pointer to a buffer the image data will be written to. OUT The Output format is given by the OutputImageFormat parameter

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.63.2.4 void Pylon::CPixelFormatConverterGammaPacked::Init (const struct SImageFormat & *imageFormat*) `[virtual]`

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Implements Pylon::CPixelFormatConverterMonoPacked.

4.63.2.5 void Pylon::CPixelFormatConverterGammaPacked::SetGamma (double *gamma*)

Sets the gamma value, this is a slow function, Internal LUTS are created

Parameters:

gamma Gamma value IN

4.63.2.6 void Pylon::CPixelFormatConverterMonoPacked::Unpack10 (unsigned short * *pDest16*, const unsigned char * *pSrc*, int *nPixels*) [static, inherited]

Unpacks data from Mono10packed to Mono16 (short).

Unpacks data pSrc from Mono10packed to Mono16 (short). Output image is in Align_Mono10 Format. No linepitch for the destination buffer is supported.

Parameters:

pDest16 Pointer to destination buffer.

pSrc Pointer to source buffer.

nPixels Number of Pixel to be unpacked.

4.63.2.7 void Pylon::CPixelFormatConverterMonoPacked::Unpack12 (unsigned short * *pDest16*, const unsigned char * *pSrc*, int *nPixels*) [static, inherited]

Unpacks data from Mono12packed to Mono16 (short).

Unpacks data pSrc from Mono12packed to Mono16 (short). Output image is in Align_Mono12 Format. No linepitch for the destination buffer is supported.

Parameters:

pDest16 Pointer to destination buffer.

pSrc Pointer to source buffer.

nPixels Number of Pixel to be unpacked.

4.64 Pylon::CPixelFormatConverterMonoPacked Class Reference

Abstract base class for converter classes converting Mono packed images to given output formats.

Inherits Pylon::CPixelFormatConverter.

Inherited by Pylon::CPixelFormatConverterGammaPacked, and Pylon::CPixelFormatConverterTruncate-Packed.

Public Member Functions

- CPixelFormatConverterMonoPacked (void)

  Default constructor.

- virtual ~CPixelFormatConverterMonoPacked (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)=0

  Initializes the class with data of the input image format given by the struct SImageFormat. Image format of the input and output image (width and hight), IN.

- virtual void Convert (void *pDest, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Static Public Member Functions

- static void Unpack12 (unsigned short *pDest16, const unsigned char *pSrc, int nPixels)

  Unpacks data from Mono12packed to Mono16 (short).

- static void Unpack10 (unsigned short *pDest16, const unsigned char *pSrc, int nPixels)

  Unpacks data from Mono10packed to Mono16 (short).

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImage-Format &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- Pylon::CLUTMono * m_pLUTMono

  Lookup table needed in derived classes.

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.64.1 Detailed Description

Abstract base class for converter classes converting Mono packed images to given output formats.

4.64.2 Member Function Documentation

4.64.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.64.2.2 void Pylon::CPixelFormatConverterMonoPacked::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.64.2.3 void Pylon::CPixelFormatConverterMonoPacked::Convert (void ∗ *pDest*, unsigned int *sizeDest*, const void ∗ *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer ∗pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDest Pointer to a buffer the image data will be written to. OUT The Output format is given by the OutputImageFormat parameter

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.64.2.4 void Pylon::CPixelFormatConverterMonoPacked::Init (const struct SImageFormat & *imageFormat*) [pure virtual]

Initializes the class with data of the input image format given by the struct SImageFormat. Image format of the input and output image (width and hight), IN.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails.

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented from Pylon::CPixelFormatConverter.

Implemented in Pylon::CPixelFormatConverterGammaPacked, and Pylon::CPixelFormatConverter-TruncatePacked.

4.64.2.5 void Pylon::CPixelFormatConverterMonoPacked::Unpack10 (unsigned short ∗ *pDest16*, const unsigned char ∗ *pSrc*, int *nPixels*) [static]

Unpacks data from Mono10packed to Mono16 (short).

Unpacks data pSrc from Mono10packed to Mono16 (short). Output image is in Align_Mono10 Format. No linepitch for the destination buffer is supported.

Parameters:

pDest16 Pointer to destination buffer.

pSrc Pointer to source buffer.

nPixels Number of Pixel to be unpacked.

4.64.2.6 void Pylon::CPixelFormatConverterMonoPacked::Unpack12 (unsigned short * *pDest16*, const unsigned char * *pSrc*, int *nPixels*) [static]

Unpacks data from Mono12packed to Mono16 (short).

Unpacks data pSrc from Mono12packed to Mono16 (short). Output image is in Align_Mono12 Format. No linepitch for the destination buffer is supported.

Parameters:

pDest16 Pointer to destination buffer.

pSrc Pointer to source buffer.

nPixels Number of Pixel to be unpacked.

4.65 Pylon::CPixelFormatConverterMonoXX Class Reference

Abstract base class for converter classes converting Mono images to given output formats.

Inherits Pylon::CPixelFormatConverter.

Inherited by Pylon::CPixelFormatConverterGamma, and Pylon::CPixelFormatConverterTruncate.

Public Member Functions

- CPixelFormatConverterMonoXX (void)

  Default constructor.

- virtual ~CPixelFormatConverterMonoXX (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- virtual void Convert (void *pDest, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImageFormat &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- virtual void BuildLUTMono8 ()=0

  Build Lookup table for generating Mono 8. A derived class must implement this method.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- unsigned char * m_pLUTMono

  Lookup table needed in derived classes.

- unsigned int m_LUTMonoSize

  Size of Lookup table.

- int m_PixelSize

  Imaging PixelSize in bits as given by camera.

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.65.1 Detailed Description

Abstract base class for converter classes converting Mono images to given output formats.

4.65.2 Member Function Documentation

4.65.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.65.2.2 void Pylon::CPixelFormatConverterMonoXX::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.65.2.3 void Pylon::CPixelFormatConverterMonoXX::Convert (void ∗ *pDest*, unsigned int *sizeDest*, const void ∗ *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer ∗pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDest Pointer to a buffer the image data will be written to, OUT Output format is unsigned char∗

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.65.2.4 void Pylon::CPixelFormatConverterMonoXX::Init (const struct SImageFormat & *imageFormat*) [virtual]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented from Pylon::CPixelFormatConverter.

Reimplemented in Pylon::CPixelFormatConverterGamma, and Pylon::CPixelFormatConverterTruncate.

4.66 Pylon::CPixelFormatConverterRGB Class Reference

Class for converting RGB and BGR images to given output formats by using truncation.

Inherits Pylon::CPixelFormatConverter.

Public Member Functions

- CPixelFormatConverterRGB (void)

  Default constructor.

- virtual ~CPixelFormatConverterRGB (void)

  Destructor, virtual.

- virtual void Convert (void *pDestRGBA, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- bool IsInitialized () const

  Check initialisation.

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImageFormat &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.66.1 Detailed Description

Class for converting RGB and BGR images to given output formats by using truncation.

Supported input format: RGB(A) 8, BGR(A) 8, RGB 10, BRG 10, RGB 12, BRG 12, RGB12 packed.

Supported output format: RGBA (8-Bit per color).

Source strides (line pitch) are only supported for unpacked formats.

For packed formats source line pitch must be 0.

4.66.2 Member Function Documentation

4.66.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.66.2.2 void Pylon::CPixelFormatConverterRGB::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.66.2.3 void Pylon::CPixelFormatConverterRGB::Convert (void * *pDestRGBA*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer *pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDestRGBA Pointer to a buffer the RGBA data will be written to, OUT value of A is not changed (same as input)

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.66.2.4 void Pylon::CPixelFormatConverter::Init (const struct SImageFormat & *imageFormat*) [virtual, inherited]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented in Pylon::CPixelFormatConverterGamma, Pylon::CPixelFormatConverterGammaPacked, Pylon::CPixelFormatConverterMonoPacked, Pylon::CPixelFormatConverterMonoXX, Pylon::CPixelFormatConverterTruncate, Pylon::CPixelFormatConverterTruncatePacked, Pylon::CPixelFormatConverterYUV422, and Pylon::CPixelFormatConverterYUV422YUYV.

4.67 Pylon::CPixelFormatConverterTruncate Class Reference

Class for converting Mono images to given output formats by using truncation.

Inherits Pylon::CPixelFormatConverterMonoXX.

Public Member Functions

- CPixelFormatConverterTruncate (void)

  Default constructor.

- virtual ~CPixelFormatConverterTruncate (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- int GetHighOffsetBit () const

  Returns the value of offset correction in bit.

- int GetRangeHighOffsetBit () const

  Returns maximal value of HighOffsetBit allowed for input.

- void SetHighOffsetBit (unsigned int highOffset)

  Sets the the value of offset correction in bit.

- virtual void Convert (void *pDest, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImageFormat &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- unsigned char * m_pLUTMono

  Lookup table needed in derived classes.

- unsigned int m_LUTMonoSize

  Size of Lookup table.

- int m_PixelSize

  Imaging PixelSize in bits as given by camera.

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.67.1 Detailed Description

Class for converting Mono images to given output formats by using truncation.

Supported input format: Mono 10, Mono 12, Mono 16.

Supported output format: Mono 8.

4.67.2 Member Function Documentation

4.67.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.67.2.2 void Pylon::CPixelFormatConverterMonoXX::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual, inherited]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.67.2.3 void Pylon::CPixelFormatConverterMonoXX::Convert (void * *pDest*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual, inherited]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer *pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDest Pointer to a buffer the image data will be written to, OUT Output format is unsigned char*

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.67.2.4 void Pylon::CPixelFormatConverterTruncate::Init (const struct SImageFormat & *imageFormat*) [virtual]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented from Pylon::CPixelFormatConverterMonoXX.

4.67.2.5 void Pylon::CPixelFormatConverterTruncate::SetHighOffsetBit (unsigned int *highOffset*)

Sets the the value of offset correction in bit.

Sets the the value of offset correction in bit This is a slow function, internal LUTS are updated Value allows noise detection when highOffset= PixelSize-8 Domain of highOffset is 0 .. PixelSize, highOffset>8 amplifies the noise

Parameters:

highOffset Value of offset, IN

4.68 Pylon::CPixelFormatConverterTruncatePacked Class Reference

Class for converting Mono packed images to given output formats by using truncation.

Inherits Pylon::CPixelFormatConverterMonoPacked.

Public Member Functions

- CPixelFormatConverterTruncatePacked (void)

  Default constructor.

- virtual ~CPixelFormatConverterTruncatePacked (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- int GetHighOffsetBit () const

  Returns the value of offset correction in bit.

- int GetRangeHighOffsetBit () const

  Returns maximal value of HighOffsetBit allowed for input.

- void SetHighOffsetBit (unsigned int highOffset)

  Sets the the value of offset correction in bit.

- virtual void Convert (void *pDest, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Static Public Member Functions

- static void Unpack12 (unsigned short *pDest16, const unsigned char *pSrc, int nPixels)

  Unpacks data from Mono12packed to Mono16 (short).

- static void Unpack10 (unsigned short *pDest16, const unsigned char *pSrc, int nPixels)

  Unpacks data from Mono10packed to Mono16 (short).

Protected Member Functions

- virtual void CheckParameter_Impl (int linePitchDest, unsigned int sizeDest, const struct SImage-Format &imageFormat, unsigned int sizeRaw) const

  Check consitstency of input parameters.

- void CheckInput (const void ∗pDestRGB, unsigned int sizeDest, const void ∗pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- Pylon::CLUTMono ∗ m_pLUTMono

  Lookup table needed in derived classes.

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.68.1 Detailed Description

Class for converting Mono packed images to given output formats by using truncation.

Supported input format: Mono 10 packed, Mono 12 packed.

Supported output format: Mono 8.

4.68.2 Member Function Documentation

4.68.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void ∗ *pDestRGB*, unsigned int *sizeDest*, const void ∗ *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.68.2.2 void Pylon::CPixelFormatConverterMonoPacked::CheckParameter_Impl (int *linePitchDest*, unsigned int *sizeDest*, const struct SImageFormat & *imageFormat*, unsigned int *sizeRaw*) const [protected, virtual, inherited]

Check consitstency of input parameters.

This method checks the consitstency of input parameters by comparison with Initialization.

The method throws an exception of type const GenICam::GenericException & if input errors are detected.

Parameters:

linePitchDest Stride of destination buffer (in Bytes!!), IN

sizeDest size of destination buffer in bytes

imageFormat Image format of the input and output image, IN

sizeRaw size of src buffer in bytes

Implements Pylon::CPixelFormatConverter.

4.68.2.3 void Pylon::CPixelFormatConverterMonoPacked::Convert (void * *pDest*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual, inherited]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer *pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDest Pointer to a buffer the image data will be written to. OUT The Output format is given by the OutputImageFormat parameter

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.68.2.4 void Pylon::CPixelFormatConverterTruncatePacked::Init (const struct SImageFormat & *imageFormat*) [virtual]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Implements Pylon::CPixelFormatConverterMonoPacked.

4.68.2.5 void Pylon::CPixelFormatConverterTruncatePacked::SetHighOffsetBit (unsigned int *highOffset*)

Sets the the value of offset correction in bit.

Sets the the value of offset correction in bit This is a slow function, internal LUTS are updated Value allows noise detection when highOffset= PixelSize-8 Domain of highOffset is 0 .. PixelSize, highOffset>8 amplifies the noise

Parameters:

highOffset Value of offset, IN

4.68.2.6 void Pylon::CPixelFormatConverterMonoPacked::Unpack10 (unsigned short * *pDest16*, const unsigned char * *pSrc*, int *nPixels*) [static, inherited]

Unpacks data from Mono10packed to Mono16 (short).

Unpacks data pSrc from Mono10packed to Mono16 (short). Output image is in Align_Mono10 Format. No linepitch for the destination buffer is supported.

Parameters:

pDest16 Pointer to destination buffer.

pSrc Pointer to source buffer.

nPixels Number of Pixel to be unpacked.

4.68.2.7 void Pylon::CPixelFormatConverterMonoPacked::Unpack12 (unsigned short * *pDest16*, const unsigned char * *pSrc*, int *nPixels*) [static, inherited]

Unpacks data from Mono12packed to Mono16 (short).

Unpacks data pSrc from Mono12packed to Mono16 (short). Output image is in Align_Mono12 Format. No linepitch for the destination buffer is supported.

Parameters:

pDest16 Pointer to destination buffer.

pSrc Pointer to source buffer.

nPixels Number of Pixel to be unpacked.

4.69 Pylon::CPixelFormatConverterYUV422 Class Reference

Class for converting YUV422 UYVY images to given output formats.

Inherits Pylon::CPixelFormatConverter.

Public Member Functions

- CPixelFormatConverterYUV422 (void)

  Default constructor.

- virtual ~CPixelFormatConverterYUV422 (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- virtual void Convert (void *pDestRGBA, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Protected Member Functions

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.69.1 Detailed Description

Class for converting YUV422 UYVY images to given output formats.

Supported input format: YUV422 (UYVY) Supported output format: RGBA. (8-Bit per color)

4.69.2 Member Function Documentation

4.69.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.69.2.2 void Pylon::CPixelFormatConverterYUV422::Convert (void * *pDestRGBA*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer *pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDestRGBA Pointer to a buffer the RGBA data will be written to, OUT value of A is not changed (same as input)

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.69.2.3 void Pylon::CPixelFormatConverterYUV422::Init (const struct SImageFormat & *imageFormat*) [virtual]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails.

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented from Pylon::CPixelFormatConverter.

4.70 Pylon::CPixelFormatConverterYUV422YUYV Class Reference

Class for converting YUV422 YUYV images to given output formats.

Inherits Pylon::CPixelFormatConverter.

Public Member Functions

- CPixelFormatConverterYUV422YUYV (void)

  Default constructor.

- virtual ~CPixelFormatConverterYUV422YUYV (void)

  Destructor, virtual.

- virtual void Init (const struct SImageFormat &imageFormat)

  Initializes the class with data of the input image format given by the struct SImageFormat.

- virtual void Convert (void *pDestRGBA, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, const struct SOutputImageFormat &OutputImageFormat) const

  Converts input image to output image.

- bool IsInitialized () const

  Check initialisation.

Protected Member Functions

- void CheckInput (const void *pDestRGB, unsigned int sizeDest, const void *pSrcRaw, unsigned int sizeRaw, const struct SImageFormat &imageFormat, int linePitchDest) const

  Check consistency of input parameters.

Protected Attributes

- bool m_SwapImage

  Source and dest. images are swapped,.

- bool m_BigEndian

  Selected endian type, default is little endian.

- SImageFormat m_ImageFormat

  Image format of the input image.

4.70.1 Detailed Description

Class for converting YUV422 YUYV images to given output formats.

Supported input format: YUV422 (YUYV) Supported output format: RGBA. (8-Bit per color)

4.70.2 Member Function Documentation

4.70.2.1 void Pylon::CPixelFormatConverter::CheckInput (const void * *pDestRGB*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, int *linePitchDest*) const [protected, inherited]

Check consistency of input parameters.

This method checks the consitstency of input parameters by calling the CheckParameter_Impl method.

Parameters:

pDestRGB Pointer to the output RGB data, OUT

sizeDest size of destination buffer in bytes

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes

imageFormat Image format of the input and output image, IN

linePitchDest Stride of destination buffer (in bytes), IN

4.70.2.2 void Pylon::CPixelFormatConverterYUV422YUYV::Convert (void * *pDestRGBA*, unsigned int *sizeDest*, const void * *pSrcRaw*, unsigned int *sizeRaw*, const struct SImageFormat & *imageFormat*, const struct SOutputImageFormat & *OutputImageFormat*) const [virtual]

Converts input image to output image.

Converts input image pSrcRaw to output image pDest in given format. Struct imageFormat describes the input image format (Width, Height, etc.) according to Standard Feature Nameing conventions Output buffer *pDest must have sufficient size to hold the image. Outpt image has same Width and Height, than input image, possibly different line stride The method throws an exception of type const GenICam::GenericException & if input errors are detected. In this case the output is not changed.

Parameters:

pDestRGBA Pointer to a buffer the RGBA data will be written to, OUT value of A is not changed (same as input)

sizeDest size of destination buffer in bytes!

pSrcRaw Pointer to the pixel data to be converted, IN

sizeRaw size of src buffer in bytes!

imageFormat Image format of the input and output image, IN

OutputImageFormat Image format of the output image, IN

Implements Pylon::CPixelFormatConverter.

4.70.2.3 void Pylon::CPixelFormatConverterYUV422YUYV::Init (const struct SImageFormat & *imageFormat*) [virtual]

Initializes the class with data of the input image format given by the struct SImageFormat.

Initializes the class with struct SImageFormat for later image processing. Init must be called befor convert. Multiple inits are allowed. The method throws an exception of type const GenICam::GenericException & when input errors are detected or Init fails.

Parameters:

imageFormat Image format of the input and output image (width and hight), IN

Reimplemented from Pylon::CPixelFormatConverter.

4.71 Pylon::CPylonDeviceProxy Class Reference

A Handle class (aka Envelope class) for IPylonDevice implementations.

Inherited by Pylon::CBasler1394Camera, and Pylon::CPylonGigEDeviceProxy.

4.71.1 Detailed Description

A Handle class (aka Envelope class) for IPylonDevice implementations.

4.72 Pylon::CStreamGrabberProxy Class Reference

A Handle class (aka Envelope class) for the IStreamGrabber interface.

Inherited by Pylon::CBasler1394StreamGrabber, and Pylon::CBaslerGigEStreamGrabber.

4.72.1 Detailed Description

A Handle class (aka Envelope class) for the IStreamGrabber interface.

4.73 Pylon::CTIFactory Class Reference

the Transport Layer Factory

Inherits Pylon::IDeviceFactory.

Public Member Functions

- int EnumerateTIs (TIInfoList_t &)
  Retrieve a list of available transport layers.
- ITransportLayer * CreateTI (const CTIInfo &)
  Create a transport layer object from a TIInfo object.
- ITransportLayer * CreateTI (const String_t &DeviceClass)
  Create a transport layer object specified by the transport layer's device class identifier.
- void ReleaseTI (const ITransportLayer *)
  Destroys a transport layer object.
- virtual int EnumerateDevices (DeviceInfoList_t &, bool addToList=false)
  returns a list of available devices
- virtual IPylonDevice * CreateDevice (const CDeviceInfo &di)
  creates a device from a device info object
- virtual IPylonDevice * CreateDevice (const String_t &)
  creates a device by its unique name (i.e. fullname)
- virtual void DestroyDevice (IPylonDevice *)
  destroys a device

Static Public Member Functions

- static CTIFactory & GetInstance ()
  retrieve the transport layer factory

Friends

- void PylonTerminate (bool)
  Frees resources allocated by the pylon runtime system.

4.73.1 Detailed Description

the Transport Layer Factory

Creates, Destroys and Enumerates transport layers as well as their devices.

4.73.2 Friends And Related Function Documentation

4.73.2.1 void PylonTerminate (bool) [friend]

Frees resources allocated by the pylon runtime system.

Call this function before terminating the application. Don't use any pylon methods or pylon objects after having called PylonTerminate().

4.74 Pylon::DeviceInfoList Class Reference

STL std::vector like container for Pylon::CDeviceInfo objects.

4.74.1 Detailed Description

STL std::vector like container for Pylon::CDeviceInfo objects.

Based on the GenICam::gcstring_vector class

4.75 Pylon::EventResult Class Reference

An event result.

4.75.1 Detailed Description

An event result.

4.76 Pylon::Function_CallbackBody< Function, P > Class Template Reference

Container for a function pointer.

Inherits Pylon::Base_Callback1Body< P >.

Public Member Functions

- Function_CallbackBody (const Function &function)

  Constructor.

- void operator() (P p) const

  execute operation: call the function

- Function_CallbackBody< Function, P > * clone () const

  virtual copy constructor

4.76.1 Detailed Description

template<class Function, class P> class Pylon::Function_CallbackBody< Function, P >

Container for a function pointer.

4.77 Pylon::GrabResult Class Reference

A grab result that combines the used image buffer and status information.

Public Member Functions

- GrabResult ()

  Default constructor.

- bool Succeeded () const

  True iff status is grabbed.

- StreamBufferHandle Handle () const

  Get the buffer handle.

- void ∗ Buffer () const

  Get the pointer to the buffer.

- GrabStatus Status () const

  Get the grab status.

- const void ∗ Context () const

  Get the pointer the user provided context.

- uint32_t FrameNr () const

  Get the index of the grabbed frame.

- PayloadType GetPayloadType () const

  Get the actual payload type.

- PixelType GetPixelType () const

  Get the actual pixel type.

- uint64_t GetTimeStamp () const

  Get the camera specific tick count.

- int32_t GetSizeX () const

  Get the actual number of columns in pixel.

- int32_t GetSizeY () const

  Get the actual number of rows in pixel.

- int32_t GetOffsetX () const

  Get the actual starting column.

- int32_t GetOffsetY () const

  Get the actual starting row.

- int32_t GetPaddingX () const

Get the number of extra data at the end of each row in bytes.

- int32_t GetPaddingY () const

  Get the number of extra data at the end of the image data in bytes.

- int64_t GetPayloadSize () const

  Get the actual payload size in bytes.

- size_t GetPayloadSize_t () const

  Get the actual payload size in bytes as size_t.

- String_t GetErrorDescription () const

  Get a description of the current error.

- uint32_t GetErrorCode () const

  Get the current error code.

4.77.1 Detailed Description

A grab result that combines the used image buffer and status information.

Note that depending on the used interface technology, the specific camera and the situation some of the attributes are not meaningful, e. g. timestamp in case of an canceled grab.

4.77.2 Member Function Documentation

4.77.2.1 uint32_t Pylon::GrabResult::FrameNr () const [inline]

Get the index of the grabbed frame.

In case of GigE-Vision this sequence number starts with 1 and wraps around at 65535. The value 0 has the special meaning that this feature is not supported by the camera.

This index is specific to a single stream.

4.77.2.2 int32_t Pylon::GrabResult::GetOffsetX () const [inline]

Get the actual starting column.

This is only defined in case of image data.

4.77.2.3 int32_t Pylon::GrabResult::GetOffsetY () const [inline]

Get the actual starting row.

This is only defined in case of image data.

4.77.2.4 int32_t Pylon::GrabResult::GetPaddingX () const [inline]

Get the number of extra data at the end of each row in bytes.

This is only defined in case of image data.

4.77.2.5 int32_t Pylon::GrabResult::GetPaddingY () const [inline]

Get the number of extra data at the end of the image data in bytes.

This is only defined in case of image data.

4.77.2.6 PixelType Pylon::GrabResult::GetPixelType () const [inline]

Get the actual pixel type.

This is only defined in case of image data.

4.77.2.7 int32_t Pylon::GrabResult::GetSizeX () const [inline]

Get the actual number of columns in pixel.

This is only defined in case of image data.

4.77.2.8 int32_t Pylon::GrabResult::GetSizeY () const [inline]

Get the actual number of rows in pixel.

This is only defined in case of image data.

4.77.2.9 uint64_t Pylon::GrabResult::GetTimeStamp () const [inline]

Get the camera specific tick count.

In case of GigE-Vision this describes when the image exposure was started. Cameras that do not support this feature return zero. If supported this may be used to determine which ROIs were acquired simultaneously.

In case of FireWire this value describes the cycle time when the first packet arrives.

4.78 Pylon::IChunkParser Interface Reference

The interface for chunk parsers analyzing a data stream consisting of chunks.

Inherited by Pylon::CChunkParser.

Public Member Functions

- virtual void AttachBuffer (void *pBuffer, int64_t BufferLength, GenApi::AttachStatistics_t *pAttachStatistics=NULL)=0

  Pass in a buffer and let the chunk parser analyze it.

- virtual void DetachBuffer ()=0

  Detaches a buffer from the chunk parser. The buffer will no longer accessed by the chunk parser.

- virtual void UpdateBuffer (void *pBaseAddress)=0

  Pass in a buffer and let the chunk parser update the camera object's parameters.

- virtual bool HasCRC ()=0

  Checks if buffer has a CRC attached.

- virtual bool CheckCRC ()=0

  Checks CRC sum of buffer.

4.78.1 Detailed Description

The interface for chunk parsers analyzing a data stream consisting of chunks.

When chunk data is enabled in the camera the stream sends along with the image additional data. This data needs to be extracted with help of a chunk parser. These are created using the camera's member function IPylonDevice::CreateChunkParser().

When an image buffer containing chunked data is passed to a chunk parser, it updates the camera's chunk related parameters. After changing the data layout either by changing the AOI or enabling/disabling chunk features the parser needs to reinitialized by calling IChunkParser::AttachBuffer(). Thereafter buffers with the same layout are processed with the IChunkParser::UpdateBuffer(). Chunk related properties remain unchanged until a new buffer is passed in.

After usage return the chunk parser with a call of the camera's IPylonDevice::DestroyChunkParser() member function.

The code snippet below demonstrates the usage of a chunk parser. A complete program is found in the sample AcquireSingleFrame_ChunkImage.

Note:
Check the payload type of the GrabResult_t before attaching or updating the buffer.

```
IChunkParser &ChunkParser = *Camera.CreateChunkParser();

// enable Frame Counter chunk
Camera.ChunkModeActive.SetValue( true );
Camera.ChunkSelector.SetValue( ChunkSelector_Framecounter );
```

```
Camera.ChunkEnable.SetValue( true );

// retrieve data from the chunk
GrabResult Result;
if (StreamGrabber.RetrieveResult( Result ) &&
    Result.Succeeded() &&
    Result.PayloadType() == PayloadType_ChunkData)
{
    ChunkParser.AttachBuffer( Result.Buffer(), Result.GetPayloadSize() );
    int64_t frameCounter = Camera.ChunkFramecounter.GetValue();
}
```

4.78.2 Member Function Documentation

4.78.2.1 virtual void Pylon::IChunkParser::AttachBuffer (void * *pBuffer*, int64_t *BufferLength*, GenApi::AttachStatistics_t * *pAttachStatistics* = NULL) [pure virtual]

Pass in a buffer and let the chunk parser analyze it.

Corresponding parameters of the camera object reflecting the chunked data will be updated.

Parameters:

pBuffer Pointer to the new buffer

BufferLength Size of the new buffer in bytes

pAttachStatistics (optional) Pointer to a record taking statistic data of the analyzed buffer

Implemented in Pylon::CChunkParser.

4.78.2.2 virtual bool Pylon::IChunkParser::CheckCRC () [pure virtual]

Checks CRC sum of buffer.

Parameters:

pBuffer a pointer to the new buffer

BufferLength size of the new buffer in bytes

Returns:

true if the contained CRC equals the computed value.

4.78.2.3 virtual void Pylon::IChunkParser::DetachBuffer () [pure virtual]

Detaches a buffer from the chunk parser. The buffer will no longer accessed by the chunk parser.

An attached buffer must be detached before freeing it. When attaching a new buffer, the previous one gets detached automatically.

Implemented in Pylon::CChunkParser.

4.78.2.4 virtual bool Pylon::IChunkParser::HasCRC () [pure virtual]

Checks if buffer has a CRC attached.

Parameters:

pBuffer a pointer to the new buffer

BufferLength size of the new buffer in bytes

Returns:

true if the buffer contains CRC value.

4.78.2.5 virtual void Pylon::IChunkParser::UpdateBuffer (void * *pBaseAddress*) [pure virtual]

Pass in a buffer and let the chunk parser update the camera object's parameters.

This method can be used when the layout of the chunk data hasn't changed since a previous buffer has been attached to the chunk parser. In this case UpdateBuffer is slightly faster then AttachBuffer, because the buffer's layout is remembered.

Parameters:

pBaseAddress Pointer to the new buffer

Implemented in Pylon::CChunkParser.

4.79 Pylon::IDevice Interface Reference

The interface implemented by all device objects.

Inherited by Pylon::IPylonDevice.

Public Member Functions

- virtual void Open (AccessModeSet mode=(Exclusive|Stream|Control))=0

  Opens a device.

- virtual void Close ()=0

  Closes a device.

- virtual bool IsOpen () const =0

  Checks if a device already is opened.

- virtual AccessModeSet AccessMode (void) const =0

  Returns the access mode used to open the device.

- virtual const CDeviceInfo & GetDeviceInfo () const =0

  Returns the device info object storing information like the device's name.

4.79.1 Detailed Description

The interface implemented by all device objects.

4.79.2 Member Function Documentation

4.79.2.1 virtual void Pylon::IDevice::Close () [pure virtual]

Closes a device.

The close method closes all involved drivers and an existing connection to the device will be released. Other applications now can access the device.

4.79.2.2 virtual const CDeviceInfo& Pylon::IDevice::GetDeviceInfo () const [pure virtual]

Returns the device info object storing information like the device's name.

Returns:

A reference to the device info object used to create the device by a device factory

4.79.2.3 virtual bool Pylon::IDevice::IsOpen () const [pure virtual]

Checks if a device already is opened.

Returns:
true, when the device already has been opened by the calling application.

Note:
When a device has been opened an application A, IsOpen() will return false when called by an application B not having called the device's open method.

4.79.2.4 virtual void Pylon::IDevice::Open (AccessModeSet *mode* = `(Exclusive|Stream|Control)`) `[pure virtual]`

Opens a device.

The open method initializes all involved drivers and establishes a connection to the device.

A device may support different access modes, e.g. AccessMode::Exclusive providing an exclusive access to the device.

Parameters:
mode The desired device access mode

4.80 Pylon::IDeviceFactory Interface Reference

Interface to be implemented by device factories used to create devices.

Inherited by Pylon::CTlFactory, and Pylon::ITransportLayer.

Public Member Functions

- virtual int EnumerateDevices (DeviceInfoList_t &list, bool addToList=false)=0

  Retrieves a list of available devices.

- virtual IPylonDevice * CreateDevice (const CDeviceInfo &di)=0

  Creates a camera object from a device info object.

- virtual IPylonDevice * CreateDevice (const String_t &)=0

  Creates a device by its unique name (i.e. fullname as returned by CDeviceInfo::GetFullName).

- virtual void DestroyDevice (IPylonDevice *)=0

  Destroys a device.

4.80.1 Detailed Description

Interface to be implemented by device factories used to create devices.

Each transport layer object is a device factory. These factories must implement the IDeviceFactory interface

4.80.2 Member Function Documentation

4.80.2.1 virtual IPylonDevice* Pylon::IDeviceFactory::CreateDevice (const CDeviceInfo & *di*) [pure virtual]

Creates a camera object from a device info object.

If the device creation fails, a GenApi::GenericException will be thrown.

Parameters:

di Device Info object containing all information needed to create the device

Implemented in Pylon::CTlFactory.

4.80.2.2 virtual void Pylon::IDeviceFactory::DestroyDevice (IPylonDevice *) [pure virtual]

Destroys a device.

Note:

Never try to delete a pointer to a camera device by calling free or delete. Always use the DestroyDevice method.

Implemented in Pylon::CTlFactory.

4.80.2.3 virtual int Pylon::IDeviceFactory::EnumerateDevices (DeviceInfoList_t & *list*, bool *addToList* = `false`) `[pure virtual]`

Retrieves a list of available devices.

The list contains Pylon::CDeviceInfo objects used for the device creation.

By default, the list passed in will be cleared before the device discovery is started.

Parameters:

list The list to be filled with device info objects

addToList If true, found devices will be added to the list instead of deleting the list

Returns:

number of devices found

Implemented in Pylon::CTlFactory.

4.81 Pylon::IEventAdapter Interface Reference

IEventAdapter delivers event messages to ports.

Public Member Functions

- virtual void DeliverMessage (const uint8_t msg[], uint32_t numBytes)=0

  Deliver message.

4.81.1 Detailed Description

IEventAdapter delivers event messages to ports.

4.82 Pylon::IEventGrabber Interface Reference

Interface of an object receiving asynchronous events.

Asynchronous event messages are received from the camera. Internal Buffers are filled and stored in an output queue. While the output queue contains data the associated waitobject is signalled.

Public Member Functions

- virtual void Open ()=0

  Open the event grabber.

- virtual void Close ()=0

  Close the event grabber.

- virtual bool IsOpen () const =0

  Retrieve whether the event grabber is open.

- virtual bool RetrieveEvent (EventResult &)=0

  Retrieve an event message from the output queue.

- virtual WaitObject & GetWaitObject () const =0

  Return the event object associated with the grabber.

- virtual GenApi::INodeMap * GetNodeMap ()=0

  Return the associated event grabber parameters.

4.82.1 Detailed Description

Interface of an object receiving asynchronous events.

Asynchronous event messages are received from the camera. Internal Buffers are filled and stored in an output queue. While the output queue contains data the associated waitobject is signalled.

With RetrieveEvent() the first event message is copied into a user buffer.

4.82.2 Member Function Documentation

4.82.2.1 virtual GenApi::INodeMap* Pylon::IEventGrabber::GetNodeMap () [pure virtual]

Return the associated event grabber parameters.

If no parameters are available, NULL is returned.

4.82.2.2 virtual WaitObject& Pylon::IEventGrabber::GetWaitObject () const [pure virtual]

Return the event object associated with the grabber.

This object get signalled as soon as a event has occurred. It will be reset when the output queue is empty.

4.82.2.3 virtual bool Pylon::IEventGrabber::RetrieveEvent (EventResult &) [pure virtual]

Retrieve an event message from the output queue.

Returns:

When the event was available true is returned and the event message is copied into the EventResult.

4.83 Pylon::IInfo Interface Reference

interface for an info object

Inherits Pylon::IProperties.

Inherited by Pylon::CInfoBase.

Public Member Functions

- virtual String_t GetFriendlyName () const =0

  Retrieves a human readable name.

- virtual String_t GetFullName () const =0

  Retrieves the unique device name.

- virtual String_t GetVendorName () const =0

  Retrieves the device vendor name.

- virtual int GetPropertyNames (StringList_t &) const =0

  Retrieves a list of property names.

- virtual bool GetPropertyValue (const String_t &Name, String_t &Value) const =0

  Retrieves a property value.

- virtual void SetPropertyValue (const String_t &Name, const String_t &Value)=0

  Modifies a property value.

4.83.1 Detailed Description

interface for an info object

4.84 Pylon::IProperties Interface Reference

interface for a property container

Inherited by Pylon::IInfo.

Public Member Functions

- virtual int GetPropertyNames (StringList_t &) const =0

  Retrieves a list of property names.

- virtual bool GetPropertyValue (const String_t &Name, String_t &Value) const =0

  Retrieves a property value.

- virtual void SetPropertyValue (const String_t &Name, const String_t &Value)=0

  Modifies a property value.

4.84.1 Detailed Description

interface for a property container

4.85 Pylon::IPylonDevice Interface Reference

Interface for camera objects.

Inherits Pylon::IDevice.

Inherited by Pylon::IPylonGigEDevice.

Public Member Functions

- virtual uint32_t GetNumStreamGrabberChannels () const =0

  Returns the number of stream grabbers the camera object provides.

- virtual IStreamGrabber ∗ GetStreamGrabber (uint32_t index)=0

  Returns a pointer to a stream grabber.

- virtual IEventGrabber ∗ GetEventGrabber ()=0

  Returns a pointer to an event grabber.

- virtual GenApi::INodeMap ∗ GetNodeMap ()=0

  Returns the set of camera parameters.

- virtual GenApi::INodeMap ∗ GetTLNodeMap ()=0

  Returns the set of camera related transport layer parameters.

- virtual Pylon::IChunkParser ∗ CreateChunkParser ()=0

  Creates a chunk parser used to update those camera object members reflecting the content of additional data chunks appended to the image data.

- virtual void DestroyChunkParser (Pylon::IChunkParser ∗pChunkParser)=0

  Deletes a chunk parser.

- virtual IEventAdapter ∗ CreateEventAdapter ()=0
- virtual DeviceCallbackHandle RegisterRemovalCallback (DeviceCallback &d)=0

  Registers a surprise removal callback object.

- virtual bool DeregisterRemovalCallback (DeviceCallbackHandle h)=0

  Deregisters a surprise removal callback object.

- virtual void Open (AccessModeSet mode=(Exclusive|Stream|Control))=0

  Opens a device.

- virtual void Close ()=0

  Closes a device.

- virtual bool IsOpen () const =0

  Checks if a device already is opened.

- virtual AccessModeSet AccessMode (void) const =0

  Returns the access mode used to open the device.

- virtual const CDeviceInfo & GetDeviceInfo () const =0

  Returns the device info object storing information like the device's name.

4.85.1 Detailed Description

Interface for camera objects.

4.85.2 Member Function Documentation

4.85.2.1 virtual void Pylon::IDevice::Close () [pure virtual, inherited]

Closes a device.

The close method closes all involved drivers and an existing connection to the device will be released. Other applications now can access the device.

4.85.2.2 virtual Pylon::IChunkParser* Pylon::IPylonDevice::CreateChunkParser () [pure virtual]

Creates a chunk parser used to update those camera object members reflecting the content of additional data chunks appended to the image data.

Returns:
Pointer to the created chunk parser

Note:
Don't try to delate a chunk parser pointer by calling free or delete. Instead, use the DestroyChunkParser() method

4.85.2.3 virtual IEventAdapter* Pylon::IPylonDevice::CreateEventAdapter () [pure virtual]

Creates a Event adapter

4.85.2.4 virtual bool Pylon::IPylonDevice::DeregisterRemovalCallback (DeviceCallbackHandle *h*) [pure virtual]

Deregisters a surprise removal callback object.

Parameters:
h Handle of the callback to be removed

4.85.2.5 virtual void Pylon::IPylonDevice::DestroyChunkParser (Pylon::IChunkParser * *pChunkParser*) [pure virtual]

Deletes a chunk parser.

Parameters:
Pointer to the chunk parser to be deleted

4.85.2.6 virtual const CDeviceInfo& Pylon::IDevice::GetDeviceInfo () const [pure virtual, inherited]

Returns the device info object storing information like the device's name.

Returns:
A reference to the device info object used to create the device by a device factory

4.85.2.7 virtual IEventGrabber* Pylon::IPylonDevice::GetEventGrabber () [pure virtual]

Returns a pointer to an event grabber.

Event grabbers are used to handle events sent from a camera device.

Note:
Currently the implementation of camera objects don't support event grabbers, so NULL will be returned.

4.85.2.8 virtual GenApi::INodeMap* Pylon::IPylonDevice::GetNodeMap () [pure virtual]

Returns the set of camera parameters.

Returns:
Pointer to the GenApi node map holding the parameters

4.85.2.9 virtual IStreamGrabber* Pylon::IPylonDevice::GetStreamGrabber (uint32_t *index*) [pure virtual]

Returns a pointer to a stream grabber.

Stream grabbers (IStreamGrabber) are the objects used to grab images from a camera device. A camera device might be able to send image data over more than one logical channel called stream. A stream grabber grabs data from one single stream.

A camera objects always has at least one stream grabber having the index 0

Parameters:
index The number of the grabber to return

Returns:
A pointer to a stream grabber, NULL if index is out of range

4.85.2.10 virtual GenApi::INodeMap* Pylon::IPylonDevice::GetTLNodeMap () [pure virtual]

Returns the set of camera related transport layer parameters.

Returns:
Pointer to the GenApi node holding the transport layer parameter. If there are no transport layer parameters for the device, NULL is returned.

4.85.2.11 virtual bool Pylon::IDevice::IsOpen () const [pure virtual, inherited]

Checks if a device already is opened.

Returns:
true, when the device already has been opened by the calling application.

Note:
When a device has been opened an application A, IsOpen() will return false when called by an application B not having called the device's open method.

4.85.2.12 virtual void Pylon::IDevice::Open (AccessModeSet *mode* = (Exclusive|Stream|Control)) [pure virtual, inherited]

Opens a device.

The open method initializes all involved drivers and establishes a connection to the device.

A device may support different access modes, e.g. AccessMode::Exclusive providing an exclusive access to the device.

Parameters:
mode The desired device access mode

4.85.2.13 virtual DeviceCallbackHandle Pylon::IPylonDevice::RegisterRemovalCallback (DeviceCallback & *d*) [pure virtual]

Registers a surprise removal callback object.

Parameters:
d reference to a device callback object

Returns:
A handle which must be used to deregister a callback It is recommended to use one of the RegisterRemovalCallback() helper functions to register a callback.

Example how to register a C function

```
void MyRemovalCallback( Pylon::IPylonDevice* pDevice)
{
    // handle removal
}

DeviceCallbackHandle h =
    Pylon::RegisterRemovalCallback( m_pCamera, &MyRemovalCallback);
```

Example how to register a class member function

```
class C
{
    void MyRemovalCallback( Pylon::IPylonDevice* pDevice )
```

```
    {
      // handle removal
    }
} c;

DeviceCallbackHandle h =
    Pylon::RegisterRemovalCallback( m_pCamera, &c, &C::MyRemovalCallback);
```

4.86 Pylon::IStreamGrabber Interface Reference

Interface to an (input) data stream.

Public Member Functions

- virtual void Open ()=0

  Opens the stream grabber.

- virtual void Close ()=0

  Closes the stream grabber.

- virtual bool IsOpen () const =0

  Retrieve whether the stream grabber is open.

- virtual StreamBufferHandle RegisterBuffer (void ∗Buffer, size_t BufferSize)=0

  Registers a buffer for subsequent use.

- virtual const void ∗ DeregisterBuffer (StreamBufferHandle)=0

  Deregisters the buffer.

- virtual void PrepareGrab ()=0

  Prepares grabbing.

- virtual void FinishGrab ()=0

  Stops grabbing.

- virtual void QueueBuffer (StreamBufferHandle, const void ∗Context=NULL)=0

  Enqueues a buffer in the input queue.

- virtual void CancelGrab ()=0

  Cancels pending requests.

- virtual bool RetrieveResult (GrabResult &)=0

  Retrieves a grab result from the output queue.

- virtual WaitObject & GetWaitObject () const =0

  Returns the result event object.

- virtual GenApi::INodeMap ∗ GetNodeMap ()=0

  Returns the associated stream grabber parameter.

4.86.1 Detailed Description

Interface to an (input) data stream.

Data is filled into user provided buffers. Before usage the buffers must be registered. After calling PrepareGrab() enter them into a queue to become filled.

The streamgrabber provides a output queue which contains filled data buffers. While the queue is not empty the associated wait object is signalled. Get the buffer using the RetrieveResult() method. When FinishGrab() is called the result queue remains open, so the data may be collected. It remains open until Close() resp. the device is closed.

Parameter of streamgrabber are controlled by the elements of a own NodeMap, which can be retrieved by calling GetNodeMap().

See also:
For an introduction into the usage of stream grabbers see GrabbingImages in the Programmer's Guide.

4.86.2 Member Function Documentation

4.86.2.1 virtual void Pylon::IStreamGrabber::CancelGrab () [pure virtual]

Cancels pending requests.

, resources remain allocated. Following, the results must be retrieved from the Output Queue.

4.86.2.2 virtual void Pylon::IStreamGrabber::Close () [pure virtual]

Closes the stream grabber.

Flushes the result queue and stops the thread.

4.86.2.3 virtual const void* Pylon::IStreamGrabber::DeregisterBuffer (StreamBufferHandle) [pure virtual]

Deregisters the buffer.

Deregistering fails while the buffer is in use, so retrieve the buffer from the output queue after grabbing.

Note:
Do not delete buffers before they are deregistered.

4.86.2.4 virtual void Pylon::IStreamGrabber::FinishGrab () [pure virtual]

Stops grabbing.

Releases the resources and camera. Pending grab requests are canceled.

4.86.2.5 virtual GenApi::INodeMap* Pylon::IStreamGrabber::GetNodeMap () [pure virtual]

Returns the associated stream grabber parameter.

If no parameter are available NULL is returned.

4.86.2.6 virtual WaitObject& Pylon::IStreamGrabber::GetWaitObject () const [pure virtual]

Returns the result event object.

This object associated with result queue. The event is signaled, when queue is non-empty

4.86.2.7 virtual void Pylon::IStreamGrabber::PrepareGrab () [pure virtual]

Prepares grabbing.

Allocates resources, synchronizes with the camera and locks critical parameter

4.86.2.8 virtual void Pylon::IStreamGrabber::QueueBuffer (StreamBufferHandle, const void * *Context* = NULL) [pure virtual]

Enqueues a buffer in the input queue.

PrepareGrab is required to queue buffers. The context is returned together with the buffer and the grab result. It is not touched by the stream grabber. It is illegal to queue a buffer a second time before it is fetched from the result queue.

4.86.2.9 virtual StreamBufferHandle Pylon::IStreamGrabber::RegisterBuffer (void * *Buffer*, size_t *BufferSize*) [pure virtual]

Registers a buffer for subsequent use.

Stream must be locked to register buffers The Buffer size may not exceed the value specified when PrepareGrab was called.

4.86.2.10 virtual bool Pylon::IStreamGrabber::RetrieveResult (GrabResult &) [pure virtual]

Retrieves a grab result from the output queue.

Returns:

When result was available true is returned and and the first result is copied into the grabresult. Otherwise the grabresult remains unchanged and false is returned.

4.87 Pylon::LockableObject< Object > Class Template Reference

Base class the makes the Object lockable.

Public Member Functions

- Lock GetLock () const

  Get a new lock.

Public Attributes

- CLock m_Lock

  the Lock

Classes

- class Lock

  A scopelevel Lock class.

4.87.1 Detailed Description

template<class Object> class Pylon::LockableObject< Object >

Base class the makes the Object lockable.

4.88 Pylon::LockableObject< Object >::Lock Class Reference

A scopelevel Lock class.

4.88.1 Detailed Description

template<class Object> class Pylon::LockableObject< Object >::Lock

A scopelevel Lock class.

Automatically aquires the lock when created and releases it when destroyed.

4.89 Pylon::Member_CallbackBody< Client, Member, P > Class Template Reference

Container for a member function pointer.

Inherits Pylon::Base_Callback1Body< P >.

Public Types

- typedef void(Client::*) PMEMBERFUNC (P)

  Member function type.

Public Member Functions

- Member_CallbackBody (Client &client, Member member)

  Constructor.

- void operator() (P pP) const

  execute operation

- Member_CallbackBody< Client, Member, P > * clone () const

  virtual copy constructor

4.89.1 Detailed Description

template<class Client, class Member, class P> class Pylon::Member_CallbackBody< Client, Member, P >

Container for a member function pointer.

4.90 Pylon::PylonAutoInitTerm Class Reference

helperclass to automagically call PylonInitialize and PylonTerminate in constructor and destructor

4.90.1 Detailed Description

helperclass to automagically call PylonInitialize and PylonTerminate in constructor and destructor

```
#include <pylon/TlFactory.h>

// ...

using namespace Pylon;

int main(int argc, char* argv[])
{

    PylonAutoInitTerm autoInitTerm;

    // Get the transport layer factory
    CTlFactory& TlFactory = CTlFactory::GetInstance();

    // Get all attached cameras and exit application if no camera is found
    DeviceInfoList_t devices;
    if ( 0 == TlFactory.EnumerateDevices( devices ) )
    {
        cerr << "No camera present!" << endl;
        return 1;
    }
    else
    {
        // do something with devices ...
        //...
    }


    return 0;

}  // When leaving the main function, the desctructor of the PylonAutoInitTerm object will be ca
   // and it will in turn call PylonTerminate.
```

4.91 Pylon::SystemWaitObject< NativeObject > Class Template Reference

Public Member Functions

- bool IsSignalled () const

  retrieve the current status

- operator NativeObject () const

  conversion operator

- bool Wait (unsigned timeout)

  wait for the object to be signalled

4.91.1 Detailed Description

template<class NativeObject> class Pylon::SystemWaitObject< NativeObject >

A wait-only object, typically provided by the system

4.92 Pylon::TlInfoList Class Reference

STL std::vector like container for Pylon::CTlInfo objects.

4.92.1 Detailed Description

STL std::vector like container for Pylon::CTlInfo objects.

Based on the GenICam::gcstring_vector class

4.93 Pylon::WaitObject Class Reference

A wait object.

Inherits Pylon::SystemWaitObject< HANDLE >.

Inherited by Pylon::AlertableWaitObject.

Public Member Functions

- WaitObject (HANDLE h, bool duplicate=true)

  constructor using an existing object

- bool IsSignalled () const

  check whether the object is signalled.

- operator HANDLE () const

  conversion operator

- bool Wait (unsigned timeout)

  wait for the object to be signalled

4.93.1 Detailed Description

A wait object.

4.94 Pylon::WaitObjects Class Reference

A set of wait objects.

Inherits Pylon::WaitObjectSet< HANDLE >.

Public Types

- typedef SystemWaitObject< HANDLE > value_type

  type of the elements

Public Member Functions

- WaitObjects ()

  default constructor

- ~WaitObjects ()

  destructor

- WaitObjects (const WaitObjects &)

  copy constructor

- WaitObjects & operator= (const WaitObjects &)

  assignment operator

- bool WaitForAll (unsigned timeout)

  Wait for all objectes to get signalled.

- bool WaitForAny (unsigned timeout, unsigned *pIndex=NULL)

  Wait for any one object to get signalled.

- void Add (value_type &o)

  add an object to wait on

- void Add (const value_type &)

  add a waitobject

4.94.1 Detailed Description

A set of wait objects.

4.94.2 Member Function Documentation

4.94.2.1 bool Pylon::WaitObjects::WaitForAll (unsigned *timeout*)

Wait for all objectes to get signalled.

Parameters:

timeout maximum wait period in milliseconds

Returns:

true if all objects were signalled

Reimplemented from Pylon::WaitObjectSet< HANDLE >.

4.94.2.2 bool Pylon::WaitObjects::WaitForAny (unsigned *timeout*, unsigned * *pIndex* = NULL)

Wait for any one object to get signalled.

Parameters:

timeout maximum wait period in milliseconds

****pIndex*** (optional) pointer to buffer taking the index of the signalled object

Returns:

true if any object was signalled.

Reimplemented from Pylon::WaitObjectSet< HANDLE >.

4.95 Pylon::WaitObjectSet< NativeObject > Class Template Reference

Public Types

- typedef SystemWaitObject< NativeObject > value_type

  type of the elements

Public Member Functions

- void Add (const value_type &)

  add a waitobject

- bool WaitForAny (unsigned timeout, unsigned *pIndex=NULL)

  wait for one of the objects to be signalled

- bool WaitForAll (unsigned timeout)

  wait for all objects to be signalled

4.95.1 Detailed Description

template<class NativeObject> class Pylon::WaitObjectSet< NativeObject >

A collection of waitobjects

Revision History

Doc. ID Number	Date	Changes
AW00013101000	13 Sep 2006	Initial release. This release is a preliminary version of the document.
AW00013102000	8 Jan 2007	Updated the document to reflect the new events capability that was added to the GigE cameras.
AW00013103000	6 Jun 2007	Updated the document to reflect the change from pylon 0.9 to pylon 1.0. Added Section 1.5 which contains software licensing information.
AW00013104000	14 Feb 2008	Updated the document to reflect the change from pylon 1.0 to pylon 2.0.
AW00013105000	18 Feb 2008	Added information that was mistakenly omitted from the previous release.

Feedback

Your feedback will help us improve our documentation. Please click the link below to access an online feedback form. Your input is greatly appreciated.

http://www.baslerweb.com/umfrage/survey.html

Index

Abstract interface classes of the GenApi module, 54
AccessMode
    Pylon, 74
AcquisitionAbort
    Pylon::CBaslerGigECamera, 171
AcquisitionFrameRateAbs
    Pylon::CBasler1394Camera, 138
    Pylon::CBaslerGigECamera, 171
AcquisitionFrameRateEnable
    Pylon::CBasler1394Camera, 138
    Pylon::CBaslerGigECamera, 172
AcquisitionLineRateAbs
    Pylon::CBasler1394Camera, 138
    Pylon::CBaslerGigECamera, 172
AcquisitionMode
    Pylon::CBasler1394Camera, 138
    Pylon::CBaslerGigECamera, 172
AcquisitionStart
    Pylon::CBasler1394Camera, 138
    Pylon::CBaslerGigECamera, 172
AcquisitionStop
    Pylon::CBasler1394Camera, 138
    Pylon::CBaslerGigECamera, 172
Attach
    Pylon::CBasler1394Camera, 138
    Pylon::CBasler1394EventGrabber, 155
    Pylon::CBasler1394StreamGrabber, 156
    Pylon::CBaslerGigECamera, 171
    Pylon::CBaslerGigEEventGrabber, 197
    Pylon::CBaslerGigEStreamGrabber, 200
    Pylon::CBaslerGigETLParams, 201
AttachBuffer
    GenApi::CChunkAdapter, 81
    Pylon::CChunkParser, 203
    Pylon::IChunkParser, 259
AttachStatistics_t
    GenApi_PublicUtilities, 59

BalanceRatioAbs
    Pylon::CBasler1394Camera, 139
    Pylon::CBaslerGigECamera, 172
BalanceRatioRaw
    Pylon::CBasler1394Camera, 139
    Pylon::CBaslerGigECamera, 172
BalanceRatioSelector
    Pylon::CBasler1394Camera, 139
    Pylon::CBaslerGigECamera, 172
Base_PublicUtilities
    DECLARE_EXCEPTION, 65
    operator<<, 65
    operator>>, 65
BinningHorizontal
    Pylon::CBasler1394Camera, 139
    Pylon::CBaslerGigECamera, 173
BinningVertical
    Pylon::CBasler1394Camera, 139
    Pylon::CBaslerGigECamera, 173
BitPerPixel
    Pylon, 75
BlackLevelAbs
    Pylon::CBaslerGigECamera, 173
BlackLevelRaw
    Pylon::CBasler1394Camera, 139
    Pylon::CBaslerGigECamera, 173
BlackLevelSelector
    Pylon::CBasler1394Camera, 139
    Pylon::CBaslerGigECamera, 173

Canceled
    Pylon, 75
CancelGrab
    Pylon::IStreamGrabber, 276
CBaseRef
    GenApi_PublicImpl, 61
CBasler1394Camera
    Pylon::CBasler1394Camera, 137
CBaslerGigECamera
    Pylon::CBaslerGigECamera, 171
CBooleanRef
    GenApi_PublicImpl, 61
CCategoryRef
    GenApi_PublicImpl, 61
CCommandRef
    GenApi_PublicImpl, 61
CEnumEntryRef
    GenApi_PublicImpl, 61
CFloatRef
    GenApi_PublicImpl, 61
CheckBufferLayout
    GenApi::CChunkAdapter, 81

CheckCRC
Pylon::CChunkParser, 203
Pylon::IChunkParser, 259
CheckInput
Pylon::CPixelFormatConverter, 212
Pylon::CPixelFormatConverterBayer, 215
Pylon::CPixelFormatConverterGamma, 218
Pylon::CPixelFormatConverterGammaPacked, 221
Pylon::CPixelFormatConverterMonoPacked, 225
Pylon::CPixelFormatConverterMonoXX, 229
Pylon::CPixelFormatConverterRGB, 232
Pylon::CPixelFormatConverterTruncate, 235
Pylon::CPixelFormatConverterTruncate-Packed, 239
Pylon::CPixelFormatConverterYUV422, 243
Pylon::CPixelFormatConverterYUV422YUYV, 246
CheckParameter_Impl
Pylon::CPixelFormatConverter, 212
Pylon::CPixelFormatConverterBayer, 215
Pylon::CPixelFormatConverterGamma, 218
Pylon::CPixelFormatConverterGammaPacked, 221
Pylon::CPixelFormatConverterMonoPacked, 225
Pylon::CPixelFormatConverterMonoXX, 229
Pylon::CPixelFormatConverterRGB, 232
Pylon::CPixelFormatConverterTruncate, 235
Pylon::CPixelFormatConverterTruncate-Packed, 239
ChunkDynamicRangeMax
Pylon::CBaslerGigECamera, 173
ChunkDynamicRangeMin
Pylon::CBaslerGigECamera, 173
ChunkEnable
Pylon::CBasler1394Camera, 140
Pylon::CBaslerGigECamera, 174
ChunkFramecounter
Pylon::CBasler1394Camera, 140
Pylon::CBaslerGigECamera, 174
ChunkHeight
Pylon::CBaslerGigECamera, 174
ChunkLineStatusAll
Pylon::CBasler1394Camera, 140
Pylon::CBaslerGigECamera, 174
ChunkModeActive
Pylon::CBasler1394Camera, 140
Pylon::CBaslerGigECamera, 174
ChunkOffsetX
Pylon::CBaslerGigECamera, 174
ChunkOffsetY
Pylon::CBaslerGigECamera, 174
ChunkPayloadCRC16
Pylon::CBaslerGigECamera, 174
ChunkPixelFormat
Pylon::CBaslerGigECamera, 175
ChunkSelector
Pylon::CBasler1394Camera, 140
Pylon::CBaslerGigECamera, 175
ChunkStride
Pylon::CBaslerGigECamera, 175
ChunkTimestamp
Pylon::CBasler1394Camera, 140
Pylon::CBaslerGigECamera, 175
ChunkTriggerinputcounter
Pylon::CBasler1394Camera, 140
Pylon::CBaslerGigECamera, 175
ChunkWidth
Pylon::CBaslerGigECamera, 175
CIntegerRef
GenApi_PublicImpl, 61
Close
Pylon::IDevice, 261
Pylon::IPylonDevice, 271
Pylon::IStreamGrabber, 276
Control
Pylon, 74
Convert
Pylon::CPixelFormatConverter, 212
Pylon::CPixelFormatConverterBayer, 215
Pylon::CPixelFormatConverterGamma, 219
Pylon::CPixelFormatConverterGammaPacked, 222
Pylon::CPixelFormatConverterMonoPacked, 226
Pylon::CPixelFormatConverterMonoXX, 230
Pylon::CPixelFormatConverterRGB, 232
Pylon::CPixelFormatConverterTruncate, 236
Pylon::CPixelFormatConverterTruncate-Packed, 240
Pylon::CPixelFormatConverterYUV422, 243
Pylon::CPixelFormatConverterYUV422YUYV, 246
ConvertToRGB
Pylon, 76
ConvertToRGBA
Pylon, 76
CPortRef
GenApi_PublicImpl, 61
CreateChunkParser
Pylon::IPylonDevice, 271
CreateDevice
Pylon::IDeviceFactory, 263
CreateEventAdapter
Pylon::IPylonDevice, 271
CRegisterRef

GenApi_PublicImpl, 61
CSelectorRef
GenApi_PublicImpl, 62
CStringRef
GenApi_PublicImpl, 62
CValueRef
GenApi_PublicImpl, 62

DECLARE_EXCEPTION
Base_PublicUtilities, 65
DeregisterBuffer
Pylon::IStreamGrabber, 276
DeregisterRemovalCallback
Pylon::IPylonDevice, 271
DestroyChunkParser
Pylon::IPylonDevice, 271
DestroyDevice
Pylon::IDeviceFactory, 263
DetachBuffer
Pylon::CChunkParser, 203
Pylon::IChunkParser, 259
DeviceFirmwareVersion
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 175
DeviceID
Pylon::CBaslerGigECamera, 175
DeviceManufacturerInfo
Pylon::CBaslerGigECamera, 176
DeviceModelName
Pylon::CBaslerGigECamera, 176
DeviceReset
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 176
DeviceScanType
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 176
DeviceSerialNumber
Pylon::CBasler1394Camera, 141
DeviceUserID
Pylon::CBaslerGigECamera, 176
DeviceVendorName
Pylon::CBaslerGigECamera, 176
DeviceVersion
Pylon::CBaslerGigECamera, 176
DigitalShift
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 176

EnumerateDevices
Pylon::IDeviceFactory, 263
Event
Pylon, 74
EventNotification
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 177
EventOverrunEventFrameID
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 177
EventOverrunEventStreamChannelIndex
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 177
EventOverrunEventTimestamp
Pylon::CBasler1394Camera, 141
Pylon::CBaslerGigECamera, 177
EventSelector
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 177
Exclusive
Pylon, 74
ExposureEndEventFrameID
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 177
ExposureEndEventStreamChannelIndex
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 177
ExposureEndEventTimestamp
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 177
ExposureMode
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 178
ExposureTimeAbs
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 178
ExposureTimeBaseAbs
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 178
ExposureTimeBaseAbsEnable
Pylon::CBasler1394Camera, 142
Pylon::CBaslerGigECamera, 178
ExposureTimeRaw
Pylon::CBasler1394Camera, 143
Pylon::CBaslerGigECamera, 178

Failed
Pylon, 75
FinishGrab
Pylon::IStreamGrabber, 276
FrameNr
Pylon::GrabResult, 256

GainAbs
Pylon::CBaslerGigECamera, 178
GainRaw
Pylon::CBasler1394Camera, 143
Pylon::CBaslerGigECamera, 178
GainSelector
Pylon::CBasler1394Camera, 143

Pylon::CBaslerGigECamera, 179
Gamma
Pylon::CBasler1394Camera, 143
Pylon::CBaslerGigECamera, 179
GammaEnable
Pylon::CBasler1394Camera, 143
Pylon::CBaslerGigECamera, 179
GenApi::AttachStatistics_t, 79
GenApi::CChunkAdapter, 80
GenApi::CChunkAdapter
AttachBuffer, 81
CheckBufferLayout, 81
GenApi::CChunkAdapterDcam, 82
GenApi::CChunkAdapterGEV, 84
GenApi::CNodeMapRef, 85
GenApi::CPointer, 87
GenApi::Function_NodeCallbackBody, 89
GenApi::GVCP_CHUNK_TRAILER, 90
GenApi::IBase, 91
GenApi::IBoolean, 92
GenApi::ICategory, 93
GenApi::ICommand, 94
GenApi::IEnumEntry, 95
GenApi::IEnumeration, 96
GenApi::IEnumerationT, 99
GenApi::IEnumerationT
operator=, 100
GenApi::IFloat, 101
GenApi::IInteger, 102
GenApi::INode, 103
GenApi::INode
GetProperty, 104
GenApi::INodeMap, 105
GenApi::INodeMap
GetDeviceName, 105
GenApi::IPort, 106
GenApi::IRegister, 107
GenApi::ISelector, 108
GenApi::IString, 109
GenApi::IValue, 110
GenApi::Member_NodeCallbackBody, 111
GenApi::NodeCallback, 112
GenApi::NodeCallbackBase, 113
GenApi_PublicImpl
CBaseRef, 61
CBooleanRef, 61
CCategoryRef, 61
CCommandRef, 61
CEnumEntryRef, 61
CFloatRef, 61
CIntegerRef, 61
CPortRef, 61
CRegisterRef, 61
CSelectorRef, 62
CStringRef, 62
CValueRef, 62
GenApi_PublicInterface
IsVisible, 57
GenApi_PublicUtilities
AttachStatistics_t, 59
GenICam::AccessException, 114
GenICam::DynamicCastException, 115
GenICam::ExceptionReporter, 116
GenICam::gcstring, 117
GenICam::GenericException, 118
GenICam::InvalidArgumentException, 119
GenICam::LogicalErrorException, 120
GenICam::OutOfRangeException, 121
GenICam::PropertyException, 122
GenICam::RuntimeException, 123
GenICam::TimeoutException, 124
GetDeviceInfo
Pylon::IDevice, 261
Pylon::IPylonDevice, 271
GetDeviceName
GenApi::INodeMap, 105
GetEventGrabber
Pylon::IPylonDevice, 272
GetNodeMap
Pylon::IEventGrabber, 266
Pylon::IPylonDevice, 272
Pylon::IStreamGrabber, 276
GetOffsetX
Pylon::GrabResult, 256
GetOffsetY
Pylon::GrabResult, 256
GetPaddingX
Pylon::GrabResult, 256
GetPaddingY
Pylon::GrabResult, 257
GetPixelColorFilter
Pylon, 75
GetPixelType
Pylon::GrabResult, 257
GetProperty
GenApi::INode, 104
GetSizeX
Pylon::GrabResult, 257
GetSizeY
Pylon::GrabResult, 257
GetStreamGrabber
Pylon::IPylonDevice, 272
GetTimeStamp
Pylon::GrabResult, 257
GetTLNodeMap
Pylon::IPylonDevice, 272
GetWaitObject
Pylon::IEventGrabber, 266

Pylon::IStreamGrabber, 276
GevCCP
Pylon::CBaslerGigECamera, 179
GevCurrentDefaultGateway
Pylon::CBaslerGigECamera, 179
GevCurrentIPAddress
Pylon::CBaslerGigECamera, 179
GevCurrentIPConfiguration
Pylon::CBaslerGigECamera, 179
GevCurrentSubnetMask
Pylon::CBaslerGigECamera, 179
GevDeviceModeCharacterSet
Pylon::CBaslerGigECamera, 180
GevDeviceModeIsBigEndian
Pylon::CBaslerGigECamera, 180
GevFirstURL
Pylon::CBaslerGigECamera, 180
GevHeartbeatTimeout
Pylon::CBaslerGigECamera, 180
GevInterfaceSelector
Pylon::CBaslerGigECamera, 180
GevLinkSpeed
Pylon::CBaslerGigECamera, 180
GevMACAddress
Pylon::CBaslerGigECamera, 180
GevMessageChannelCount
Pylon::CBaslerGigECamera, 180
GevNumberOfInterfaces
Pylon::CBaslerGigECamera, 181
GevPersistentDefaultGateway
Pylon::CBaslerGigECamera, 181
GevPersistentIPAddress
Pylon::CBaslerGigECamera, 181
GevPersistentSubnetMask
Pylon::CBaslerGigECamera, 181
GevSCBWA
Pylon::CBaslerGigECamera, 181
GevSCBWR
Pylon::CBaslerGigECamera, 181
GevSCBWRA
Pylon::CBaslerGigECamera, 182
GevSCDA
Pylon::CBaslerGigECamera, 182
GevSCDCT
Pylon::CBaslerGigECamera, 182
GevSCDMT
Pylon::CBaslerGigECamera, 182
GevSCFJM
Pylon::CBaslerGigECamera, 182
GevSCFTD
Pylon::CBaslerGigECamera, 182
GevSCPD
Pylon::CBaslerGigECamera, 182
GevSCPHostPort
Pylon::CBaslerGigECamera, 183
GevSCPInterfaceIndex
Pylon::CBaslerGigECamera, 183
GevSCPSPacketSize
Pylon::CBaslerGigECamera, 183
GevSecondURL
Pylon::CBaslerGigECamera, 183
GevStreamChannelCount
Pylon::CBaslerGigECamera, 183
GevStreamChannelSelector
Pylon::CBaslerGigECamera, 183
GevSupportedIPConfigurationDHCP
Pylon::CBaslerGigECamera, 183
GevSupportedIPConfigurationLLA
Pylon::CBaslerGigECamera, 184
GevSupportedIPConfigurationPersistentIP
Pylon::CBaslerGigECamera, 184
GevSupportedOptionalCommandsConcatenation
Pylon::CBaslerGigECamera, 184
GevSupportedOptionalCommandsEVENT
Pylon::CBaslerGigECamera, 184
GevSupportedOptionalCommandsEVENTDATA
Pylon::CBaslerGigECamera, 184
GevSupportedOptionalCommandsPACKETRESEND
Pylon::CBaslerGigECamera, 184
GevSupportedOptionalCommandsWRITEMEM
Pylon::CBaslerGigECamera, 184
GevTimestampControlLatch
Pylon::CBaslerGigECamera, 185
GevTimestampControlLatchReset
Pylon::CBaslerGigECamera, 185
GevTimestampControlReset
Pylon::CBaslerGigECamera, 185
GevTimestampTickFrequency
Pylon::CBaslerGigECamera, 185
GevTimestampValue
Pylon::CBaslerGigECamera, 185
GevVersionMajor
Pylon::CBaslerGigECamera, 185
GevVersionMinor
Pylon::CBaslerGigECamera, 185
Grabbed
Pylon, 75
GrabStatus
Pylon, 74

HasCRC
Pylon::CChunkParser, 203
Pylon::IChunkParser, 259
Height
Pylon::CBasler1394Camera, 143
Pylon::CBaslerGigECamera, 185
HeightMax
Pylon::CBasler1394Camera, 143

Pylon::CBaslerGigECamera, 186

Idle
Pylon, 75
Implementation parts of the GenApi module which must be public, 60
Implementation parts of the GenICam Base module, 66
Init
Pylon::CPixelFormatConverter, 213
Pylon::CPixelFormatConverterBayer, 216
Pylon::CPixelFormatConverterGamma, 219
Pylon::CPixelFormatConverterGammaPacked, 222
Pylon::CPixelFormatConverterMonoPacked, 226
Pylon::CPixelFormatConverterMonoXX, 230
Pylon::CPixelFormatConverterRGB, 233
Pylon::CPixelFormatConverterTruncate, 236
Pylon::CPixelFormatConverterTruncate-Packed, 240
Pylon::CPixelFormatConverterYUV422, 243
Pylon::CPixelFormatConverterYUV422YUYV, 246
IsOpen
Pylon::IDevice, 261
Pylon::IPylonDevice, 272
IsVisible
GenApi_PublicInterface, 57

LegacyBinningVertical
Pylon::CBasler1394Camera, 144
Pylon::CBaslerGigECamera, 186
LineDebouncerTimeAbs
Pylon::CBasler1394Camera, 144
Pylon::CBaslerGigECamera, 186
LineDebouncerTimeRaw
Pylon::CBasler1394Camera, 144
Pylon::CBaslerGigECamera, 186
LineFormat
Pylon::CBasler1394Camera, 144
Pylon::CBaslerGigECamera, 186
LineInverter
Pylon::CBasler1394Camera, 144
Pylon::CBaslerGigECamera, 186
LineMode
Pylon::CBasler1394Camera, 144
Pylon::CBaslerGigECamera, 186
LineSelector
Pylon::CBasler1394Camera, 144
Pylon::CBaslerGigECamera, 187
LineSource
Pylon::CBasler1394Camera, 145
Pylon::CBaslerGigECamera, 187
LineStatus
Pylon::CBasler1394Camera, 145
Pylon::CBaslerGigECamera, 187
LineStatusAll
Pylon::CBasler1394Camera, 145
Pylon::CBaslerGigECamera, 187
LineTermination
Pylon::CBaslerGigECamera, 187
LUTEnable
Pylon::CBasler1394Camera, 145
Pylon::CBaslerGigECamera, 187
LUTIndex
Pylon::CBasler1394Camera, 145
Pylon::CBaslerGigECamera, 187
LUTSelector
Pylon::CBasler1394Camera, 145
Pylon::CBaslerGigECamera, 187
LUTValue
Pylon::CBasler1394Camera, 145
Pylon::CBaslerGigECamera, 188

MaxBufferSize
Pylon::CBasler1394StreamGrabber, 157
MaxNumBuffer
Pylon::CBasler1394StreamGrabber, 157

NumBuffer
Pylon::CBasler1394EventGrabber, 155
Pylon::CBaslerGigEEventGrabber, 198

OffsetX
Pylon::CBasler1394Camera, 146
Pylon::CBaslerGigECamera, 188
OffsetY
Pylon::CBasler1394Camera, 146
Pylon::CBaslerGigECamera, 188
Open
Pylon::IDevice, 262
Pylon::IPylonDevice, 273
operator+
Pylon, 75
operator<<
Base_PublicUtilities, 65
operator=
GenApi::IEnumerationT, 100
operator>>
Base_PublicUtilities, 65
operator|
PYLON_PUBLIC, 51

PacketSize
Pylon::CBasler1394Camera, 146
ParameterSelector
Pylon::CBasler1394Camera, 146

Pylon::CBaslerGigECamera, 188
PayloadSize
Pylon::CBasler1394Camera, 146
Pylon::CBaslerGigECamera, 188
PixelColorFilter
Pylon::CBasler1394Camera, 146
Pylon::CBaslerGigECamera, 188
PixelDynamicRangeMax
Pylon::CBasler1394Camera, 146
Pylon::CBaslerGigECamera, 188
PixelDynamicRangeMin
Pylon::CBasler1394Camera, 147
Pylon::CBaslerGigECamera, 189
PixelFormat
Pylon::CBasler1394Camera, 147
Pylon::CBaslerGigECamera, 189
PixelSize
Pylon, 75
Pylon::CBasler1394Camera, 147
Pylon::CBaslerGigECamera, 189
PrepareGrab
Pylon::IStreamGrabber, 276
Public interface and definitions in Pylon, 49
Public utilities of the GenApi Log module, 63
Public utilities of the GenApi module, 58
Public utilities of the GenICam Base module, 64
Pylon, 69
AccessMode, 74
BitPerPixel, 75
Canceled, 75
Control, 74
ConvertToRGB, 76
ConvertToRGBA, 76
Event, 74
Exclusive, 74
Failed, 75
GetPixelColorFilter, 75
Grabbed, 75
GrabStatus, 74
Idle, 75
operator+, 75
PixelSize, 75
PylonInitialize, 76
PylonTerminate, 76
Queued, 75
Stream, 74
WaitResult, 75
Pylon::AccessModeSet, 125
Pylon::AlertableWaitObject, 126
Pylon::AutoLock, 127
Pylon::Base_Callback1Body, 128
Pylon::Callback1, 129
Pylon::CBasler1394Camera, 130
AcquisitionFrameRateAbs, 138
AcquisitionFrameRateEnable, 138
AcquisitionLineRateAbs, 138
AcquisitionMode, 138
AcquisitionStart, 138
AcquisitionStop, 138
Attach, 138
BalanceRatioAbs, 139
BalanceRatioRaw, 139
BalanceRatioSelector, 139
BinningHorizontal, 139
BinningVertical, 139
BlackLevelRaw, 139
BlackLevelSelector, 139
CBasler1394Camera, 137
ChunkEnable, 140
ChunkFramecounter, 140
ChunkLineStatusAll, 140
ChunkModeActive, 140
ChunkSelector, 140
ChunkTimestamp, 140
ChunkTriggerinputcounter, 140
DeviceFirmwareVersion, 141
DeviceReset, 141
DeviceScanType, 141
DeviceSerialNumber, 141
DigitalShift, 141
EventNotification, 141
EventOverrunEventFrameID, 141
EventOverrunEventStreamChannelIndex, 141
EventOverrunEventTimestamp, 141
EventSelector, 142
ExposureEndEventFrameID, 142
ExposureEndEventStreamChannelIndex, 142
ExposureEndEventTimestamp, 142
ExposureMode, 142
ExposureTimeAbs, 142
ExposureTimeBaseAbs, 142
ExposureTimeBaseAbsEnable, 142
ExposureTimeRaw, 143
GainRaw, 143
GainSelector, 143
Gamma, 143
GammaEnable, 143
Height, 143
HeightMax, 143
LegacyBinningVertical, 144
LineDebouncerTimeAbs, 144
LineDebouncerTimeRaw, 144
LineFormat, 144
LineInverter, 144
LineMode, 144
LineSelector, 144
LineSource, 145
LineStatus, 145

LineStatusAll, 145
LUTEnable, 145
LUTIndex, 145
LUTSelector, 145
LUTValue, 145
OffsetX, 146
OffsetY, 146
PacketSize, 146
ParameterSelector, 146
PayloadSize, 146
PixelColorFilter, 146
PixelDynamicRangeMax, 146
PixelDynamicRangeMin, 147
PixelFormat, 147
PixelSize, 147
RecommendedPacketSize, 147
RemoveLimits, 147
ResultingFrameRateAbs, 147
ResultingLineRateAbs, 148
SensorHeight, 148
SensorWidth, 148
TemperatureAbs, 148
TemperatureSelector, 148
TestImageSelector, 148
TimerDelayAbs, 148
TimerDelayRaw, 148
TimerDelayTimebaseAbs, 149
TimerDurationAbs, 149
TimerDurationRaw, 149
TimerDurationTimebaseAbs, 149
TimerSelector, 149
TimerTriggerActivation, 149
TimerTriggerSource, 149
TriggerActivation, 150
TriggerMode, 150
TriggerSelector, 150
TriggerSoftware, 150
TriggerSource, 150
UserOutputSelector, 150
UserOutputValue, 151
UserOutputValueAll, 151
UserSetDefaultSelector, 151
UserSetLoad, 151
UserSetSave, 151
UserSetSelector, 151
Width, 152
WidthMax, 152
Pylon::CBasler1394DeviceInfo, 153
Pylon::CBasler1394EventGrabber, 155
Pylon::CBasler1394EventGrabber
Attach, 155
NumBuffer, 155
Pylon::CBasler1394StreamGrabber, 156
Pylon::CBasler1394StreamGrabber
Attach, 156
MaxBufferSize, 157
MaxNumBuffer, 157
Status, 157
ThreadPriority, 157
Pylon::CBasler1394TI, 158
Pylon::CBaslerGigECamera, 159
Pylon::CBaslerGigECamera
AcquisitionAbort, 171
AcquisitionFrameRateAbs, 171
AcquisitionFrameRateEnable, 172
AcquisitionLineRateAbs, 172
AcquisitionMode, 172
AcquisitionStart, 172
AcquisitionStop, 172
Attach, 171
BalanceRatioAbs, 172
BalanceRatioRaw, 172
BalanceRatioSelector, 172
BinningHorizontal, 173
BinningVertical, 173
BlackLevelAbs, 173
BlackLevelRaw, 173
BlackLevelSelector, 173
CBaslerGigECamera, 171
ChunkDynamicRangeMax, 173
ChunkDynamicRangeMin, 173
ChunkEnable, 174
ChunkFramecounter, 174
ChunkHeight, 174
ChunkLineStatusAll, 174
ChunkModeActive, 174
ChunkOffsetX, 174
ChunkOffsetY, 174
ChunkPayloadCRC16, 174
ChunkPixelFormat, 175
ChunkSelector, 175
ChunkStride, 175
ChunkTimestamp, 175
ChunkTriggerinputcounter, 175
ChunkWidth, 175
DeviceFirmwareVersion, 175
DeviceID, 175
DeviceManufacturerInfo, 176
DeviceModelName, 176
DeviceReset, 176
DeviceScanType, 176
DeviceUserID, 176
DeviceVendorName, 176
DeviceVersion, 176
DigitalShift, 176
EventNotification, 177
EventOverrunEventFrameID, 177
EventOverrunEventStreamChannelIndex, 177

EventOverrunEventTimestamp, 177
EventSelector, 177
ExposureEndEventFrameID, 177
ExposureEndEventStreamChannelIndex, 177
ExposureEndEventTimestamp, 177
ExposureMode, 178
ExposureTimeAbs, 178
ExposureTimeBaseAbs, 178
ExposureTimeBaseAbsEnable, 178
ExposureTimeRaw, 178
GainAbs, 178
GainRaw, 178
GainSelector, 179
Gamma, 179
GammaEnable, 179
GevCCP, 179
GevCurrentDefaultGateway, 179
GevCurrentIPAddress, 179
GevCurrentIPConfiguration, 179
GevCurrentSubnetMask, 179
GevDeviceModeCharacterSet, 180
GevDeviceModeIsBigEndian, 180
GevFirstURL, 180
GevHeartbeatTimeout, 180
GevInterfaceSelector, 180
GevLinkSpeed, 180
GevMACAddress, 180
GevMessageChannelCount, 180
GevNumberOfInterfaces, 181
GevPersistentDefaultGateway, 181
GevPersistentIPAddress, 181
GevPersistentSubnetMask, 181
GevSCBWA, 181
GevSCBWR, 181
GevSCBWRA, 182
GevSCDA, 182
GevSCDCT, 182
GevSCDMT, 182
GevSCFJM, 182
GevSCFTD, 182
GevSCPD, 182
GevSCPHostPort, 183
GevSCPInterfaceIndex, 183
GevSCPSPacketSize, 183
GevSecondURL, 183
GevStreamChannelCount, 183
GevStreamChannelSelector, 183
GevSupportedIPConfigurationDHCP, 183
GevSupportedIPConfigurationLLA, 184
GevSupportedIPConfigurationPersistentIP, 184
GevSupportedOptionalCommandsConcatenation, 184
GevSupportedOptionalCommandsEVENT, 184
GevSupportedOptionalCommandsEVENTDATA, 184
GevSupportedOptionalCommandsPACKETRESEND, 184
GevSupportedOptionalCommandsWRITEMEM, 184
GevTimestampControlLatch, 185
GevTimestampControlLatchReset, 185
GevTimestampControlReset, 185
GevTimestampTickFrequency, 185
GevTimestampValue, 185
GevVersionMajor, 185
GevVersionMinor, 185
Height, 185
HeightMax, 186
LegacyBinningVertical, 186
LineDebouncerTimeAbs, 186
LineDebouncerTimeRaw, 186
LineFormat, 186
LineInverter, 186
LineMode, 186
LineSelector, 187
LineSource, 187
LineStatus, 187
LineStatusAll, 187
LineTermination, 187
LUTEnable, 187
LUTIndex, 187
LUTSelector, 187
LUTValue, 188
OffsetX, 188
OffsetY, 188
ParameterSelector, 188
PayloadSize, 188
PixelColorFilter, 188
PixelDynamicRangeMax, 188
PixelDynamicRangeMin, 189
PixelFormat, 189
PixelSize, 189
RemoveLimits, 189
ResultingFrameRateAbs, 189
ResultingLineRateAbs, 189
SensorHeight, 190
SensorWidth, 190
SpatialCorrection, 190
SpatialCorrectionAmount, 190
SpatialCorrectionStartingLine, 190
TemperatureAbs, 190
TemperatureSelector, 190
TestImageSelector, 191
TimerDelayAbs, 191
TimerDelayRaw, 191
TimerDelayTimebaseAbs, 191
TimerDurationAbs, 191

TimerDurationRaw, 191
TimerDurationTimebaseAbs, 191
TimerSelector, 191
TimerTriggerActivation, 192
TimerTriggerSource, 192
TriggerActivation, 192
TriggerMode, 192
TriggerSelector, 192
TriggerSoftware, 192
TriggerSource, 192
UserOutputSelector, 193
UserOutputValue, 193
UserOutputValueAll, 193
UserOutputValueAllMask, 193
UserSetDefaultSelector, 193
UserSetLoad, 193
UserSetSave, 194
UserSetSelector, 194
Width, 194
WidthMax, 194
Pylon::CBaslerGigEDeviceInfo, 195
Pylon::CBaslerGigEEventGrabber, 197
Pylon::CBaslerGigEEventGrabber
Attach, 197
NumBuffer, 198
RetryCount, 198
Timeout, 198
Pylon::CBaslerGigEStreamGrabber, 199
Pylon::CBaslerGigEStreamGrabber
Attach, 200
Type, 200
Pylon::CBaslerGigETLParams, 201
Pylon::CBaslerGigETLParams
Attach, 201
Pylon::CChunkParser, 202
Pylon::CChunkParser
AttachBuffer, 203
CheckCRC, 203
DetachBuffer, 203
HasCRC, 203
UpdateBuffer, 203
Pylon::CDeviceInfo, 205
Pylon::CEventGrabberProxy, 207
Pylon::CInfoBase, 208
Pylon::CLock, 209
Pylon::CNodeMapProxy, 210
Pylon::CPixelFormatConverter, 211
Pylon::CPixelFormatConverter
CheckInput, 212
CheckParameter_Impl, 212
Convert, 212
Init, 213
Pylon::CPixelFormatConverterBayer, 214
Pylon::CPixelFormatConverterBayer
CheckInput, 215
CheckParameter_Impl, 215
Convert, 215
Init, 216
Pylon::CPixelFormatConverterGamma, 217
Pylon::CPixelFormatConverterGamma
CheckInput, 218
CheckParameter_Impl, 218
Convert, 219
Init, 219
SetGamma, 219
Pylon::CPixelFormatConverterGammaPacked, 220
Pylon::CPixelFormatConverterGammaPacked
CheckInput, 221
CheckParameter_Impl, 221
Convert, 222
Init, 222
SetGamma, 222
Unpack10, 222
Unpack12, 223
Pylon::CPixelFormatConverterMonoPacked, 224
Pylon::CPixelFormatConverterMonoPacked
CheckInput, 225
CheckParameter_Impl, 225
Convert, 226
Init, 226
Unpack10, 226
Unpack12, 227
Pylon::CPixelFormatConverterMonoXX, 228
Pylon::CPixelFormatConverterMonoXX
CheckInput, 229
CheckParameter_Impl, 229
Convert, 230
Init, 230
Pylon::CPixelFormatConverterRGB, 231
Pylon::CPixelFormatConverterRGB
CheckInput, 232
CheckParameter_Impl, 232
Convert, 232
Init, 233
Pylon::CPixelFormatConverterTruncate, 234
Pylon::CPixelFormatConverterTruncate
CheckInput, 235
CheckParameter_Impl, 235
Convert, 236
Init, 236
SetHighOffsetBit, 236
Pylon::CPixelFormatConverterTruncatePacked, 238
Pylon::CPixelFormatConverterTruncatePacked
CheckInput, 239
CheckParameter_Impl, 239
Convert, 240
Init, 240
SetHighOffsetBit, 241

Unpack10, 241
Unpack12, 241
Pylon::CPixelFormatConverterYUV422, 242
Pylon::CPixelFormatConverterYUV422
CheckInput, 243
Convert, 243
Init, 243
Pylon::CPixelFormatConverterYUV422YUYV, 245
Pylon::CPixelFormatConverterYUV422YUYV
CheckInput, 246
Convert, 246
Init, 246
Pylon::CPylonDeviceProxy, 248
Pylon::CStreamGrabberProxy, 249
Pylon::CTlFactory, 250
Pylon::CTlFactory
PylonTerminate, 251
Pylon::DeviceInfoList, 252
Pylon::EventResult, 253
Pylon::Function_CallbackBody, 254
Pylon::GrabResult, 255
Pylon::GrabResult
FrameNr, 256
GetOffsetX, 256
GetOffsetY, 256
GetPaddingX, 256
GetPaddingY, 257
GetPixelType, 257
GetSizeX, 257
GetSizeY, 257
GetTimeStamp, 257
Pylon::IChunkParser, 258
Pylon::IChunkParser
AttachBuffer, 259
CheckCRC, 259
DetachBuffer, 259
HasCRC, 259
UpdateBuffer, 260
Pylon::IDevice, 261
Close, 261
GetDeviceInfo, 261
IsOpen, 261
Open, 262
Pylon::IDeviceFactory, 263
Pylon::IDeviceFactory
CreateDevice, 263
DestroyDevice, 263
EnumerateDevices, 263
Pylon::IEventAdapter, 265
Pylon::IEventGrabber, 266
Pylon::IEventGrabber
GetNodeMap, 266
GetWaitObject, 266
RetrieveEvent, 266
Pylon::IInfo, 268
Pylon::IProperties, 269
Pylon::IPylonDevice, 270
Pylon::IPylonDevice
Close, 271
CreateChunkParser, 271
CreateEventAdapter, 271
DeregisterRemovalCallback, 271
DestroyChunkParser, 271
GetDeviceInfo, 271
GetEventGrabber, 272
GetNodeMap, 272
GetStreamGrabber, 272
GetTLNodeMap, 272
IsOpen, 272
Open, 273
RegisterRemovalCallback, 273
Pylon::IStreamGrabber, 275
Pylon::IStreamGrabber
CancelGrab, 276
Close, 276
DeregisterBuffer, 276
FinishGrab, 276
GetNodeMap, 276
GetWaitObject, 276
PrepareGrab, 276
QueueBuffer, 277
RegisterBuffer, 277
RetrieveResult, 277
Pylon::Key, 78
Pylon::LockableObject, 278
Pylon::LockableObject::Lock, 279
Pylon::Member_CallbackBody, 280
Pylon::PylonAutoInitTerm, 281
Pylon::SystemWaitObject, 282
Pylon::TlInfoList, 283
Pylon::WaitObject, 284
Pylon::WaitObjects, 285
Pylon::WaitObjects
WaitForAll, 285
WaitForAny, 286
Pylon::WaitObjectSet, 287
PYLON_PUBLIC
operator|, 51
RegisterRemovalCallback, 51
PylonInitialize
Pylon, 76
PylonTerminate
Pylon, 76
Pylon::CTlFactory, 251

QueueBuffer
Pylon::IStreamGrabber, 277
Queued

Pylon, 75

RecommendedPacketSize
Pylon::CBasler1394Camera, 147
RegisterBuffer
Pylon::IStreamGrabber, 277
RegisterRemovalCallback
Pylon::IPylonDevice, 273
PYLON_PUBLIC, 51
RemoveLimits
Pylon::CBasler1394Camera, 147
Pylon::CBaslerGigECamera, 189
ResultingFrameRateAbs
Pylon::CBasler1394Camera, 147
Pylon::CBaslerGigECamera, 189
ResultingLineRateAbs
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 189
RetrieveEvent
Pylon::IEventGrabber, 266
RetrieveResult
Pylon::IStreamGrabber, 277
RetryCount
Pylon::CBaslerGigEEventGrabber, 198

SensorHeight
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 190
SensorWidth
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 190
SetGamma
Pylon::CPixelFormatConverterGamma, 219
Pylon::CPixelFormatConverterGammaPacked, 222
SetHighOffsetBit
Pylon::CPixelFormatConverterTruncate, 236
Pylon::CPixelFormatConverterTruncate-Packed, 241
SpatialCorrection
Pylon::CBaslerGigECamera, 190
SpatialCorrectionAmount
Pylon::CBaslerGigECamera, 190
SpatialCorrectionStartingLine
Pylon::CBaslerGigECamera, 190
Status
Pylon::CBasler1394StreamGrabber, 157
Stream
Pylon, 74

TemperatureAbs
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 190
TemperatureSelector
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 190
TestImageSelector
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 191
The GigE specific definitions, 52
The IEEE1394 specific definitions, 53
ThreadPriority
Pylon::CBasler1394StreamGrabber, 157
Timeout
Pylon::CBaslerGigEEventGrabber, 198
TimerDelayAbs
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 191
TimerDelayRaw
Pylon::CBasler1394Camera, 148
Pylon::CBaslerGigECamera, 191
TimerDelayTimebaseAbs
Pylon::CBasler1394Camera, 149
Pylon::CBaslerGigECamera, 191
TimerDurationAbs
Pylon::CBasler1394Camera, 149
Pylon::CBaslerGigECamera, 191
TimerDurationRaw
Pylon::CBasler1394Camera, 149
Pylon::CBaslerGigECamera, 191
TimerDurationTimebaseAbs
Pylon::CBasler1394Camera, 149
Pylon::CBaslerGigECamera, 191
TimerSelector
Pylon::CBasler1394Camera, 149
Pylon::CBaslerGigECamera, 191
TimerTriggerActivation
Pylon::CBasler1394Camera, 149
Pylon::CBaslerGigECamera, 192
TimerTriggerSource
Pylon::CBasler1394Camera, 149
Pylon::CBaslerGigECamera, 192
TriggerActivation
Pylon::CBasler1394Camera, 150
Pylon::CBaslerGigECamera, 192
TriggerMode
Pylon::CBasler1394Camera, 150
Pylon::CBaslerGigECamera, 192
TriggerSelector
Pylon::CBasler1394Camera, 150
Pylon::CBaslerGigECamera, 192
TriggerSoftware
Pylon::CBasler1394Camera, 150
Pylon::CBaslerGigECamera, 192
TriggerSource
Pylon::CBasler1394Camera, 150
Pylon::CBaslerGigECamera, 192
Type

Pylon::CBaslerGigEStreamGrabber, 200
Pylon::CBaslerGigECamera, 194

Unpack10
Pylon::CPixelFormatConverterGammaPacked, 222
Pylon::CPixelFormatConverterMonoPacked, 226
Pylon::CPixelFormatConverterTruncate-Packed, 241
Unpack12
Pylon::CPixelFormatConverterGammaPacked, 223
Pylon::CPixelFormatConverterMonoPacked, 227
Pylon::CPixelFormatConverterTruncate-Packed, 241
UpdateBuffer
Pylon::CChunkParser, 203
Pylon::IChunkParser, 260
UserOutputSelector
Pylon::CBasler1394Camera, 150
Pylon::CBaslerGigECamera, 193
UserOutputValue
Pylon::CBasler1394Camera, 151
Pylon::CBaslerGigECamera, 193
UserOutputValueAll
Pylon::CBasler1394Camera, 151
Pylon::CBaslerGigECamera, 193
UserOutputValueAllMask
Pylon::CBaslerGigECamera, 193
UserSetDefaultSelector
Pylon::CBasler1394Camera, 151
Pylon::CBaslerGigECamera, 193
UserSetLoad
Pylon::CBasler1394Camera, 151
Pylon::CBaslerGigECamera, 193
UserSetSave
Pylon::CBasler1394Camera, 151
Pylon::CBaslerGigECamera, 194
UserSetSelector
Pylon::CBasler1394Camera, 151
Pylon::CBaslerGigECamera, 194

WaitForAll
Pylon::WaitObjects, 285
WaitForAny
Pylon::WaitObjects, 286
WaitResult
Pylon, 75
Width
Pylon::CBasler1394Camera, 152
Pylon::CBaslerGigECamera, 194
WidthMax
Pylon::CBasler1394Camera, 152