

# Imperx Camera SDK

## 1.4.0.52

Generated by Doxygen 1.8.13



# Contents

<b>1 Imperx Camera SDK</b>	<b>1</b>
1.1 Main Page . . . . .	1
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List . . . . .	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy . . . . .	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List . . . . .	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 IpxCam Namespace Reference . . . . .	9
5.1.1 Detailed Description . . . . .	10
5.1.2 Typedef Documentation . . . . .	10
5.1.2.1 InterfaceList . . . . .	10
5.1.2.2 DeviceInfoList . . . . .	10
5.1.2.3 DeviceList . . . . .	11
5.1.2.4 EventCallback2 . . . . .	11
5.1.3 Enumeration Type Documentation . . . . .	11
5.1.3.1 InterfaceType . . . . .	11
5.1.3.2 FlushOperation . . . . .	11

5.1.3.3	ServiceFileType	12
5.1.3.4	DeviceAccess	12
5.1.4	Function Documentation	12
5.1.4.1	IpxCam_GetSystem()	13
5.2	IpxGenParam Namespace Reference	13
5.2.1	Detailed Description	14
5.2.2	Enumeration Type Documentation	14
5.2.2.1	ParamType	14
5.2.2.2	NameSpace	15
5.2.2.3	Visibility	15
5.3	IpxGui Namespace Reference	15
5.3.1	Detailed Description	17
5.3.2	Enumeration Type Documentation	17
5.3.2.1	Visibility	17
5.3.3	Function Documentation	17
5.3.3.1	CreateGenParamTreeViewForArrayA()	17
5.3.3.2	CreateGenParamTreeViewForArrayW()	18
5.3.3.3	CreateGenParamTreeViewForNodemapA()	19
5.3.3.4	CreateGenParamTreeViewForNodemapW()	20
5.3.3.5	DestroyGenParamTreeView()	21
5.3.3.6	SelectCameraA()	22
5.3.3.7	SelectCameraW()	22
5.3.3.8	ShowCamConfigDialog()	23
5.3.3.9	ShowFrameABDialog()	24
5.3.3.10	ShowTriggerDialog()	24
5.3.3.11	ShowPulseDialog()	25
5.3.3.12	ShowStrobeDialog()	26
5.3.3.13	ShowOutputDialog()	26
5.3.3.14	ShowColorDialog()	27

<b>6</b>	<b>Class Documentation</b>	<b>29</b>
6.1	IpxGenParam::Array Class Reference	29
6.1.1	Detailed Description	30
6.1.2	Constructor & Destructor Documentation	31
6.1.2.1	~Array()	31
6.1.3	Member Function Documentation	31
6.1.3.1	GetParam()	31
6.1.3.2	GetBoolean()	32
6.1.3.3	GetCommand()	32
6.1.3.4	GetEnum()	33
6.1.3.5	GetFloat()	33
6.1.3.6	GetInt()	34
6.1.3.7	GetString()	34
6.1.3.8	GetRootCategory()	35
6.1.3.9	GetNodeMap()	35
6.1.3.10	GetCount()	36
6.1.3.11	GetParamByIndex()	36
6.1.3.12	SetBooleanValue()	36
6.1.3.13	GetBooleanValue()	37
6.1.3.14	SetEnumValueStr()	37
6.1.3.15	SetEnumValue()	38
6.1.3.16	GetEnumValueStr()	38
6.1.3.17	GetEnumValue()	39
6.1.3.18	SetFloatValue()	39
6.1.3.19	GetFloatValue()	40
6.1.3.20	SetIntegerValue()	41
6.1.3.21	GetIntegerValue()	41
6.1.3.22	SetStringValue()	42

6.1.3.23	GetStringValue()	42
6.1.3.24	ExecuteCommand()	43
6.1.3.25	IsCommandDone()	43
6.1.3.26	Poll()	44
6.2	IpxGenParam::Boolean Class Reference	44
6.2.1	Detailed Description	45
6.2.2	Member Function Documentation	45
6.2.2.1	GetType()	45
6.2.2.2	SetValue()	45
6.2.2.3	GetValue()	46
6.3	IpxCam::Buffer Class Reference	46
6.3.1	Detailed Description	47
6.3.2	Constructor & Destructor Documentation	48
6.3.2.1	~Buffer()	48
6.3.3	Member Function Documentation	48
6.3.3.1	GetImage()	48
6.3.3.2	GetBufferPtr()	48
6.3.3.3	GetImageOffset()	49
6.3.3.4	GetBufferSize()	49
6.3.3.5	GetPixelFormat()	49
6.3.3.6	GetUserPtr()	49
6.3.3.7	GetTimestamp()	50
6.3.3.8	GetFrameID()	50
6.3.3.9	IsIncomplete()	50
6.3.3.10	GetWidth()	50
6.3.3.11	GetHeight()	51
6.3.3.12	GetXOffset()	51
6.3.3.13	GetYOffset()	51

6.3.3.14	<a href="#">GetXPadding()</a>	51
6.3.3.15	<a href="#">GetYPadding()</a>	52
6.3.3.16	<a href="#">GetDeliveredHeight()</a>	52
6.3.3.17	<a href="#">IsKacFrameB()</a>	52
6.4	<a href="#">IpxGenParam::Category Class Reference</a>	53
6.4.1	<a href="#">Detailed Description</a>	53
6.4.2	<a href="#">Member Function Documentation</a>	53
6.4.2.1	<a href="#">GetType()</a>	54
6.4.2.2	<a href="#">GetCount()</a>	54
6.4.2.3	<a href="#">GetParamByIndex()</a>	54
6.5	<a href="#">IpxGenParam::Command Class Reference</a>	55
6.5.1	<a href="#">Detailed Description</a>	55
6.5.2	<a href="#">Member Function Documentation</a>	56
6.5.2.1	<a href="#">GetType()</a>	56
6.5.2.2	<a href="#">Execute()</a>	56
6.5.2.3	<a href="#">IsDone()</a>	56
6.6	<a href="#">IpxCam::Device Class Reference</a>	57
6.6.1	<a href="#">Detailed Description</a>	59
6.6.2	<a href="#">Member Typedef Documentation</a>	59
6.6.2.1	<a href="#">UploadEventCallback</a>	59
6.6.3	<a href="#">Member Enumeration Documentation</a>	59
6.6.3.1	<a href="#">UploadEventType</a>	59
6.6.3.2	<a href="#">Endianness</a>	59
6.6.4	<a href="#">Constructor &amp; Destructor Documentation</a>	60
6.6.4.1	<a href="#">~Device()</a>	60
6.6.5	<a href="#">Member Function Documentation</a>	60
6.6.5.1	<a href="#">GetNumStreams()</a>	60
6.6.5.2	<a href="#">GetStreamByIndex()</a>	60

6.6.5.3	GetStreamById()	61
6.6.5.4	GetInfo()	61
6.6.5.5	ReadMem()	61
6.6.5.6	WriteMem()	62
6.6.5.7	UploadFile()	62
6.6.5.8	RegisterEvent()	63
6.6.5.9	UnRegisterEvent()	63
6.6.5.10	GetTransportParameters()	64
6.6.5.11	GetCameraParameters()	64
6.6.5.12	SaveConfiguration()	65
6.6.5.13	LoadConfiguration()	65
6.6.5.14	GetEndianness()	66
6.7	IpxCam::DeviceInfo Class Reference	66
6.7.1	Detailed Description	67
6.7.2	Constructor & Destructor Documentation	67
6.7.2.1	~DeviceInfo()	67
6.7.3	Member Function Documentation	67
6.7.3.1	GetInterface()	67
6.7.3.2	GetID()	67
6.7.3.3	GetVendor()	68
6.7.3.4	GetModel()	68
6.7.3.5	GetDisplayName()	68
6.7.3.6	GetUserDefinedName()	68
6.7.3.7	GetSerialNumber()	69
6.7.3.8	GetVersion()	69
6.7.3.9	GetAccessStatus()	69
6.7.3.10	GetUSB3HostInfo()	70
6.7.3.11	ForceIP() <sup>[1/2]</sup>	70



6.7.3.12	ForceIP() [2/2]	70
6.8	IpxGenParam::Enum Class Reference	71
6.8.1	Detailed Description	72
6.8.2	Member Function Documentation	72
6.8.2.1	GetType()	72
6.8.2.2	GetEnumEntriesCount()	72
6.8.2.3	GetEnumEntryByIndex()	73
6.8.2.4	GetEnumEntryByName()	73
6.8.2.5	GetEnumEntryByValue()	74
6.8.2.6	GetValue()	74
6.8.2.7	GetValueStr()	75
6.8.2.8	SetValue()	75
6.8.2.9	SetValueStr()	76
6.9	IpxGenParam::EnumEntry Class Reference	76
6.9.1	Detailed Description	77
6.9.2	Member Function Documentation	77
6.9.2.1	GetType()	77
6.9.2.2	GetValue()	77
6.9.2.3	GetValueStr()	78
6.10	IpxGenParam::Float Class Reference	78
6.10.1	Detailed Description	79
6.10.2	Member Function Documentation	79
6.10.2.1	GetType()	80
6.10.2.2	SetValue()	80
6.10.2.3	GetValue()	80
6.10.2.4	GetMin()	81
6.10.2.5	GetMax()	81
6.10.2.6	GetUnit()	82

6.11 IpxGui::IpxGenParamTreeView Class Reference . . . . .	83
6.11.1 Detailed Description . . . . .	83
6.11.2 Constructor & Destructor Documentation . . . . .	84
6.11.2.1 ~IpxGenParamTreeView() . . . . .	84
6.11.3 Member Function Documentation . . . . .	85
6.11.3.1 setParams() [1/2] . . . . .	85
6.11.3.2 setParams() [2/2] . . . . .	85
6.11.3.3 clearParams() . . . . .	86
6.11.3.4 visibility() . . . . .	86
6.11.3.5 setVisibility() . . . . .	86
6.11.3.6 saveState() . . . . .	87
6.11.3.7 loadState() . . . . .	87
6.11.3.8 setPollingTime() . . . . .	87
6.11.3.9 getPollingTime() . . . . .	88
6.12 IpxGenParam::Int Class Reference . . . . .	88
6.12.1 Detailed Description . . . . .	89
6.12.2 Member Function Documentation . . . . .	89
6.12.2.1 GetType() . . . . .	89
6.12.2.2 SetValue() . . . . .	89
6.12.2.3 GetValue() . . . . .	90
6.12.2.4 GetMin() . . . . .	90
6.12.2.5 GetMax() . . . . .	91
6.12.2.6 GetIncrement() . . . . .	91
6.13 IpxCam::Interface Class Reference . . . . .	92
6.13.1 Detailed Description . . . . .	93
6.13.2 Constructor & Destructor Documentation . . . . .	93
6.13.2.1 ~Interface() . . . . .	93
6.13.3 Member Function Documentation . . . . .	93

6.13.3.1	GetDeviceInfoList()	94
6.13.3.2	GetFirstDeviceInfo()	94
6.13.3.3	GetDeviceInfoById()	94
6.13.3.4	ReEnumerateDevices()	95
6.13.3.5	GetDescription()	95
6.13.3.6	GetType()	96
6.13.3.7	GetId()	96
6.13.3.8	GetVersion()	96
6.13.3.9	RegisterEvent()	96
6.13.3.10	UnRegisterEvent()	97
6.13.3.11	GetParameters()	97
6.13.3.12	CreateDeviceFromConfig()	98
6.14	IpxCam::List<_T> Class Template Reference	98
6.14.1	Detailed Description	99
6.14.2	Member Typedef Documentation	100
6.14.2.1	elem_type	101
6.14.3	Constructor & Destructor Documentation	101
6.14.3.1	~List()	101
6.14.4	Member Function Documentation	101
6.14.4.1	Release()	101
6.14.4.2	GetCount()	101
6.14.4.3	GetFirst()	102
6.14.4.4	GetNext()	102
6.15	IpxGenParam::Param Class Reference	102
6.15.1	Detailed Description	104
6.15.2	Constructor & Destructor Documentation	104
6.15.2.1	~Param()	105
6.15.3	Member Function Documentation	105

6.15.3.1	<a href="#">GetType()</a>	105
6.15.3.2	<a href="#">GetName()</a>	105
6.15.3.3	<a href="#">GetToolTip()</a>	105
6.15.3.4	<a href="#">GetDescription()</a>	106
6.15.3.5	<a href="#">GetDisplayName()</a>	106
6.15.3.6	<a href="#">GetVisibility()</a>	106
6.15.3.7	<a href="#">IsValueCached()</a>	106
6.15.3.8	<a href="#">IsAvailable()</a>	107
6.15.3.9	<a href="#">IsWritable()</a>	107
6.15.3.10	<a href="#">IsReadable()</a>	107
6.15.3.11	<a href="#">IsStreamable()</a>	107
6.15.3.12	<a href="#">IsVisible()</a>	107
6.15.3.13	<a href="#">RegisterEventSink()</a>	108
6.15.3.14	<a href="#">UnregisterEventSink()</a>	108
6.15.3.15	<a href="#">GetNode()</a>	109
6.15.3.16	<a href="#">ToCategory()</a>	109
6.15.3.17	<a href="#">ToBoolean()</a>	109
6.15.3.18	<a href="#">ToCommand()</a>	109
6.15.3.19	<a href="#">ToEnumEntry()</a>	110
6.15.3.20	<a href="#">ToEnum()</a>	110
6.15.3.21	<a href="#">ToFloat()</a>	110
6.15.3.22	<a href="#">ToInt()</a>	110
6.15.3.23	<a href="#">ToString()</a>	111
6.16	<a href="#">IpxGenParam::ParamEventSink Class Reference</a>	111
6.16.1	<a href="#">Detailed Description</a>	111
6.16.2	<a href="#">Constructor &amp; Destructor Documentation</a>	111
6.16.2.1	<a href="#">~ParamEventSink()</a>	111
6.16.3	<a href="#">Member Function Documentation</a>	112

6.16.3.1 OnParameterUpdate()	112
6.17 IpxCam::Stream Class Reference	113
6.17.1 Detailed Description	114
6.17.2 Constructor & Destructor Documentation	114
6.17.2.1 ~Stream()	114
6.17.3 Member Function Documentation	115
6.17.3.1 Release()	115
6.17.3.2 CreateBuffer()	115
6.17.3.3 SetBuffer()	115
6.17.3.4 RevokeBuffer()	116
6.17.3.5 QueueBuffer()	116
6.17.3.6 GetBuffer()	117
6.17.3.7 CancelBuffer()	117
6.17.3.8 FlushBuffers()	118
6.17.3.9 StartAcquisition()	119
6.17.3.10 StopAcquisition()	119
6.17.3.11 AllocBufferQueue()	120
6.17.3.12 ReleaseBufferQueue()	120
6.17.3.13 GetBufferQueueSize()	120
6.17.3.14 RegisterEvent()	121
6.17.3.15 UnRegisterEvent()	121
6.17.3.16 GetParameters()	121
6.17.3.17 GetNumDelivered()	122
6.17.3.18 GetNumUnderrun()	122
6.17.3.19 GetNumAnnounced()	122
6.17.3.20 GetNumQueued()	123
6.17.3.21 GetNumAwaitDelivery()	123
6.17.3.22 GetBufferSize()	123

6.17.3.23 IsGrabbing()	123
6.17.3.24 GetMinNumBuffers()	124
6.17.3.25 GetBufferAlignment()	124
6.18 IpxGenParam::String Class Reference	124
6.18.1 Detailed Description	125
6.18.2 Member Function Documentation	125
6.18.2.1 GetType()	125
6.18.2.2 GetMaxLength()	125
6.18.2.3 GetValue()	126
6.18.2.4 SetValue()	126
6.19 IpxCam::System Class Reference	128
6.19.1 Detailed Description	129
6.19.2 Constructor & Destructor Documentation	129
6.19.2.1 ~System()	130
6.19.3 Member Function Documentation	130
6.19.3.1 Release()	130
6.19.3.2 GetInterfaceList()	130
6.19.3.3 GetInterfaceById()	131
6.19.3.4 GetDisplayName()	132
6.19.3.5 GetVersion()	132
6.19.3.6 CreateDeviceFromConfig()	133
6.19.3.7 RegisterGenTLProvider()	133
6.20 IpxCam::Device::UploadEventData Struct Reference	133
6.20.1 Detailed Description	134

## Chapter 1

# Imperx Camera SDK

The Imperx Camera SDK is designed to provide software developers with API methods for ease of integrating Imperx cameras into their software application. The API consists of two namespaces [lpxCam](#) and [lpxGenParam](#) which contains enum, classes and functions which can be used to control the camera parameters and acquire images or video from the Imperx cameras. The [lpxCam](#) namespace provides the scope to the GenICam GenTL transport layer interface to acquire images with an Imperx Camera. The [lpxGenParam](#) namespace provides the scope to the GenICam features.

### 1.1 Main Page

The [lpxCam](#) namespace consist of several main classes that represent the GenTL modules. The main classes are

**System** - The System class is the entry point to the GenTL Producer software driver.

**Interface** - The Interface class provides method to represents an individual physical interface, like GigE or USB3

**Device** - The Device class provides methods to enable the communication with the camera device and enumerate/instantiate the video data streams.

**Stream** -The Stream class purpose is to access the image buffer data acquirement from the Acquisition engine.

**Buffer** - The Buffer class contains the methods to access the image data and parameters of the acquired image buffer.

#### Example of GenTL System Hierarchy





## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">lpxCam</a>	A namespace providing scope to the GenICam GenTL transport layer interface to acquire images with an Imperx Camera using enum, classes, etc . . . . .	9
<a href="#">lpxGenParam</a>	A namespace providing scope to the GenICam features using enum, classes, etc . . . . .	13
<a href="#">lpxGui</a>	The lpxGUI namespace is a declarative region that provides a scope to the Imperx Camera GUI API classes and functions . . . . .	15



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

IpxGenParam::Array . . . . .	29
IpxCam::Buffer . . . . .	46
IpxCam::Device . . . . .	57
IpxCam::DeviceInfo . . . . .	66
IpxGui::IpxGenParamTreeView . . . . .	83
IpxCam::Interface . . . . .	92
IpxCam::List<_T> . . . . .	98
IpxGenParam::Param . . . . .	102
IpxGenParam::Boolean . . . . .	44
IpxGenParam::Category . . . . .	53
IpxGenParam::Command . . . . .	55
IpxGenParam::Enum . . . . .	71
IpxGenParam::EnumEntry . . . . .	76
IpxGenParam::Float . . . . .	78
IpxGenParam::Int . . . . .	88
IpxGenParam::String . . . . .	124
IpxGenParam::ParamEventSink . . . . .	111
IpxCam::Stream . . . . .	113
IpxCam::System . . . . .	128
IpxCam::Device::UploadEventData . . . . .	133



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">IpxGenParam::Array</a>	This class contains functions that can access each node from the camera descriptor XML file by type and name . . . . .	29
<a href="#">IpxGenParam::Boolean</a>	A class containing methods that map the integer element value of a GenICam IBoolean interface feature to true or false . . . . .	44
<a href="#">IpxCam::Buffer</a>	<a href="#">Buffer</a> module in the GenTL module hierarchy . . . . .	46
<a href="#">IpxGenParam::Category</a>	A class containing methods that the user can access the categories GenICam features. It will access the node object's of an ICategory interface. Each feature of a device will be placed in a <a href="#">Category</a> . The <a href="#">Category</a> feature is used to present the user with a group of features for the named category . .	53
<a href="#">IpxGenParam::Command</a>	A Class for GenICam <a href="#">Command</a> properties contains methods that lets the user submit a command for execution as well as poll the status . . . . .	55
<a href="#">IpxCam::Device</a>	<a href="#">Device</a> module in the GenTL module hierarchy . . . . .	57
<a href="#">IpxCam::DeviceInfo</a>	<a href="#">DeviceInfo</a> class provides the information about the device . . . . .	66
<a href="#">IpxGenParam::Enum</a>	Interface Class for Enumeration properties . . . . .	71
<a href="#">IpxGenParam::EnumEntry</a>	A Class for GenICam <a href="#">Enum</a> Entries has methods to access the Enumeration node <a href="#">Enum</a> Entry of the GenICam . . . . .	76
<a href="#">IpxGenParam::Float</a>	Interface Class for GenICam <a href="#">Float</a> properties . . . . .	78
<a href="#">IpxGui::IpxGenParamTreeView</a>	Composed of functions to set and clear parameters of the node tree of the camera. The node tree can be set with the current parameters stored in the <a href="#">IpxGenParam::Array</a> and <a href="#">GenApi::INodeMap</a> class . . . . .	83
<a href="#">IpxGenParam::Int</a>	Interface Class for GenICam <a href="#">Int</a> properties . . . . .	88

---

<a href="#">IpxCam::Interface</a>	
<a href="#">Interface</a> module in the GenTL module hierarchy	92
<a href="#">IpxCam::List&lt;_T&gt;</a>	
The <a href="#">List</a> class is used to list the specified template type objects	98
<a href="#">IpxGenParam::Param</a>	
A Class for accessing the GenICam feature node parameters of the Camera Descriptor File	102
<a href="#">IpxGenParam::ParamEventSink</a>	
An Event Sink class designed to receive incoming events from Parameter Node Updates	111
<a href="#">IpxCam::Stream</a>	
Data stream module in the GenTL module hierarchy	113
<a href="#">IpxGenParam::String</a>	
Interface Class for GenICam <a href="#">String</a> properties	124
<a href="#">IpxCam::System</a>	
Abstraction of the system module of the GenTL module hierarchy. The <a href="#">System</a> class is the entry point to the GenTL Producer software driver	128
<a href="#">IpxCam::Device::UploadEventData</a>	
A structure representing data for uploading to a device	133

## Chapter 5

# Namespace Documentation

### 5.1 IpxCam Namespace Reference

A namespace providing scope to the GenICam GenTL transport layer interface to acquire images with an Imperx Camera using enum, classes, etc.

#### Classes

- class [Buffer](#)  
*The [Buffer](#) class represents the buffer module in the GenTL module hierarchy.*
- class [Device](#)  
*The [Device](#) class represents the device module in the GenTL module hierarchy.*
- class [DeviceInfo](#)  
*[DeviceInfo](#) class provides the information about the device.*
- class [Interface](#)  
*The [Interface](#) class represents a interface module in the GenTL module hierarchy.*
- class [List](#)  
*The [List](#) class is used to list the specified template type objects.*
- class [Stream](#)  
*The [Stream](#) class represents the data stream module in the GenTL module hierarchy.*
- class [System](#)  
*The [System](#) class represents an abstraction of the system module of the GenTL module hierarchy. The [System](#) class is the entry point to the GenTL Producer software driver.*

#### Typedefs

- typedef [List](#)< [Interface](#) > [InterfaceList](#)
- typedef [List](#)< [DeviceInfo](#) > [DeviceInfoList](#)
- typedef [List](#)< [Device](#) > [DeviceList](#)
- typedef void IPXCAM\_CALL [EventCallback2](#)(uint32\_t eventType, const void \*eventData, size\_t eventSize, void \*pPrivate)

## Enumerations

- enum [InterfaceType](#) : uint32\_t {  
[USB3Vision](#) = 1, [GigEVision](#) = 2, [CameraLink](#) = 3, [CoaxPress](#) = 4,  
[HdSdi](#) = 5, [AllInterfaces](#) = 0xff }  
*An enum of Interface Types. Interface Node Types representing physical interface in the system.*
- enum [FlushOperation](#) : uint32\_t { [Flush\\_OutputDiscard](#) = 1, [Flush\\_AllToInput](#) = 2, [Flush\\_UnqueuedToInput](#) = 3,  
[Flush\\_AllDiscard](#) = 4 }  
*An enum of Flush Operations. Flush Operations Types.*
- enum [ServiceFileType](#) : uint32\_t { [FileLUT](#) = 1, [FileDPC](#) = 2, [FileHPC](#) = 3, [FileFFC](#) = 4 }  
*An enum of Service File Types. Service File Types.*
- enum [DeviceAccess](#) : uint32\_t { [ReadOnly](#) = 0, [Control](#) = 1, [Exclusive](#) = 2 }  
*An enum of Device Access.*

## Functions

- IPXCAM\_EXTERN\_C IPX\_CAMERA\_API [System](#) \* [IpxCam\\_GetSystem](#) ()  
*This method returns the [System](#) module. It is the entry point to the GenTL Module hierarchy.*

### 5.1.1 Detailed Description

A namespace providing scope to the GenICam GenTL transport layer interface to acquire images with an Imperx Camera using enum, classes, etc.

A more detailed class description.

### 5.1.2 Typedef Documentation

#### 5.1.2.1 InterfaceList

```
typedef List<Interface> IpxCam::InterfaceList
```

[Interface List](#)

#### 5.1.2.2 DeviceInfoList

```
typedef List<DeviceInfo> IpxCam::DeviceInfoList
```

[Device Info List](#)



### 5.1.2.3 DeviceList

```
typedef List<Device> IpxCam::DeviceList
```

[Device List](#)

### 5.1.2.4 EventCallback2

```
typedef void IPXCAM_CALL IpxCam::EventCallback2(uint32_t eventType, const void *eventData, size_t eventSize, void *pPrivate)
```

EventCallback param[in] eventData pointer to event Data param[in] eventSize event Size param[in] pPrivate pointer to the context Data

## 5.1.3 Enumeration Type Documentation

### 5.1.3.1 InterfaceType

```
enum IpxCam::InterfaceType : uint32_t
```

An enum of [Interface](#) Types. [Interface](#) Node Types representing physical interface in the system.

Enumerator

USB3Vision	Enum value USB3Vision.
GigEVision	Enum value GigEVision.
CameraLink	Enum value CameraLink.
CoaxPress	Enum value CoaxPress.
HdSdi	Enum value HdSdi.
AllInterfaces	Enum value AllInterfaces.

### 5.1.3.2 FlushOperation

```
enum IpxCam::FlushOperation : uint32_t
```

An enum of Flush Operations. Flush Operations Types.

## Enumerator

Flush_OutputDiscard	Enum value Flush_OutputDiscard. Discards all buffers in the output queue and if necessary remove the entries from the event data queue.
Flush_AllToInput	Enum value Flush_AllToInput. Puts all buffers in the input pool. Even those in the output queue and discard entries in the event data queue.
Flush_UnqueuedToInput	Enum value Flush_UnqueuedToInput. Puts all buffers that are not in the input pool or the output queue in the input pool.
Flush_AllDiscard	Enum value Flush_AllDiscard. Discards all buffers in the input pool and output queue.

## 5.1.3.3 ServiceFileType

```
enum IpxCam::ServiceFileType : uint32_t
```

An enum of Service File Types. Service File Types.

## Enumerator

FileLUT	Enum value FileLUT.
FileDPC	Enum value FileDPC.
FileHPC	Enum value FileHPC.
FileFFC	Enum value FileFFC.

## 5.1.3.4 DeviceAccess

```
enum IpxCam::DeviceAccess : uint32_t
```

An enum of [Device](#) Access.

## Enumerator

ReadOnly	Enum value ReadOnly.
Control	Enum value Control.
Exclusive	Enum value Exclusive.

## 5.1.4 Function Documentation

## 5.1.4.1 IpxCam\_GetSystem()

```
IPX_CAMERA_API System * IpxCam::IpxCam_GetSystem ( )
```

This method returns the [System](#) module. It is the entry point to the GenTL Module hierarchy.

## Returns

returns the pointer to system... system is being created as soon as DLL is loaded

Here is the caller graph for this function:



## 5.2 IpxGenParam Namespace Reference

A namespace providing scope to the GenICam features using enum, classes, etc.

## Classes

- class [Array](#)  
*This class contains functions that can access each node from the camera descriptor XML file by type and name.*
- class [Boolean](#)  
*A class containing methods that map the integer element value of a GenICam IBoolean interface feature to true or false.*
- class [Category](#)  
*A class containing methods that the user can access the categories GenICam features. It will access the node object's of an ICategory interface. Each feature of a device will be placed in a [Category](#). The [Category](#) feature is used to present the user with a group of features for the named category.*
- class [Command](#)  
*A Class for GenICam [Command](#) properties contains methods that lets the user submit a command for execution as well as poll the status.*
- class [Enum](#)  
*Interface Class for Enumeration properties.*
- class [EnumEntry](#)  
*A Class for GenICam [Enum](#) Entries has methods to access the Enumeration node [Enum](#) Entry of the GenICam.*
- class [Float](#)  
*Interface Class for GenICam [Float](#) properties.*
- class [Int](#)  
*Interface Class for GenICam [Int](#) properties.*
- class [Param](#)  
*A Class for accessing the GenICam feature node parameters of the Camera Descriptor File.*
- class [ParamEventSink](#)  
*An Event Sink class designed to receive incoming events from Parameter Node Updates.*
- class [String](#)  
*Interface Class for GenICam [String](#) properties.*

## Enumerations

- enum [ParamType](#) : uint32\_t {  
[ParamUnknown](#), [ParamInt](#), [ParamFloat](#), [ParamString](#),  
[ParamEnum](#), [ParamEnumEntry](#), [ParamBoolean](#), [ParamCommand](#),  
[ParamCategory](#) }  
*An enum of Parameter Types. Parameter Node Types that can access the node object's programming interface.*
- enum [NameSpace](#) : uint32\_t { [NameSpaceStandard](#) = 0, [NameSpaceCustom](#), [NameSpaceUndefined](#) =999 }  
*An enum of GenICam NameSpace. Parameter Node Namespace.*
- enum [Visibility](#) : uint32\_t {  
[VisBeginner](#) = 0, [VisExpert](#), [VisGuru](#), [VisInvisible](#),  
[VisUndefined](#) = 99 }  
*An enum of Visibility. This element defines the type of user that has access to the feature.*

### 5.2.1 Detailed Description

A namespace providing scope to the GenICam features using enum, classes, etc.

A more detailed class description.

### 5.2.2 Enumeration Type Documentation

#### 5.2.2.1 ParamType

```
enum IpxGenParam::ParamType : uint32_t
```

An enum of Parameter Types. Parameter Node Types that can access the node object's programming interface.

#### Enumerator

<a href="#">ParamUnknown</a>	<a href="#">Enum</a> value <a href="#">ParamUnknown</a> . Unknown Parameter.
<a href="#">ParamInt</a>	<a href="#">Enum</a> value <a href="#">ParamInt</a> will access node object's of <a href="#">Integer</a> interface.
<a href="#">ParamFloat</a>	<a href="#">Enum</a> value <a href="#">ParamFloat</a> will access node object's of <a href="#">IFloat</a> interface.
<a href="#">ParamString</a>	<a href="#">Enum</a> value <a href="#">ParamString</a> will access node object's of <a href="#">IString</a> interface.
<a href="#">ParamEnum</a>	<a href="#">Enum</a> value <a href="#">ParamEnum</a> will access node object's of <a href="#">IEnumeration</a> interface.
<a href="#">ParamEnumEntry</a>	<a href="#">Enum</a> value <a href="#">ParamEnumEntry</a> will access the entry of <a href="#">Enum</a> parameter*/.
<a href="#">ParamBoolean</a>	<a href="#">Enum</a> value <a href="#">ParamBoolean</a> will access node object's of <a href="#">IBoolean</a> interface.
<a href="#">ParamCommand</a>	<a href="#">Enum</a> value <a href="#">ParamCommand</a> will access node object's of <a href="#">ICommand</a> interface.
<a href="#">ParamCategory</a>	<a href="#">Enum</a> value <a href="#">ParamCategory</a> will access node object's of <a href="#">ICategory</a> interface.

## 5.2.2.2 NameSpace

```
enum IpxGenParam::NameSpace : uint32_t
```

An enum of GenICam NameSpace. Parameter Node Namespace.

## Enumerator

NameSpaceStandard	<a href="#">Enum</a> value NameSpaceStandard. Identifies the standard namespace used in the file.
NameSpaceCustom	<a href="#">Enum</a> value NameSpaceCustom. Identifies the custom namespace used in the file.
NameSpaceUndefined	<a href="#">Enum</a> value NameSpaceUndefined. Unknown namespace.

## 5.2.2.3 Visibility

```
enum IpxGenParam::Visibility : uint32_t
```

An enum of Visibility. This element defines the type of user that has access to the feature.

## Enumerator

VisBeginner	<a href="#">Enum</a> value VisBeginner. User has visibility to all the basic features of the device.
VisExpert	<a href="#">Enum</a> value VisExpert. User has visibility to more advance features of the device.
VisGuru	<a href="#">Enum</a> value VisGuru. User has visibility to even more advance features that if set improperly can cause device to be in an improper state.
VisInvisible	<a href="#">Enum</a> value VisInvisible. Not visible.
VisUndefined	<a href="#">Enum</a> value VisUndefined. Unknown visibility.

## 5.3 IpxGui Namespace Reference

The IpxGUI namespace is a declarative region that provides a scope to the Imperx Camera GUI API classes and functions.

## Classes

- class [IpxGenParamTreeView](#)

*The [IpxGenParamTreeView](#) class is composed of functions to set and clear parameters of the node tree of the camera. The node tree can be set with the current parameters stored in the [IpxGenParam::Array](#) and [GenApi::INodeMap](#) class.*

## Enumerations

- enum [Visibility](#) : uint32\_t { [Beginner](#) = 0, [Expert](#), [Guru](#) }

*An enum of Visibility. Defines the visibility type of features that user will see in the Tree View.*

## Functions

- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API [IlpxGenParamTreeView](#) \* [CreateGenParamTreeViewForArrayA](#) ([IpXGenParam::Array](#) \*genParam, const char \*title, uintptr\_t parentWindow=0)  
*This method returns the pointer to the [IlpxGenParamTreeView](#) class that was created using information extracted from the [IpXGenParam::Array](#) class.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API [IlpxGenParamTreeView](#) \* [CreateGenParamTreeViewForArrayW](#) ([IpXGenParam::Array](#) \*genParam, const wchar\_t \*title, uintptr\_t parentWindow=0)  
*This method returns the pointer to the [IlpxGenParamTreeView](#) class that was created using information extracted from the [IpXGenParam::Array](#).*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API [IlpxGenParamTreeView](#) \* [CreateGenParamTreeViewForNodemapA](#) (IPX\_GENAPI\_NS::INodeMap \*nodemap, const char \*title, uintptr\_t parentWindow=0)  
*This method returns the pointer to the [IlpxGenParamTreeView](#) class that was created using information extracted from the [GenApi::INodeMap](#) class.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API [IlpxGenParamTreeView](#) \* [CreateGenParamTreeViewForNodemapW](#) (IPX\_GENAPI\_NS::INodeMap \*nodemap, const wchar\_t \*title, uintptr\_t parentWindow=0)  
*This method returns the pointer to the [IlpxGenParamTreeView](#) that was created using information extracted from the [GenApi::INodeMap](#) class.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [DestroyGenParamTreeView](#) ([IlpxGenParamTreeView](#) \*view)  
*This method destroys the [IlpxGenParamTreeView](#) object previously created.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API [IpXCam::DeviceInfo](#) \* [SelectCameraA](#) ([IpXCam::System](#) \*pSystem, const char \*title, uintptr\_t parentWindow=0, bool poll=true)  
*This method pops-up the "Select camera" modal dialog, where user can select the Camera and obtain the pointer to [IpXCam::DeviceInfo](#) object for the selected camera.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API [IpXCam::DeviceInfo](#) \* [SelectCameraW](#) ([IpXCam::System](#) \*pSystem, const wchar\_t \*title, uintptr\_t parentWindow=0, bool poll=true)  
*This method pops-up the "Select camera" modal dialog, where user can select the Camera and obtain the pointer to [IpXCam::DeviceInfo](#) object for the selected camera.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [ShowCamConfigDialog](#) ([IpXCam::Device](#) \*device, uintptr\_t parentWindow=0)  
*Show Camera Configuration Dialog.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [ShowFrameABDialog](#) ([IpXCam::Device](#) \*device, uintptr\_t parentWindow=0)
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [ShowTriggerDialog](#) ([IpXCam::Device](#) \*device, uintptr\_t parentWindow=0)  
*Show Trigger Dialog.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [ShowPulseDialog](#) ([IpXCam::Device](#) \*device, uintptr\_t parentWindow=0)  
*Show Pulse Dialog.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [ShowStrobeDialog](#) ([IpXCam::Device](#) \*device, uintptr\_t parentWindow=0)  
*Show Strobe Dialog.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [ShowOutputDialog](#) ([IpXCam::Device](#) \*device, uintptr\_t parentWindow=0)  
*Show Output Data Dialog.*
- IPXCAM\_GUI\_EXTERN\_C IPX\_CAMERA\_GUI\_API void [ShowColorDialog](#) ([IpXCam::Device](#) \*device, uintptr\_t parentWindow=0)  
*Show Color Dialog.*

### 5.3.1 Detailed Description

The IpxGUI namespace is a declarative region that provides a scope to the Imperx Camera GUI API classes and functions.

The IpxGUI namespace includes Imperx Camera GUI API classes and functions, such as: [IpxGenParamTreeView](#), [SelectCameraA](#) [SelectCameraW](#) [IpxGenParamTreeView](#), [IpxCameraSelectorDialog](#)

### 5.3.2 Enumeration Type Documentation

#### 5.3.2.1 Visibility

```
enum IpxGui::Visibility : uint32_t
```

An enum of Visibility. Defines the visibility type of features that user will see in the Tree View.

##### Enumerator

Beginner	Enum value Beginner. User has visibility to all the basic features of the device.
Expert	Enum value Expert. User has visibility to more advance features of the device.
Guru	Enum value Guru. User has visibility to even more advance features that if set improperly can cause device to be in an improper state.

### 5.3.3 Function Documentation

#### 5.3.3.1 CreateGenParamTreeViewForArrayA()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IpxGenParamTreeView* IpxGui::CreateGenParamTreeViewForArrayA (
    IpxGenParam::Array * genParam,
    const char * title,
    uintptr_t parentWindow = 0 )
```

This method returns the pointer to the [IpxGenParamTreeView](#) class that was created using information extracted from the [IpxGenParam::Array](#) class.

##### Parameters

in	<i>genParam</i>	The pointer to the <a href="#">IpxGenParam::Array</a> class.
in	<i>title</i>	The title of the <a href="#">IpxGenParamTreeView</a> class as a const char.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

**Returns**

If the method succeeds, the return value is the pointer to the [IpxGenParamTreeView](#) class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:

**5.3.3.2 CreateGenParamTreeViewForArrayW()**

```

IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IpxGenParamTreeView* IpxGui::CreateGenParamTreeViewFor↵
ArrayW (
    IpxGenParam::Array * genParam,
    const wchar_t * title,
    uintptr_t parentWindow = 0 )
  
```

This method returns the pointer to the [IpxGenParamTreeView](#) class that was created using information extracted from the [IpxGenParam::Array](#).

**Parameters**

in	<i>genParam</i>	The pointer to the <a href="#">IpxGenParam::Array</a> class.
in	<i>title</i>	The title of the <a href="#">IpxGenParamTreeView</a> class as a wchar_t variable.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget



**Returns**

If the method succeeds, the return value is the pointer to the [IpxGenParamTreeView](#) class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:

**5.3.3.3 CreateGenParamTreeViewForNodemapA()**

```

IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IpxGenParamTreeView* IpxGui::CreateGenParamTreeViewFor↵
NodemapA (
    IPX_GENAPI_NS::INodeMap * nodemap,
    const char * title,
    uintptr_t parentWindow = 0 )
  
```

This method returns the pointer to the [IpxGenParamTreeView](#) class that was created using information extracted from the `GenApi::INodeMap` class.

**Parameters**

in	<i>nodemap</i>	The pointer to the <code>GenApi::INodeMap</code> class.
in	<i>title</i>	The title of the <a href="#">IpxGenParamTreeView</a> class as a <code>wchar_t</code> variable.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

**Returns**

If the method succeeds, the return value is the pointer to the [IpxGenParamTreeView](#) class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:

**5.3.3.4 CreateGenParamTreeViewForNodemapW()**

```

IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IpxGenParamTreeView\* IpxGui::CreateGenParamTreeViewFor↵
NodemapW (
    IPX_GENAPI_NS::INodeMap * nodemap,
    const wchar_t * title,
    uintptr_t parentWindow = 0 )
  
```

This method returns the pointer to the [IpxGenParamTreeView](#) that was created using information extracted from the `GenApi::INodeMap` class.

**Parameters**

in	<i>nodemap</i>	The pointer to the <code>GenApi::INodeMap</code> class.
in	<i>title</i>	The title of the <a href="#">IpxGenParamTreeView</a> as a <code>wchar_t</code> variable.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

**Returns**

If the method succeeds, the return value is the pointer to the [IpxGenParamTreeView](#) class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:

**5.3.3.5 DestroyGenParamTreeView()**

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::DestroyGenParamTreeView (  
    IpxGenParamTreeView * view )
```

This method destroys the [IpxGenParamTreeView](#) object previously created.

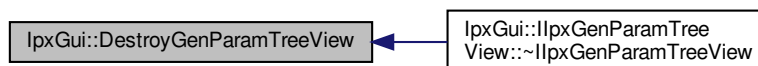
**Parameters**

in	<i>view</i>	A pointer to the <a href="#">IpxGenParamTreeView</a> class.
----	-------------	---

**Returns**

Void.

Here is the caller graph for this function:



### 5.3.3.6 SelectCameraA()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IpxCam::DeviceInfo* IpxGui::SelectCameraA (
    IpxCam::System * pSystem,
    const char * title,
    uintptr_t parentWindow = 0,
    bool poll = true )
```

This method pops-up the "Select camera" modal dialog, where user can select the Camera and obtain the pointer to [IpxCam::DeviceInfo](#) object for the selected camera.

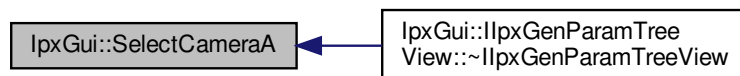
#### Parameters

in	<i>pSystem</i>	The pointer to the <a href="#">IpxCam::System</a> class.
in	<i>title</i>	The title of the selected Camera as a const char variable.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget
in	<i>poll</i>	Specifies if poll check box should be checked by default, so the System will be polled for new devices to appear

#### Returns

If the method succeeds, the return value is the pointer to the [IpxCam::DeviceInfo](#) class. Otherwise, the return value is nullptr.

Here is the caller graph for this function:



### 5.3.3.7 SelectCameraW()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IpxCam::DeviceInfo* IpxGui::SelectCameraW (
    IpxCam::System * pSystem,
    const wchar_t * title,
    uintptr_t parentWindow = 0,
    bool poll = true )
```

This method pops-up the "Select camera" modal dialog, where user can select the Camera and obtain the pointer to [IpxCam::DeviceInfo](#) object for the selected camera.

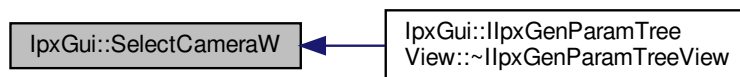
## Parameters

in	<i>pSystem</i>	The pointer to the <a href="#">IpxCam::System</a> class.
in	<i>title</i>	The title of the <a href="#">IpxGenParamTreeView</a> as a <code>wchar_t</code> variable.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget
in	<i>poll</i>	Specifies if poll check box should be checked by default, so the System will be polled for new devices to appear

## Returns

If the method succeeds, the return value is the pointer to the [IpxCam::DeviceInfo](#) class. Otherwise, the return value is `nullptr`.

Here is the caller graph for this function:



## 5.3.3.8 ShowCamConfigDialog()

```

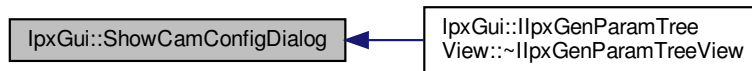
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowCamConfigDialog (
    IpxCam::Device * device,
    uintptr_t parentWindow = 0 )
  
```

Show Camera Configuration Dialog.

## Parameters

in	<i>device</i>	The pointer to the <a href="#">IpxCam::Device</a> class.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

Here is the caller graph for this function:



#### 5.3.3.9 ShowFrameABDialog()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowFrameABDialog (
    IpxCam::Device * device,
    uintptr_t parentWindow = 0 )
```

##### Parameters

in	<i>device</i>	The pointer to the <a href="#">IpxCam::Device</a> class.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

Here is the caller graph for this function:



#### 5.3.3.10 ShowTriggerDialog()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowTriggerDialog (
    IpxCam::Device * device,
    uintptr_t parentWindow = 0 )
```

Show Trigger Dialog.

## Parameters

in	<i>device</i>	The pointer to the <a href="#">IpxCam::Device</a> class.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

Here is the caller graph for this function:



## 5.3.3.11 ShowPulseDialog()

```

IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowPulseDialog (
    IpxCam::Device * device,
    uintptr_t parentWindow = 0 )
  
```

Show Pulse Dialog.

## Parameters

in	<i>device</i>	The pointer to the <a href="#">IpxCam::Device</a> class.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

Here is the caller graph for this function:



### 5.3.3.12 ShowStrobeDialog()

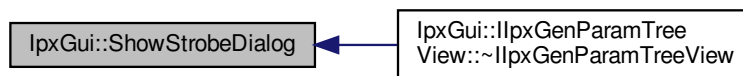
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowStrobeDialog (
    IpxCam::Device * device,
    uintptr_t parentWindow = 0 )
```

Show Strobe Dialog.

#### Parameters

in	<i>device</i>	The pointer to the <a href="#">IpxCam::Device</a> class.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

Here is the caller graph for this function:



### 5.3.3.13 ShowOutputDialog()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowOutputDialog (
    IpxCam::Device * device,
    uintptr_t parentWindow = 0 )
```

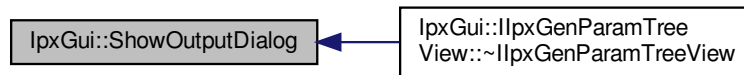
Show Output Data Dialog.

#### Parameters

in	<i>device</i>	The pointer to the <a href="#">IpxCam::Device</a> class.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget



Here is the caller graph for this function:



#### 5.3.3.14 ShowColorDialog()

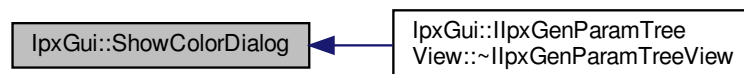
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowColorDialog (
    IpxCam::Device * device,
    uintptr_t parentWindow = 0 )
```

Show Color Dialog.

##### Parameters

in	<i>device</i>	The pointer to the <a href="#">IpxCam::Device</a> class.
in	<i>parentWindow</i>	A pointer to the parent Window. If the parent is set to 0, this widget is setup to become a window. If not this widget becomes a child widget

Here is the caller graph for this function:





## Chapter 6

# Class Documentation

### 6.1 IpxGenParam::Array Class Reference

This class contains functions that can access each node from the camera descriptor XML file by type and name.

```
#include <IpxCameraApi.h>
```

#### Public Member Functions

- virtual [~Array](#) ()  
*A destructor of the [Array](#) class.*
- virtual [Param](#) \* [GetParam](#) (const char \*name, IpxCamErr \*err)=0  
*This method gets the pointer to the [Param](#) class for the specified node name from the node map declared in the camera descriptor XML file.*
- virtual [Boolean](#) \* [GetBoolean](#) (const char \*name, IpxCamErr \*err)=0  
*This method gets the pointer to the [Boolean](#) class for the specified node name of the camera descriptor XML file.*
- virtual [Command](#) \* [GetCommand](#) (const char \*name, IpxCamErr \*err)=0  
*This method gets the pointer to the [Command](#) class for the specified node name of the camera descriptor XML file.*
- virtual [Enum](#) \* [GetEnum](#) (const char \*name, IpxCamErr \*err)=0  
*This method gets the pointer to the [Enum](#) class for the specified node name of the camera descriptor XML file.*
- virtual [Float](#) \* [GetFloat](#) (const char \*name, IpxCamErr \*err)=0  
*This method gets the pointer to the [Float](#) class for the specified node name of the camera descriptor XML file.*
- virtual [Int](#) \* [GetInt](#) (const char \*name, IpxCamErr \*err)=0  
*This method gets the pointer to the [Int](#) class for the specified node name of the camera descriptor XML file.*
- virtual [String](#) \* [GetString](#) (const char \*name, IpxCamErr \*err)=0  
*This method gets the pointer to the [String](#) class for the specified node name of the camera descriptor XML file.*
- virtual [Category](#) \* [GetRootCategory](#) (IpxCamErr \*err)=0  
*This method gets the pointer to the root node class. The Root node is considered a special node. It has no parent node. In the topology graph, it is the top node which connects to at least one child node. The child node may connect to the device node that provides the connection to the transport layer.*
- virtual IPX\_GENAPI\_NS::INodeMap \* [GetNodeMap](#) (IpxCamErr \*err)=0

*This method gets the pointer to the NodeMap interface. The NodeMap interface will provide methods to retrieves all nodes in the node map.*

- virtual uint32\_t [GetCount](#) ()=0

*This method gets the number of nodes.*

- virtual [Param](#) \* [GetParamByIndex](#) (uint32\_t idx, lpxCamErr \*err)=0

*This method gets the parameter by index.*

- virtual lpxCamErr [SetBooleanValue](#) (const char \*name, bool aValue)=0

*This method sets the [Boolean](#) value of the [Boolean](#) node.*

- virtual bool [GetBooleanValue](#) (const char \*name, lpxCamErr \*err=nullptr)=0

*This method gets the [Boolean](#) value of the [Boolean](#) node.*

- virtual lpxCamErr [SetEnumValueStr](#) (const char \*name, const char \*val)=0

*This method sets the [Enum](#) node maps and the [Enum](#) interface to a name and index value. Each of the enum entries are represented by a name and index pair. This method sets the [Enum](#) value [String](#) of the corresponding node. The enum nodes map to a drop down box.*

- virtual lpxCamErr [SetEnumValue](#) (const char \*name, int64\_t val)=0

*This method sets the [Enum](#) value of the enum node.*

- virtual const char \* [GetEnumValueStr](#) (const char \*name, lpxCamErr \*err=nullptr)=0

*This method gets the [Enum](#) value string of the current set [Enum](#) value entry.*

- virtual int64\_t [GetEnumValue](#) (const char \*name, lpxCamErr \*err=nullptr)=0

*This method gets the [Enum](#) value of the [Enum](#) node.*

- virtual lpxCamErr [SetFloatValue](#) (const char \*name, double val)=0

*This method sets the [Float](#) value of the [Float](#) node.*

- virtual double [GetFloatValue](#) (const char \*name, lpxCamErr \*err=nullptr)=0

*This method gets the [Float](#) value of the [Float](#) node.*

- virtual lpxCamErr [SetIntegerValue](#) (const char \*name, int64\_t val)=0

*This method sets the [Integer](#) value of the [Integer](#) node.*

- virtual int64\_t [GetIntegerValue](#) (const char \*name, lpxCamErr \*err=nullptr)=0

*This method gets the [Integer](#) value of the [Integer](#) node.*

- virtual lpxCamErr [SetStringValue](#) (const char \*name, const char \*val)=0

*This method sets the [String](#) value of the [String](#) node.*

- virtual const char \* [GetStringValue](#) (const char \*name, lpxCamErr \*err=nullptr)=0

*This method gets the [String](#) value of the [String](#) node.*

- virtual lpxCamErr [ExecuteCommand](#) (const char \*name)=0

*This method executes/submits the command.*

- virtual bool [IsCommandDone](#) (const char \*name, lpxCamErr \*err=nullptr)=0

*This method polls the corresponding executed command to see if the executed command is done or not.*

- virtual lpxCamErr [Poll](#) (int64\_t elapsedTime)=0

*This method fires nodes which have a polling time.*

### 6.1.1 Detailed Description

This class contains functions that can access each node from the camera descriptor XML file by type and name.

An [Array](#) class.

## 6.1.2 Constructor & Destructor Documentation

### 6.1.2.1 ~Array()

```
virtual IpxGenParam::Array::~~Array ( ) [inline], [virtual]
```

A destructor of the [Array](#) class.

Destructor. Destroys the [Array](#) and all its descendants.

## 6.1.3 Member Function Documentation

### 6.1.3.1 GetParam()

```
virtual Param* IpxGenParam::Array::GetParam (
    const char * name,
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the [Param](#) class for the specified node name from the node map declared in the camera descriptor XML file.

#### Parameters

in	<i>name</i>	Unique name of a node in node map.
out	<i>err</i>	Returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">Param</a> class of the specified node name</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM</code> - specified node name not found in camera descriptor XML file</li> </ul>

#### Returns

If the method succeeds, it returns the pointer to the [Param](#) class for the specific node name. Otherwise, it returns a nullptr.

### 6.1.3.2 GetBoolean()

```
virtual Boolean* IpxGenParam::Array::GetBoolean (
    const char * name,
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the [Boolean](#) class for the specified node name of the camera descriptor XML file.

#### Parameters

in	<i>name</i>	A unique name of <a href="#">Boolean</a> type node in the camera descriptor XML file.
out	<i>err</i>	Returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">Boolean</a> class of the specified node name</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - specified node name not found in camera descriptor XML file</li> </ul>

#### Returns

If the method succeeds, it returns the pointer to the [Boolean](#) class for the specific node name. Otherwise, it returns a nullptr.

### 6.1.3.3 GetCommand()

```
virtual Command* IpxGenParam::Array::GetCommand (
    const char * name,
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the [Command](#) class for the specified node name of the camera descriptor XML file.

#### Parameters

in	<i>name</i>	Unique name of <a href="#">Command</a> type node in the camera descriptor XML file.
out	<i>err</i>	returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">Command</a> class of the specified node name</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - specified node name not found in camera descriptor XML file</li> </ul>

**Returns**

If method succeeds, it returns the pointer to the [Command](#) class for the specific node name. Otherwise, it returns a nullptr.

**6.1.3.4 GetEnum()**

```
virtual Enum* IpxGenParam::Array::GetEnum (
    const char * name,
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the [Enum](#) class for the specified node name of the camera descriptor XML file.

**Parameters**

in	<i>name</i>	Unique name of Enumeration type node in the camera descriptor XML file.
out	<i>err</i>	returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">Enum</a> class of the specified node name</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - specified node name not found in camera descriptor XML file</li> </ul>

**Returns**

If the method succeeds, it returns the pointer to the [Enum](#) parameter class for the specific node name. Otherwise, it returns a nullptr.

**6.1.3.5 GetFloat()**

```
virtual Float* IpxGenParam::Array::GetFloat (
    const char * name,
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the [Float](#) class for the specified node name of the camera descriptor XML file.

**Parameters**

in	<i>name</i>	Unique name of <a href="#">Float</a> type node in the camera descriptor XML file.
out	<i>err</i>	returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">Float</a> class of the specified node name</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - specified node name not found in camera descriptor XML file</li> </ul>
Generated by Doxygen		

**Returns**

If the method succeeds, it returns the pointer to the [Float](#) parameter class for the specific node name

**6.1.3.6 GetInt()**

```
virtual Int* IpxGenParam::Array::GetInt (
    const char * name,
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the [Int](#) class for the specified node name of the camera descriptor XML file.

**Parameters**

in	<i>name</i>	Unique name of Integer type node in the camera descriptor XML file.
out	<i>err</i>	returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">Int</a> class of the specified node name</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - specified node name not found in camera descriptor XML file</li> </ul>

**Returns**

If the method succeeds, it returns the pointer to the [Int](#) class for the specific node name

**6.1.3.7 GetString()**

```
virtual String* IpxGenParam::Array::GetString (
    const char * name,
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the [String](#) class for the specified node name of the camera descriptor XML file.

**Parameters**

in	<i>name</i>	Unique name of <a href="#">String</a> type node in the camera descriptor XML file.
out	<i>err</i>	returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">String</a> class of the specified node name</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - specified node name not found in camera descriptor XML file</li> </ul>



**Returns**

If the method succeeds, it returns the pointer to the [String](#) class for the specific node name

**6.1.3.8 GetRootCategory()**

```
virtual Category* IpxGenParam::Array::GetRootCategory (
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the root node class. The Root node is considered a special node. It has no parent node. In the topology graph, it is the top node which connects to at least one child node. The child node may connect to the device node that provides the connection to the transport layer.

**Parameters**

out	err	returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <a href="#">Category</a> class</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - specified Root node name not found in camera descriptor XML file</li> </ul>
-----	-----	---

**Returns**

returns the pointer to the [Category](#) (root node) class

**6.1.3.9 GetNodeMap()**

```
virtual IPX_GENAPI_NS::INodeMap* IpxGenParam::Array::GetNodeMap (
    IpxCamErr * err ) [pure virtual]
```

This method gets the pointer to the NodeMap interface. The NodeMap interface will provide methods to retrieves all nodes in the node map.

**Parameters**

out	err	returns an error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to <code>GenApi::INodeMap</code> class</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - the node map does not exist</li> </ul>
-----	-----	---

**Returns**

nodemap returns the pointer to the NodeMap interface

**6.1.3.10 GetCount()**

```
virtual uint32_t IpxGenParam::Array::GetCount ( ) [pure virtual]
```

This method gets the number of nodes.

**Returns**

The number of nodes. This number should be greater than 0.

**6.1.3.11 GetParamByIndex()**

```
virtual Param* IpxGenParam::Array::GetParamByIndex (
    uint32_t idx,
    IpxCamErr * err ) [pure virtual]
```

This method gets the parameter by index.

**Parameters**

in	idx	Index
out	err	returns the error code: <ul style="list-style-type: none"> <li>• IpxCamErr::IPX_CAM_ERR_OK - Successfully returns pointer to <a href="#">Param</a> class</li> <li>• IpxCamErr::IPX_CAM_ERR_INVALID_INDEX - entered invalid index</li> </ul>

**Returns**

Returns param pointer to Parameter class of the specified node referenced by the index value

**6.1.3.12 SetBooleanValue()**

```
virtual IpxCamErr IpxGenParam::Array::SetBooleanValue (
    const char * name,
    bool aValue ) [pure virtual]
```

This method sets the [Boolean](#) value of the [Boolean](#) node.

## Parameters

in	<i>name</i>	Unique name of <a href="#">Boolean</a> node to set
in	<i>aValue</i>	<a href="#">Boolean</a> value to set

## Returns

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully set the [Boolean](#) value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node

6.1.3.13 `GetBooleanValue()`

```
virtual bool IpxGenParam::Array::GetBooleanValue (
    const char * name,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [Boolean](#) value of the [Boolean](#) node.

## Parameters

in	<i>name</i>	Unique name of <a href="#">Boolean</a> node to get
out	<i>err</i>	returns the error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the <a href="#">Boolean</a> value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM</code> - unknown parameter</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node</li> </ul>

## Returns

Returns the [Boolean](#) Value

6.1.3.14 `SetEnumValueStr()`

```
virtual IpxCamErr IpxGenParam::Array::SetEnumValueStr (
    const char * name,
    const char * val ) [pure virtual]
```

This method sets the [Enum](#) node maps and the [Enum](#) interface to a name and index value. Each of the enum entries are represented by a name and index pair. This method sets the [Enum](#) value [String](#) of the corresponding node. The enum nodes map to a drop down box.

**Parameters**

in	<i>name</i>	Name of <a href="#">Enum</a> entry node to set
in	<i>val</i>	<a href="#">Enum</a> node value set

**Returns**

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the [Enum](#) Value string
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the const value entered is out of range

**6.1.3.15 SetEnumValue()**

```
virtual IpxCamErr IpxGenParam::Array::SetEnumValue (  
    const char * name,  
    int64_t val ) [pure virtual]
```

This method sets the [Enum](#) value of the enum node.

**Parameters**

in	<i>name</i>	Unique ame of <a href="#">Enum</a> entry to set
in	<i>val</i>	<a href="#">Enum</a> entry value to set

**Returns**

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the [Enum](#) value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the value entered is out of range

**6.1.3.16 GetEnumValueStr()**

```
virtual const char* IpxGenParam::Array::GetEnumValueStr (  
    const char * name,  
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [Enum](#) value string of the current set [Enum](#) value entry.

## Parameters

in	<i>name</i>	Unique name of <a href="#">Enum</a> entry
out	<i>err</i>	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the <a href="#">Enum</a> string value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM</code> - unknown parameter</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>

## Returns

Get the [Enum](#) value [String](#) of the current set [Enum](#) Value Entry

## 6.1.3.17 GetEnumValue()

```
virtual int64_t IpxGenParam::Array::GetEnumValue (
    const char * name,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [Enum](#) value of the [Enum](#) node.

## Parameters

in	<i>name</i>	Unique name of <a href="#">Enum</a> type node in the camera descriptor XML file.
out	<i>err</i>	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the <a href="#">Enum</a> value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM</code> - unknown parameter</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>

## Returns

Returns the [Enum](#) Value

## 6.1.3.18 SetFloatValue()

```
virtual IpxCamErr IpxGenParam::Array::SetFloatValue (
    const char * name,
    double val ) [pure virtual]
```

This method sets the [Float](#) value of the [Float](#) node.

#### Parameters

in	<i>name</i>	Unique name of <a href="#">Float</a> type node in the camera descriptor XML file.
in	<i>val</i>	<a href="#">Float</a> value to set

#### Returns

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the [Float](#) value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the value entered is out of range

#### 6.1.3.19 GetFloatValue()

```
virtual double IpxGenParam::Array::GetFloatValue (
    const char * name,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [Float](#) value of the [Float](#) node.

#### Parameters

in	<i>name</i>	Unique name of <a href="#">Float</a> type node in the camera descriptor XML file.
out	<i>err</i>	returns the error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the <a href="#">Float</a> value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM</code> - unknown parameter</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>

#### Returns

Returns the [Float](#) value

## 6.1.3.20 SetIntegerValue()

```
virtual IpxCamErr IpxGenParam::Array::SetIntegerValue (
    const char * name,
    int64_t val ) [pure virtual]
```

This method sets the Integer value of the Integer node.

## Parameters

in	<i>name</i>	Unique name of Integer type node in the camera descriptor XML file.
in	<i>val</i>	Integer value to set

## Returns

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the Integer value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the value entered is out of range

## 6.1.3.21 GetIntegerValue()

```
virtual int64_t IpxGenParam::Array::GetIntegerValue (
    const char * name,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the Integer value of the Integer node.

## Parameters

in	<i>name</i>	Unique name of Integer type node in the camera descriptor XML file.
out	<i>err</i>	<p>returns the error code:</p> <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the Integer value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM</code> - unknown parameter</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>

**Returns**

Returns the Integer value

**6.1.3.22 SetStringValue()**

```
virtual IpxCamErr IpxGenParam::Array::SetStringValue (
    const char * name,
    const char * val ) [pure virtual]
```

This method sets the [String](#) value of the [String](#) node.

**Parameters**

in	<i>name</i>	Unique name of <a href="#">String</a> type node in the camera descriptor XML file.
in	<i>val</i>	<a href="#">String</a> value to set

**Returns**

Returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the [String](#) value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the value entered is out of range

**6.1.3.23 GetStringValue()**

```
virtual const char* IpxGenParam::Array::GetStringValue (
    const char * name,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [String](#) value of the [String](#) node.

**Parameters**

in	<i>name</i>	Unique name of <a href="#">String</a> type node in the camera descriptor XML file.
out	<i>err</i>	returns the error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the <a href="#">String</a> value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM</code> - unknown parameter</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
		Generated by Doxygen



**Returns**

Returns the [String](#) value

**6.1.3.24 ExecuteCommand()**

```
virtual IpxCamErr IpxGenParam::Array::ExecuteCommand (
    const char * name ) [pure virtual]
```

This method executes/submits the command.

**Parameters**

in	<i>name</i>	Unique name of <a href="#">Command</a> type node in the camera descriptor XML file.
----	-------------	---

**Returns**

Returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully determines state of executed command.
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node

**6.1.3.25 IsCommandDone()**

```
virtual bool IpxGenParam::Array::IsCommandDone (
    const char * name,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method polls the corresponding executed command to see if the executed command is done or not.

**Parameters**

in	<i>name</i>	Unique name of <a href="#">Command</a> type node in the camera descriptor XML file.
out	<i>err</i>	returns the error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully determines state of executed command.</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> </ul>

**Returns**

Returns true if the Execute command has finished. Otherwise, returns false.

### 6.1.3.26 Poll()

```
virtual IpxCamErr IpxGenParam::Array::Poll (
    int64_t elapsedTime ) [pure virtual]
```

This method fires nodes which have a polling time.

#### Parameters

in	<i>elapsedTime</i>	Time elapsed since last poll in msec
----	--------------------	--------------------------------------

#### Returns

Returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully determines state of executed command.
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node

The documentation for this class was generated from the following file:

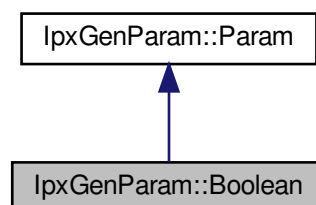
- `IpxCameraApi.h`

## 6.2 IpxGenParam::Boolean Class Reference

A class containing methods that map the integer element value of a GenICam IBoolean interface feature to true or false.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for `IpxGenParam::Boolean`:



## Public Member Functions

- virtual [ParamType](#) [GetType](#) ()  
*This method returns the node object [Boolean](#) type.*
- virtual IpxCamErr [SetValue](#) (bool val)=0  
*This method can be used to set the node value to true or false.*
- virtual bool [GetValue](#) (IpxCamErr \*err=nullptr)=0  
*This method returns the node value. It can return a true or false value.*

### 6.2.1 Detailed Description

A class containing methods that map the integer element value of a GenICam IBoolean interface feature to true or false.

A class containing methods for GenICam [Boolean](#) Properties. For example, the mapping below will illustrate the I↔ Boolean interfaces of a **LUTEnable** feature.

### 6.2.2 Member Function Documentation

#### 6.2.2.1 GetType()

```
virtual ParamType IpxGenParam::Boolean::GetType ( ) [inline], [virtual]
```

This method returns the node object [Boolean](#) type.

#### Returns

Returns the node object [Boolean](#) type

Implements [IpxGenParam::Param](#).

#### 6.2.2.2 SetValue()

```
virtual IpxCamErr IpxGenParam::Boolean::SetValue (
    bool val ) [pure virtual]
```

This method can be used to set the node value to true or false.

#### Parameters

in	val	The node value to set such as true or false
----	-----	---

**Returns**

Returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the [Boolean](#) value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the value entered is out of range

**6.2.2.3 GetValue()**

```
virtual bool IpxGenParam::Boolean::GetValue (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method returns the node value. It can return a true or false value.

**Parameters**

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the value of the <a href="#">Boolean</a> node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	---

**Returns**

The node value read.

The documentation for this class was generated from the following file:

- `IpxCameraApi.h`

**6.3 IpxCam::Buffer Class Reference**

The [Buffer](#) class represents the buffer module in the GenTL module hierarchy.

```
#include <IpxCameraApi.h>
```

## Public Member Functions

- virtual [~Buffer](#) ()  
*A destructor of the [Buffer](#) object.*
- virtual [IpxImage](#) \* [GetImage](#) ()=0  
*This method returns the pointer to the memory of the image object.*
- virtual void \* [GetBufferPtr](#) ()=0  
*This method returns the pointer to the memory of the buffer object.*
- virtual [size\\_t](#) [GetImageOffset](#) ()=0  
*This method returns the image offset of the image.*
- virtual [size\\_t](#) [GetBufferSize](#) ()=0  
*This method returns the buffer size of the allocated memory.*
- virtual [uint64\\_t](#) [GetPixelFormat](#) ()=0  
*This method returns the pixel format of the buffer object.*
- virtual void \* [GetUserPtr](#) ()=0  
*This method returns the user data buffer pointer to memory of the buffer object.*
- virtual [uint64\\_t](#) [GetTimestamp](#) ()=0  
*This method returns the timestamp of the buffer that was acquired.*
- virtual [uint64\\_t](#) [GetFrameID](#) ()=0  
*This method returns the frame id of the image stream block id of the buffer object.*
- virtual bool [IsIncomplete](#) ()=0  
*This method returns a flag indicating if the buffer data has been transferred or an incomplete transfer.*
- virtual [size\\_t](#) [GetWidth](#) ()=0  
*This method gets the width of the buffer data in number of pixels.*
- virtual [size\\_t](#) [GetHeight](#) ()=0  
*This method gets the height of the data buffer.*
- virtual [size\\_t](#) [GetXOffset](#) ()=0  
*This method returns the XOffset of the data in the buffer.*
- virtual [size\\_t](#) [GetYOffset](#) ()=0  
*This method returns the YOffset of the data in the buffer.*
- virtual [size\\_t](#) [GetXPadding](#) ()=0  
*This method returns the number of extra bytes padded in the X direction.*
- virtual [size\\_t](#) [GetYPadding](#) ()=0  
*This method returns the number of extra bytes padded in the Y direction.*
- virtual [size\\_t](#) [GetDeliveredHeight](#) ()=0  
*This method returns the actual height of delivered data.*
- virtual bool [IsKacFrameB](#) ()=0  
*This method indicates if this buffer is Frame A or Frame B, acquired from Cheetah camera with KAC-12040 or KAC-06040 CMOS sensor.*

### 6.3.1 Detailed Description

The [Buffer](#) class represents the buffer module in the GenTL module hierarchy.

This [Buffer](#) class contains a method that can be used to temporarily hold a chunk of memory from the acquisition engine. It also contains methods that can retrieve information of the received data (held data) such as pointer to an image, image size, and pixel format.

## 6.3.2 Constructor & Destructor Documentation

### 6.3.2.1 ~Buffer()

```
virtual IpxCam::Buffer::~~Buffer ( ) [inline], [virtual]
```

A destructor of the [Buffer](#) object.

Destructor. Destroys the [Buffer](#) object and all its descendants.

## 6.3.3 Member Function Documentation

### 6.3.3.1 GetImage()

```
virtual IpxImage* IpxCam::Buffer::GetImage ( ) [pure virtual]
```

This method returns the pointer to the memory of the image object.

#### Returns

returns the pointer to the memory of the image object

### 6.3.3.2 GetBufferPtr()

```
virtual void* IpxCam::Buffer::GetBufferPtr ( ) [pure virtual]
```

This method returns the pointer to the memory of the buffer object.

#### Returns

returns the pointer to the memory of [Buffer](#) Object

#### 6.3.3.3 GetImageOffset()

```
virtual size_t IpxCam::Buffer::GetImageOffset ( ) [pure virtual]
```

This method returns the image offset of the image.

##### Returns

returns the Image Offset of the Image

#### 6.3.3.4 GetBufferSize()

```
virtual size_t IpxCam::Buffer::GetBufferSize ( ) [pure virtual]
```

This method returns the buffer size of the allocated memory.

##### Returns

returns the number of bytes written into the buffers

#### 6.3.3.5 GetPixelFormat()

```
virtual uint64_t IpxCam::Buffer::GetPixelFormat ( ) [pure virtual]
```

This method returns the pixel format of the buffer object.

##### Returns

returns the Pixel Format of the [Buffer](#) Object

#### 6.3.3.6 GetUserPtr()

```
virtual void* IpxCam::Buffer::GetUserPtr ( ) [pure virtual]
```

This method returns the user data buffer pointer to memory of the buffer object.

##### Returns

returns the user data buffer pointer to memory of the [Buffer](#) Object

#### 6.3.3.7 GetTimestamp()

```
virtual uint64_t IpxCam::Buffer::GetTimestamp ( ) [pure virtual]
```

This method returns the timestamp of the buffer that was acquired.

##### Returns

returns the timestamp of the buffer that was acquired

#### 6.3.3.8 GetFrameID()

```
virtual uint64_t IpxCam::Buffer::GetFrameID ( ) [pure virtual]
```

This method returns the frame id of the image stream block id of the buffer object.

##### Returns

returns the image stream block id of the buffer object

#### 6.3.3.9 IsIncomplete()

```
virtual bool IpxCam::Buffer::IsIncomplete ( ) [pure virtual]
```

This method returns a flag indicating if the buffer data has been transferred or an incomplete transfer.

##### Returns

Is true, if an error occurred during the transferring of the buffer data

#### 6.3.3.10 GetWidth()

```
virtual size_t IpxCam::Buffer::GetWidth ( ) [pure virtual]
```

This method gets the width of the buffer data in number of pixels.

##### Returns

returns the width of the data in the buffer in number of pixels



#### 6.3.3.11 GetHeight()

```
virtual size_t IpxCam::Buffer::GetHeight ( ) [pure virtual]
```

This method gets the height of the data buffer.

##### Returns

returns the height of the data in the buffer

#### 6.3.3.12 GetXOffset()

```
virtual size_t IpxCam::Buffer::GetXOffset ( ) [pure virtual]
```

This method returns the XOffset of the data in the buffer.

##### Returns

returns the XOffset of the data in the buffer in number of pixels from the image origin

#### 6.3.3.13 GetYOffset()

```
virtual size_t IpxCam::Buffer::GetYOffset ( ) [pure virtual]
```

This method returns the YOffset of the data in the buffer.

##### Returns

returns the the YOffset of the data in the buffer in number of lines from the image origin

#### 6.3.3.14 GetXPadding()

```
virtual size_t IpxCam::Buffer::GetXPadding ( ) [pure virtual]
```

This method returns the number of extra bytes padded in the X direction.

##### Returns

returns the XPadding of the data in the buffer in number of bytes

#### 6.3.3.15 GetYPadding()

```
virtual size_t IpxCam::Buffer::GetYPadding ( ) [pure virtual]
```

This method returns the number of extra bytes padded in the Y direction.

##### Returns

returns the YPadding of the data in the buffer in number of bytes

#### 6.3.3.16 GetDeliveredHeight()

```
virtual size_t IpxCam::Buffer::GetDeliveredHeight ( ) [pure virtual]
```

This method returns the actual height of delivered data.

##### Returns

returns the actual height of delivered data

#### 6.3.3.17 IsKacFrameB()

```
virtual bool IpxCam::Buffer::IsKacFrameB ( ) [pure virtual]
```

This method indicates if this buffer is Frame A or Frame B, acquired from Cheetah camera with KAC-12040 or KAC-06040 CMOS sensor.

##### Returns

returns true for Frame B, false - otherwise

The documentation for this class was generated from the following file:

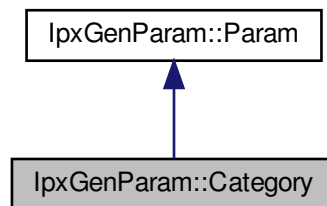
- IpxCameraApi.h

## 6.4 IpxGenParam::Category Class Reference

A class containing methods that the user can access the categories GenICam features. It will access the node object's of an ICategory interface. Each feature of a device will be placed in a [Category](#). The [Category](#) feature is used to present the user with a group of features for the named category.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::Category:



### Public Member Functions

- virtual [ParamType](#) [GetType](#) ()  
*This method returns the node object [Category](#) type.*
- virtual `uint32_t` [GetCount](#) ()=0  
*This method returns the number of categories in the array.*
- virtual `Param *` [GetParamByIndex](#) (`uint32_t` idx, `IpxCamErr *`err)=0  
*This method returns the Parameter by Index.*

#### 6.4.1 Detailed Description

A class containing methods that the user can access the categories GenICam features. It will access the node object's of an ICategory interface. Each feature of a device will be placed in a [Category](#). The [Category](#) feature is used to present the user with a group of features for the named category.

A class containing methods for GenICam [Category](#) Properties. For example, the mapping below will illustrate the I↔Category interfaces categories such as DeviceControl and EventControl.

#### 6.4.2 Member Function Documentation

#### 6.4.2.1 GetType()

```
virtual ParamType IpxGenParam::Category::GetType ( ) [inline], [virtual]
```

This method returns the node object [Category](#) type.

##### Returns

Returns the node object [Category](#) type

Implements [IpxGenParam::Param](#).

#### 6.4.2.2 GetCount()

```
virtual uint32_t IpxGenParam::Category::GetCount ( ) [pure virtual]
```

This method returns the number of categories in the array.

##### Returns

Returns the number of categories in the array

#### 6.4.2.3 GetParamByIndex()

```
virtual Param* IpxGenParam::Category::GetParamByIndex (
    uint32_t idx,
    IpxCamErr * err ) [pure virtual]
```

This method returns the Parameter by Index.

##### Parameters

in	idx	index
out	err	returns the error code: <ul style="list-style-type: none"><li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully returns pointer to the parameter specified by the node name</li><li>• <code>IpxCamErr::IPX_CAM_ERR_INVALID_INDEX</code> - an invalid index for node</li></ul>

**Returns**

returns the pointer to the parameter object

The documentation for this class was generated from the following file:

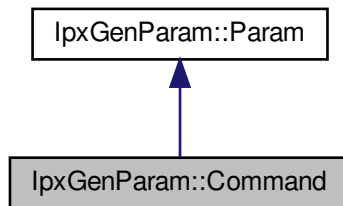
- IpxCameraApi.h

## 6.5 IpxGenParam::Command Class Reference

A Class for GenICam [Command](#) properties contains methods that lets the user submit a command for execution as well as poll the status.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::Command:

**Public Member Functions**

- virtual [ParamType](#) [GetType](#) ()  
*This method returns the node object [Command](#) type.*
- virtual IpxCamErr [Execute](#) ()=0  
*This method executes the command.*
- virtual bool [IsDone](#) (IpxCamErr \*err=NULLPTR)=0  
*This method queries whether the command is executed.*

### 6.5.1 Detailed Description

A Class for GenICam [Command](#) properties contains methods that lets the user submit a command for execution as well as poll the status.

A class containing methods for GenICam [Command](#) Properties. For example, the mapping below will illustrate the ICommand interface for AcquisitionStart. This feature starts the Acquisition of the device.

## 6.5.2 Member Function Documentation

### 6.5.2.1 GetType()

```
virtual ParamType IpxGenParam::Command::GetType ( ) [inline], [virtual]
```

This method returns the node object [Command](#) type.

#### Returns

returns the node object [Command](#) type

Implements [IpxGenParam::Param](#).

### 6.5.2.2 Execute()

```
virtual IpxCamErr IpxGenParam::Command::Execute ( ) [pure virtual]
```

This method executes the command.

#### Returns

the error code

### 6.5.2.3 IsDone()

```
virtual bool IpxGenParam::Command::IsDone (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method queries whether the command is executed.

## Parameters

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully determined that state of execute command</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TREE_ERROR</code> - Unable to access tree</li> </ul>
-----	-----	--

## Returns

If set to TRUE, the Execute command has finished. Otherwise, it returns FALSE.

The documentation for this class was generated from the following file:

- `IpxCameraApi.h`

## 6.6 IpxCam::Device Class Reference

The [Device](#) class represents the device module in the GenTL module hierarchy.

```
#include <IpxCameraApi.h>
```

## Classes

- struct [UploadEventData](#)  
*A structure representing data for uploading to a device.*

## Public Types

- enum [UploadEventType](#) : `uint32_t` { [FlashSectorErase](#), [FlashPageWrite](#), [FlashPageRead](#) }  
*An enum of events used during uploading files to a device.*
- enum [Endianness](#) : `uint8_t` { [BigEndian](#), [LittleEndian](#) }  
*An enum of endianness types of underlying protocol.*
- typedef void `IPXCAM_CALL` [UploadEventCallback](#)([UploadEventType](#) eventType, [UploadEventData](#) eventData)

## Public Member Functions

- virtual [~Device](#) ()  
*A destructor of the [Device](#) class.*
- virtual void [Release](#) ()=0  
*This method releases the instance of the device object. This method releases the device object.*
- virtual uint32\_t [GetNumStreams](#) ()=0  
*This method retrieves the number of the data streams, provided by the [Device](#).*
- virtual [Stream](#) \* [GetStreamByIndex](#) (uint32\_t idx=0)=0  
*This retrieves the pointer to the [Stream](#) object by stream index.*
- virtual [Stream](#) \* [GetStreamById](#) (const char \*id)=0  
*This method retrieves the pointer to the [Stream](#) object by stream identifier.*
- virtual [DeviceInfo](#) \* [GetInfo](#) ()=0  
*This method returns a pointer to the [DeviceInfo](#) object , associated with the [Device](#).*
- virtual lpxCamErr [ReadMem](#) (uint64\_t addr, void \*data, size\_t len)=0  
*This method reads a number of bytes from a given address of the [Device](#).*
- virtual lpxCamErr [WriteMem](#) (uint64\_t addr, const void \*data, size\_t len, size\_t \*written)=0  
*This method writes a number of bytes at a given address.*
- virtual lpxCamErr [UploadFile](#) ([ServiceFileType](#) type, const char \*fileName, uint64\_t flags=0, [UploadEventCallback](#) \*pCallback=nullptr)=0  
*This method uploads the service type file.*
- virtual lpxCamErr [RegisterEvent](#) (uint32\_t eventType, lpxCam::EventCallback \*eventCallback, void \*pPrivate)=0  
*This method registers the [Device](#) class method as a callback method to be called when a eventType occurs.*
- virtual lpxCamErr [UnRegisterEvent](#) (uint32\_t eventType, lpxCam::EventCallback \*eventCallback, void \*pPrivate)=0  
*This method un-registers the [Interface](#) class callback method for the eventType.*
- virtual [lpxGenParam::Array](#) \* [GetTransportParameters](#) (lpxCamErr \*err=nullptr)=0  
*This method returns the transport parameters of the device object.*
- virtual [lpxGenParam::Array](#) \* [GetCameraParameters](#) (lpxCamErr \*err=nullptr)=0  
*This method returns the camera parameters of the device object.*
- virtual lpxCamErr [SaveConfiguration](#) (const char \*fileName)=0  
*This method saves the camera configuration file.*
- virtual lpxCamErr [LoadConfiguration](#) (const char \*fileName)=0  
*This method loads the configuration from file.*
- virtual [Endianness](#) [GetEndianness](#) () const =0  
*This event occurs, when the camera was disconnected from the [System](#).*

## Static Public Attributes

- static const uint32\_t [CameraConnected](#) = 1003  
*This event occurs, if [GenlCam](#) event was triggered by the camera device.*
- static const uint32\_t [CameraDisconnected](#) = 1004  
*This event occurs, when the camera was connected to the [System](#).*



### 6.6.1 Detailed Description

The [Device](#) class represents the device module in the GenTL module hierarchy.

This [Device](#) class provides methods to enable the communication and control of the Imperx device and enumerate/instantiate data stream objects. The methods can be used to enumerate and instantiate the Data [Stream](#) module objects. The device must correspond to the interface transport layer technology. For example, the device could be an Imperx GEV Camera and the transport layer technology would be GEV. The [Device](#) class can be used to retrieve data information about the device by returning the pointer to the [DeviceInfo](#) class. It can be used to retrieve the pointer to the [Stream](#) object and save / load the camera configurations to / from file.

### 6.6.2 Member Typedef Documentation

#### 6.6.2.1 UploadEventCallback

```
typedef void IPXCAM_CALL IpxCam::Device::UploadEventCallback(UploadEventType eventType, UploadEvent↔  
EventData eventData)
```

UploadEventCallback param[in] eventType type of the event param[in] eventData event Data

### 6.6.3 Member Enumeration Documentation

#### 6.6.3.1 UploadEventType

```
enum IpxCam::Device::UploadEventType : uint32_t
```

An enum of events used during uploading files to a device.

Enumerator

FlashSectorErase	Enum value FlashSectorErase.
FlashPageWrite	Enum value FlashPagewrite.
FlashPageRead	Enum value FlashPageRead.

#### 6.6.3.2 Endianness

```
enum IpxCam::Device::Endianness : uint8_t
```

An enum of endianness types of underlying protocol.

#### Enumerator

BigEndian	Enum value Big-endian.
LittleEndian	Enum value Little-endian

## 6.6.4 Constructor & Destructor Documentation

### 6.6.4.1 ~Device()

```
virtual IpxCam::Device::~~Device ( ) [inline], [virtual]
```

A destructor of the [Device](#) class.

Destructor. Destroys the [Device](#) and all its descendants.

## 6.6.5 Member Function Documentation

### 6.6.5.1 GetNumStreams()

```
virtual uint32_t IpxCam::Device::GetNumStreams ( ) [pure virtual]
```

This method retrieves the number of the data streams, provided by the [Device](#).

#### Returns

retrurns the number of the data streams

### 6.6.5.2 GetStreamByIndex()

```
virtual Stream* IpxCam::Device::GetStreamByIndex (
    uint32_t idx = 0 ) [pure virtual]
```

This retrieves the pointer to the [Stream](#) object by stream index.

**Parameters**

in	<i>idx</i>	stream index value
----	------------	--------------------

**Returns**

returns the pointer to the [Stream](#) object

**6.6.5.3 GetStreamById()**

```
virtual Stream* IpxCam::Device::GetStreamById (
    const char * id ) [pure virtual]
```

This method retrieves the pointer to the [Stream](#) object by stream identifier.

**Parameters**

in	<i>id</i>	pointer to the string representing the stream identifier
----	-----------	--

**Returns**

returns the pointer to the [Stream](#) object

**6.6.5.4 GetInfo()**

```
virtual DeviceInfo* IpxCam::Device::GetInfo ( ) [pure virtual]
```

This method returns a pointer to the [DeviceInfo](#) object , associated with the [Device](#).

**Returns**

returns the pointer to the [DeviceInfo](#) object

**6.6.5.5 ReadMem()**

```
virtual IpxCamErr IpxCam::Device::ReadMem (
    uint64_t addr,
    void * data,
    size_t len ) [pure virtual]
```

This method reads a number of bytes from a given address of the [Device](#).

**Parameters**

in	<i>addr</i>	Byte address to read from
in	<i>data</i>	pointer to a user allocated byte data buffer
in	<i>len</i>	size of the amount of bytes to read from the register map address

**Returns**

returns ErrorCode

**6.6.5.6 WriteMem()**

```
virtual IpxCamErr IpxCam::Device::WriteMem (
    uint64_t addr,
    const void * data,
    size_t len,
    size_t * written ) [pure virtual]
```

This method writes a number of bytes at a given address.

**Parameters**

in	<i>addr</i>	Byte address to read from
in	<i>data</i>	pointer to a user allocated byte data buffer
in	<i>len</i>	size of the amount of bytes to write to the register map address
out	<i>written</i>	size of bytes written

**Returns**

returns ErrorCode

**6.6.5.7 UploadFile()**

```
virtual IpxCamErr IpxCam::Device::UploadFile (
    ServiceFileType type,
    const char * fileName,
    uint64_t flags = 0,
    UploadEventCallback * pCallback = nullptr ) [pure virtual]
```

This method uploads the service type file.

## Parameters

in	<i>type</i>	ServiceFile Type
in	<i>fileName</i>	Name of the ServiceFile name
in	<i>flags</i>	flag
in	<i>pCallback</i>	pointer to the event Callback

## Returns

returns ErrorCode

## 6.6.5.8 RegisterEvent()

```
virtual IpxCamErr IpxCam::Device::RegisterEvent (
    uint32_t eventType,
    IpxCam::EventCallback * eventCallback,
    void * pPrivate ) [pure virtual]
```

This method registers the [Device](#) class method as a callback method to be called when a eventType occurs.

## Parameters

in	<i>eventType</i>	Event Type, can receive one of the following values: <ul style="list-style-type: none"> <li>• <b>GenICamEvent</b> [1002] - this event occurs, if GenICam event was triggered by the camera</li> <li>• <b>CameraConnected</b> [1003] - this event occurs, when camera was connected to the <a href="#">System</a></li> <li>• <b>CameraDisconnected</b> [1004] - this event occurs, when camera was disconnected from the <a href="#">System</a></li> </ul>
in	<i>eventCallback</i>	event CallBack
in	<i>pPrivate</i>	pointer to user's data

## Returns

returns Error code

## 6.6.5.9 UnRegisterEvent()

```
virtual IpxCamErr IpxCam::Device::UnRegisterEvent (
    uint32_t eventType,
```

```
IpxCam::EventCallback * eventCallback,
void * pPrivate ) [pure virtual]
```

This method un-registers the [Interface](#) class callback method for the eventType.

#### Parameters

in	<i>eventType</i>	Event Type, can receive one of the following values: <ul style="list-style-type: none"> <li>• <b>GenICamEvent</b> [1002] - this event occurs, if GenICam event was triggered by the camera</li> <li>• <b>CameraConnected</b> [1003] - this event occurs, when camera was connected to the <a href="#">System</a></li> <li>• <b>CameraDisconnected</b> [1004] - this event occurs, when camera was disconnected from the <a href="#">System</a></li> </ul>
in	<i>eventCallback</i>	event CallBack
in	<i>pPrivate</i>	pointer to user's data

#### Returns

returns Error code

#### 6.6.5.10 GetTransportParameters()

```
virtual IpxGenParam::Array* IpxCam::Device::GetTransportParameters (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method returns the transport parameters of the device object.

#### Parameters

out	<i>err</i>	returns error code
-----	------------	--------------------

#### Returns

returns the Transport parameter array

#### 6.6.5.11 GetCameraParameters()

```
virtual IpxGenParam::Array* IpxCam::Device::GetCameraParameters (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method returns the camera parameters of the device object.

**Parameters**

out	err	returns error code
-----	-----	--------------------

**Returns**

returns the Camera Parameters array

**6.6.5.12 SaveConfiguration()**

```
virtual IpxCamErr IpxCam::Device::SaveConfiguration (  
    const char * fileName ) [pure virtual]
```

This method saves the camera configuration file.

**Parameters**

in	<i>fileName</i>	Configuration file name
----	-----------------	-------------------------

**Returns**

returns Error code

**6.6.5.13 LoadConfiguration()**

```
virtual IpxCamErr IpxCam::Device::LoadConfiguration (  
    const char * fileName ) [pure virtual]
```

This method loads the configuration from file.

**Parameters**

in	<i>fileName</i>	Configuration file name
----	-----------------	-------------------------

**Returns**

returns Error code

#### 6.6.5.14 GetEndianness()

```
virtual Endianness IpxCam::Device::GetEndianness ( ) const [pure virtual]
```

This event occurs, when the camera was disconnected from the [System](#).

This method returns endianness of underlying protocol

##### Returns

returns endianness

The documentation for this class was generated from the following file:

- IpxCameraApi.h

## 6.7 IpxCam::DeviceInfo Class Reference

[DeviceInfo](#) class provides the information about the device.

```
#include <IpxCameraApi.h>
```

### Public Member Functions

- virtual [~DeviceInfo](#) ()  
*A destructor of the [DeviceInfo](#) class.*
- virtual [Interface](#) \* [GetInterface](#) ()=0  
*This method returns the interface of the device object.*
- virtual const char \* [GetID](#) ()=0  
*This method returns the unique device identifier string for the Imperx Camera device object.*
- virtual const char \* [GetVendor](#) ()=0  
*This method returns the device vendor name of the device object.*
- virtual const char \* [GetModel](#) ()=0  
*This method returns the Camera device model name of the device object.*
- virtual const char \* [GetDisplayName](#) ()=0  
*This method returns the display name of the Camera device object.*
- virtual const char \* [GetUserDefinedName](#) ()=0  
*This method returns the user defined name of the Camera device.*
- virtual const char \* [GetSerialNumber](#) ()=0  
*This method returns the serial number of the Camera device .*
- virtual const char \* [GetVersion](#) ()=0  
*This method returns the device version of the device object.*
- virtual int32\_t [GetAccessStatus](#) ()=0  
*This method returns the information about the current device access status.*
- virtual const char \* [GetUSB3HostInfo](#) ()=0  
*This method returns the information about USB3 host controller where the camera is connected to.*
- virtual IpxCamErr [ForceIP](#) (const char \*addr, const char \*netmask, const char \*gateway)=0  
*This method sets IP address to GEV camera (using ForceIP command)*
- virtual IpxCamErr [ForceIP](#) (uint32\_t addr, uint32\_t netmask, uint32\_t gateway)=0  
*This method sets IP address to GEV camera (using ForceIP command)*



### 6.7.1 Detailed Description

[DeviceInfo](#) class provides the information about the device.

The [DeviceInfo](#) class can be used to retrieve the information about the device, and to create the [Device](#) object by `IpxCam_CreateDevice()` call

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 `~DeviceInfo()`

```
virtual IpxCam::DeviceInfo::~~DeviceInfo ( ) [inline], [virtual]
```

A destructor of the [DeviceInfo](#) class.

Destructor. Destroys the [DeviceInfo](#) and all its descendants.

### 6.7.3 Member Function Documentation

#### 6.7.3.1 `GetInterface()`

```
virtual Interface* IpxCam::DeviceInfo::GetInterface ( ) [pure virtual]
```

This method returns the interface of the device object.

Returns the [Interface](#) pointer for the device, associated with the [DeviceInfo](#) object

##### Returns

returns the [Interface](#)

#### 6.7.3.2 `GetID()`

```
virtual const char* IpxCam::DeviceInfo::GetID ( ) [pure virtual]
```

This method returns the unique device identifier string for the Imperx Camera device object.

##### Returns

returns the unique device identifier string for the Imperx Camera device

#### 6.7.3.3 GetVendor()

```
virtual const char* IpxCam::DeviceInfo::GetVendor ( ) [pure virtual]
```

This method returns the device vendor name of the device object.

##### Returns

returns the Imperx Camera device vendor name

#### 6.7.3.4 GetModel()

```
virtual const char* IpxCam::DeviceInfo::GetModel ( ) [pure virtual]
```

This method returns the Camera device model name of the device object.

##### Returns

returns the Camera device model name

#### 6.7.3.5 GetDisplayName()

```
virtual const char* IpxCam::DeviceInfo::GetDisplayName ( ) [pure virtual]
```

This method returns the display name of the Camera device object.

##### Returns

returns the user readable name of the Camera device

#### 6.7.3.6 GetUserDefinedName()

```
virtual const char* IpxCam::DeviceInfo::GetUserDefinedName ( ) [pure virtual]
```

This method returns the user defined name of the Camera device.

##### Returns

returns the user defined name of the Camera device

#### 6.7.3.7 GetSerialNumber()

```
virtual const char* IpxCam::DeviceInfo::GetSerialNumber ( ) [pure virtual]
```

This method returns the serial number of the Camera device .

##### Returns

returns the serial number of the Camera device

#### 6.7.3.8 GetVersion()

```
virtual const char* IpxCam::DeviceInfo::GetVersion ( ) [pure virtual]
```

This method returns the device version of the device object.

##### Returns

returns the [Device](#) version

#### 6.7.3.9 GetAccessStatus()

```
virtual int32_t IpxCam::DeviceInfo::GetAccessStatus ( ) [pure virtual]
```

This method returns the information about the current device access status.

Returns the information about the current device access status

##### Returns

Status Access Code, can receive one of the following values:

- **AccessStatusUnknown** [0] - The current availability of the device is unknown.
- **AccessStatusReadWrite** [1] - The device is available for Read/Write access
- **AccessStatusReadOnly** [2] - The device is available for Read only access
- **AccessStatusNoAccess** [3] - The device is not available either because it is already open or because it is not reachable.

#### 6.7.3.10 GetUSB3HostInfo()

```
virtual const char* IpxCam::DeviceInfo::GetUSB3HostInfo ( ) [pure virtual]
```

This method returns the information about USB3 host controller where the camera is connected to.

Returns the information about USB3 host controller

##### Returns

returns the pointer to string structure or nullptr for non-USB camera

#### 6.7.3.11 ForceIP() [1/2]

```
virtual IpxCamErr IpxCam::DeviceInfo::ForceIP (
    const char * addr,
    const char * netmask,
    const char * gateway ) [pure virtual]
```

This method sets IP address to GEV camera (using ForceIP command)

Set IP address to GEV camera.

##### Parameters

in	<i>addr</i>	IP Address to set
in	<i>netmask</i>	IP Address subnet mask
in	<i>gateway</i>	Gateway address

##### Returns

returns Error code

#### 6.7.3.12 ForceIP() [2/2]

```
virtual IpxCamErr IpxCam::DeviceInfo::ForceIP (
    uint32_t addr,
    uint32_t netmask,
    uint32_t gateway ) [pure virtual]
```

This method sets IP address to GEV camera (using ForceIP command)

Set IP address to GEV camera.

## Parameters

in	<i>addr</i>	IP Address to set (host byte order)
in	<i>netmask</i>	IP Address subnet mask (host byte order)
in	<i>gateway</i>	Gateway address (host byte order)

## Returns

returns Error code

The documentation for this class was generated from the following file:

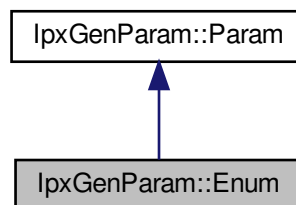
- IpxCameraApi.h

## 6.8 IpxGenParam::Enum Class Reference

Interface Class for Enumeration properties.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::Enum:



## Public Member Functions

- virtual [ParamType](#) [GetType](#) ()  
*This method returns the node object [Enum](#) type.*
- virtual [size\\_t](#) [GetEnumEntriesCount](#) (IpxCamErr \*err=nullptr)=0  
*This method gets the number of entry nodes.*
- virtual [EnumEntry](#) \* [GetEnumEntryByIndex](#) (size\_t aIndex)=0  
*This method gets the [Enum](#) Entry node by the Index number.*
- virtual [EnumEntry](#) \* [GetEnumEntryByName](#) (const char \*name)=0

*This method gets the [Enum](#) Entry node by Name.*

- virtual [EnumEntry](#) \* [GetEnumEntryByValue](#) (int64\_t val)=0

*This method gets the [Enum](#) Entry node by Value.*

- virtual int64\_t [GetValue](#) (IpxCamErr \*err=nullptr)=0

*This method gets the [Enum](#) Entry node value.*

- virtual const char \* [GetValueStr](#) (IpxCamErr \*err=nullptr)=0

*This method gets the [Enum](#) Entry node [String](#).*

- virtual IpxCamErr [SetValue](#) (int64\_t val)=0

*This method sets the [Enum](#) Entry node value.*

- virtual IpxCamErr [SetValueStr](#) (const char \*val)=0

*This method sets the [Enum](#) Entry node [String](#).*

### 6.8.1 Detailed Description

Interface Class for Enumeration properties.

A [Enum](#) class. This class is used to map the name of the Enumeration interface type for the drop down box.

For example, the mapping below illustrates the enumeration "WhiteBalanceMode".

### 6.8.2 Member Function Documentation

#### 6.8.2.1 GetType()

```
virtual ParamType IpxGenParam::Enum::GetType ( ) [inline], [virtual]
```

This method returns the node object [Enum](#) type.

#### Returns

If the method succeeds, it will returns the [Enum](#) parameter type.

Implements [IpxGenParam::Param](#).

#### 6.8.2.2 GetEnumEntriesCount()

```
virtual size_t IpxGenParam::Enum::GetEnumEntriesCount (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the number of entry nodes.

## Parameters

out	err	returns error code:
		<ul style="list-style-type: none"><li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the number of EnumEntries</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li></ul>

## Returns

Returns the number of enum entry nodes.

## 6.8.2.3 GetEnumEntryByIndex()

```
virtual EnumEntry* IpxGenParam::Enum::GetEnumEntryByIndex (
    size_t aIndex ) [pure virtual]
```

This method gets the [Enum](#) Entry node by the Index number.

## Parameters

in	<i>aIndex</i>	Index number
----	---------------	--------------

## Returns

If the method succeeds, it returns the [Enum](#) Entry node.

## 6.8.2.4 GetEnumEntryByName()

```
virtual EnumEntry* IpxGenParam::Enum::GetEnumEntryByName (
    const char * name ) [pure virtual]
```

This method gets the [Enum](#) Entry node by Name.

## Parameters

in	<i>name</i>	Entry Name
----	-------------	------------

**Returns**

If the method succeeds, it returns the [Enum](#) Entry node.

**6.8.2.5 GetEnumEntryByValue()**

```
virtual EnumEntry* IpxGenParam::Enum::GetEnumEntryByValue (
    int64_t val ) [pure virtual]
```

This method gets the [Enum](#) Entry node by Value.

**Parameters**

in	val	Entry Value
----	-----	-------------

**Returns**

If the method succeeds, it returns the [Enum](#) Entry node.

**6.8.2.6 GetValue()**

```
virtual int64_t IpxGenParam::Enum::GetValue (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [Enum](#) Entry node value.

**Parameters**

out	err	returns error code: <ul style="list-style-type: none"><li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the <a href="#">Enum</a> Entry node value</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li></ul>
-----	-----	---

**Returns**

If the method succeeds, it returns the [Enum](#) Entry node value.



### 6.8.2.7 GetValueStr()

```
virtual const char* IpxGenParam::Enum::GetValueStr (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [Enum](#) Entry node [String](#).

#### Parameters

out	<i>err</i>	returns error code:
		<ul style="list-style-type: none"><li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully get the <a href="#">Enum</a> Entry node string</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li></ul>

#### Returns

If the method succeeds, it returns the [Enum](#) Entry node string.

### 6.8.2.8 SetValue()

```
virtual IpxCamErr IpxGenParam::Enum::SetValue (
    int64_t val ) [pure virtual]
```

This method sets the [Enum](#) Entry node value.

#### Parameters

in	<i>val</i>	<a href="#">Enum</a> Entry node value
----	------------	---------------------------------------

#### Returns

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the [Enum](#) value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type

### 6.8.2.9 SetValueStr()

```
virtual IpxCamErr IpxGenParam::Enum::SetValueStr (
    const char * val ) [pure virtual]
```

This method sets the [Enum](#) Entry node [String](#).

#### Parameters

in	val	<a href="#">Enum</a> Entry node <a href="#">String</a>
----	-----	--

#### Returns

returns the error code

The documentation for this class was generated from the following file:

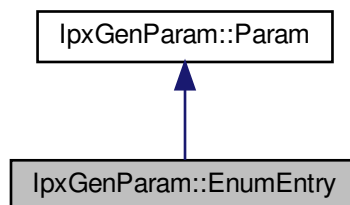
- IpxCameraApi.h

## 6.9 IpxGenParam::EnumEntry Class Reference

A Class for GenICam [Enum](#) Entries has methods to access the Enumeration node [Enum](#) Entry of the GenICam.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::EnumEntry:



### Public Member Functions

- virtual [ParamType](#) [GetType](#) ()  
*This method returns the node object [EnumEntry](#) type.*
- virtual int64\_t [GetValue](#) (IpxCamErr \*err=nullptr)=0  
*This method gets the [EnumEntry](#) numerical value.*
- virtual const char \* [GetValueStr](#) (IpxCamErr \*err=nullptr)=0  
*This method gets the [EnumEntry](#) [String](#) value.*

### 6.9.1 Detailed Description

A Class for GenICam [Enum](#) Entries has methods to access the Enumeration node [Enum](#) Entry of the GenICam.

A GenICam [EnumEntry](#) class. For example, the mapping below illustrates entries of the IEnumeration interface for the AOI2\_Select feature. This feature can select the mode of operation for Slave AOI #2. The enumeration entries could be "Off", "Include", and "Exclude".

### 6.9.2 Member Function Documentation

#### 6.9.2.1 GetType()

```
virtual ParamType IpxGenParam::EnumEntry::GetType ( ) [inline], [virtual]
```

This method returns the node object [EnumEntry](#) type.

#### Returns

If the method succeeds, it returns the ParamType object type of the [EnumEntry](#).

Implements [IpxGenParam::Param](#).

#### 6.9.2.2 GetValue()

```
virtual int64_t IpxGenParam::EnumEntry::GetValue (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [EnumEntry](#) numerical value.

#### Parameters

out	<i>err</i>	returns error code:
		<ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully indicates <a href="#">EnumEntry</a> value was retrieved</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>

**Returns**

If the method succeeds, it returns the value read of the [EnumEntry](#).

**6.9.2.3 GetValueStr()**

```
virtual const char* IpxGenParam::EnumEntry::GetValueStr (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [EnumEntry String](#) value.

**Parameters**

out	err	returns error code:
		<ul style="list-style-type: none"> <li>• IpxCamErr::IPX_CAM_ERR_OK - Successfully indicates <a href="#">EnumEntry</a> string value was retrieved</li> <li>• IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR - Unable to access genicam specified node</li> <li>• IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR - Unable to access genicam specified node type</li> </ul>

**Returns**

If the method succeeds, it returns the [String](#) value read of the [EnumEntry](#).

The documentation for this class was generated from the following file:

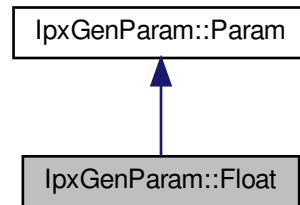
- IpxCameraApi.h

**6.10 IpxGenParam::Float Class Reference**

Interface Class for GenICam [Float](#) properties.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::Float:



## Public Member Functions

- virtual `ParamType GetType ()`  
*This method returns the node object `Float` type.*
- virtual `IpxCamErr SetValue (double val)=0`  
*This method sets the node value.*
- virtual `double GetValue (IpxCamErr *err=nullptr)=0`  
*This method gets the `Float` node value.*
- virtual `double GetMin (IpxCamErr *err=nullptr)=0`  
*This method gets the minimum value.*
- virtual `double GetMax (IpxCamErr *err=nullptr)=0`  
*This method gets the maximum value.*
- virtual `const char * GetUnit (IpxCamErr *err=nullptr)=0`  
*This method gets the Unit.*

### 6.10.1 Detailed Description

Interface Class for GenICam `Float` properties.

A GenICam `Float` class. For example, mapping a slider with value, min, and max plus a physical unit

### 6.10.2 Member Function Documentation

### 6.10.2.1 GetType()

```
virtual ParamType IpxGenParam::Float::GetType ( ) [inline], [virtual]
```

This method returns the node object [Float](#) type.

#### Returns

returns the parameter type

Implements [IpxGenParam::Param](#).

### 6.10.2.2 SetValue()

```
virtual IpxCamErr IpxGenParam::Float::SetValue (
    double val ) [pure virtual]
```

This method sets the node value.

#### Parameters

in	val	The value to set
----	-----	------------------

#### Returns

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the [Float](#) value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the value entered is out of range

### 6.10.2.3 GetValue()

```
virtual double IpxGenParam::Float::GetValue (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the [Float](#) node value.

## Parameters

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully get the <a href="#">Float</a> value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	--

## Returns

Gets the [Float](#) node value

## 6.10.2.4 GetMin()

```
virtual double IpxGenParam::Float::GetMin (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the minimum value.

## Parameters

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the Minimum float value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	---

## Returns

Returns the minimum

## 6.10.2.5 GetMax()

```
virtual double IpxGenParam::Float::GetMax (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the maximum value.

**Parameters**

out	err	returns error code:
		<ul style="list-style-type: none"><li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the Maximum float value</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li></ul>

**Returns**

Returns the maximum

**6.10.2.6 GetUnit()**

```
virtual const char* IpxGenParam::Float::GetUnit (  
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the Unit.

**Parameters**

out	err	returns error code:
		<ul style="list-style-type: none"><li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the units</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li><li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li></ul>

**Returns**

Returns the unit

The documentation for this class was generated from the following file:

- `IpxCameraApi.h`



## 6.11 IpxGui::IpxGenParamTreeView Class Reference

The [IpxGenParamTreeView](#) class is composed of functions to set and clear parameters of the node tree of the camera. The node tree can be set with the current parameters stored in the [IpxGenParam::Array](#) and [GenApi::INodeMap](#) class.

```
#include <IpxCameraGuiApi.h>
```

### Public Member Functions

- virtual [~IpxGenParamTreeView](#) ()  
*A destructor of the [IpxGenParamTreeView](#) class.*
- virtual void [setParams](#) ([IpxGenParam::Array](#) \*genParam)=0  
*This method sets the parameters of the node tree by the information extracted from the [IpxGenParam::Array](#) class.*
- virtual void [setParams](#) ([IPX\\_GENAPI\\_NS::INodeMap](#) \*nodemap)=0  
*This method sets the parameters of the node tree with parameters retrieved from the [GenApi::INodeMap](#) class.*
- virtual void [clearParams](#) ()=0  
*This method clears the parameters of the node tree that have been set by the instance of the [IpxGui::IpxGenParamTreeView](#) class.*
- virtual [Visibility](#) [visibility](#) () const =0  
*This method returns the current visibility mode.*
- virtual void [setVisibility](#) ([Visibility](#) [visibility](#))=0  
*This method sets visibility mode.*
- virtual const char \* [saveState](#) () const =0  
*This method saves the current state of the Tree View.*
- virtual void [loadState](#) (const char \*state)=0  
*This method loads the state of the Tree View.*
- virtual void [setPollingTime](#) ([uint64\\_t](#) pollingTime)=0  
*This method sets polling time.*
- virtual [uint64\\_t](#) [getPollingTime](#) ()=0  
*This method retrives current polling time.*
- virtual void [enablePolling](#) (bool enable)=0  
*This method enables polling.*
- virtual bool [isPollingEnabled](#) ()=0  
*This method retrives current polling state.*

### 6.11.1 Detailed Description

The [IpxGenParamTreeView](#) class is composed of functions to set and clear parameters of the node tree of the camera. The node tree can be set with the current parameters stored in the [IpxGenParam::Array](#) and [GenApi::INodeMap](#) class.

A Class for [IpxGenParamTreeView](#). For example, we can declare the instance of [IpxGui::IpxGenParamTreeView](#) class as `m_parameterView` as shown below:

```
IpxGui::IpxGenParamTreeView* m_ParameterView;
```

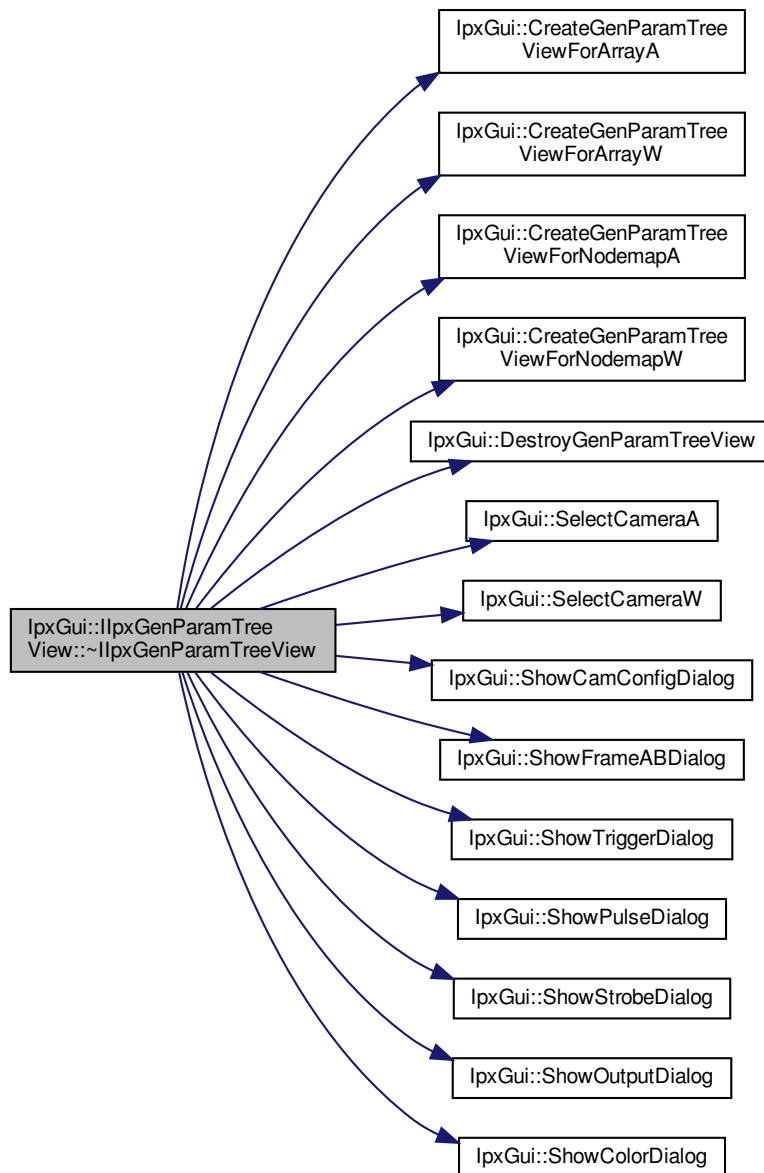
## 6.11.2 Constructor & Destructor Documentation

### 6.11.2.1 ~IpxGenParamTreeView()

```
virtual IpxGui::IpxGenParamTreeView::~IpxGenParamTreeView ( ) [inline], [virtual]
```

A destructor of the [IpxGenParamTreeView](#) class.

Destructor. Destroys the [IpxGenParamTreeView](#) and all its descendants. Here is the call graph for this function:



### 6.11.3 Member Function Documentation

#### 6.11.3.1 setParams() [1/2]

```
virtual void IpxGui::IIPxGenParamTreeView::setParams (
    IpxGenParam::Array * genParam ) [pure virtual]
```

This method sets the parameters of the node tree by the information extracted from the [IpxGenParam::Array](#) class.

##### Parameters

in	<i>genParam</i>	The pointer to the <a href="#">IpxGenParam::Array</a> class.
----	-----------------	--

##### Returns

Void. For example, set the Camera Parameters to the corresponding fields of the TreeView as shown below:

```
//Establish camera device
m_camera = IpxCam_CreateDevice(m_devInfo);

//If the camera exist, set the camera parameter to the corresponding fields of the GUI TreeView
if(m_camera){
    m_ParameterView->setParams(m_camera->GetCameraParameters());
}
```

#### 6.11.3.2 setParams() [2/2]

```
virtual void IpxGui::IIPxGenParamTreeView::setParams (
    IPX_GENAPI_NS::INodeMap * nodemap ) [pure virtual]
```

This method sets the parameters of the node tree with parameters retrieved from the [GenApi::INodeMap](#) class.

The [INodeMap](#) consists of a list of nodes representing the GenICam compliant camera high-level features.

##### Parameters

in	<i>nodemap</i>	The pointer to the <a href="#">GenApi::INodeMap</a> class.
----	----------------	--

##### Returns

Void. For example, setting the parameters of the node map.

```
//Instantiate the IpxGui::IIPxGenParamTreeView
IpxGui::IIPxGenParamTreeView* m_ParameterView;
...
auto params = GetCameraParameters(&retErr);
```

```

    if(!params) {
        return retErr;
    }
    GenApi::INodeMap *nodemap = param->GetNodeMap(&retErr);
    if(!nodemap){
        return retErr;
    }
    ...
    //Set the nodemap parameters of the GUI TreeView
    m_ParameterView->setParams(nodemap);
    ...

```

### 6.11.3.3 clearParams()

```
virtual void IpxGui::IIpxGenParamTreeView::clearParams ( ) [pure virtual]
```

This method clears the parameters of the node tree that have been set by the instance of the [IpxGui::IIpxGenParamTreeView](#) class.

#### Returns

Void. For example, clear all the parameters after we disconnect the camera as shown below:

```

//Instantiate the IpxGui::IIpxGenParamTreeView
IpxGui::IIpxGenParamTreeView* m_ParameterView;

//Connect the camera
...
//Set some camera parameters
...
//Perform some actions
...
//Clear parameters during disconnecting process of camera
m_ParameterView->clearParam();

```

### 6.11.3.4 visibility()

```
virtual Visibility IpxGui::IIpxGenParamTreeView::visibility ( ) const [pure virtual]
```

This method returns the current visibility mode.

This method retrieves the current setting of the user level access feature.

#### Returns

Returns the current setting of the user level access feature.

### 6.11.3.5 setVisibility()

```
virtual void IpxGui::IIpxGenParamTreeView::setVisibility (
    Visibility visibility ) [pure virtual]
```

This method sets visibility mode.

It sets the current visibility mode.

**Parameters**

in	<i>visibility</i>	The visibility parameter used to set the visibility mode.
----	-------------------	---

**Returns**

Void.

**6.11.3.6 saveState()**

```
virtual const char* IpxGui::IpxGenParamTreeView::saveState ( ) const [pure virtual]
```

This method saves the current state of the Tree View.

saves the current state of the Tree View.

**Returns**

If the method succeeds, the method returns pointer to the array of data to be saved. Otherwise, the return value is nullptr. The array consists of string values separated by token. Just save this data somewhere if you want to restore state later.

**6.11.3.7 loadState()**

```
virtual void IpxGui::IpxGenParamTreeView::loadState (
    const char * state ) [pure virtual]
```

This method loads the state of the Tree View.

loads the state of the Tree View. The individual node can be in a state of expanded or collapse state.

**Parameters**

in	<i>state</i>	to be loaded The array consists of string values separated by token. Load previously saved state of the Tree View.
----	--------------	--

**6.11.3.8 setPollingTime()**

```
virtual void IpxGui::IpxGenParamTreeView::setPollingTime (
    uint64_t pollingTime ) [pure virtual]
```

This method sets polling time.

sets the polling time. Polling should be enabled by [enablePolling\(\)](#) function

#### Parameters

in	<i>pollingTime</i>	time in msec to be set
----	--------------------	------------------------

#### 6.11.3.9 getPollingTime()

```
virtual uint64_t IpxGui::IIpxGenParamTreeView::getPollingTime ( ) [pure virtual]
```

This method retrives current polling time.

retrives the polling time. Polling should be enabled by [enablePolling\(\)](#) function

#### Returns

current polling time in msec

The documentation for this class was generated from the following file:

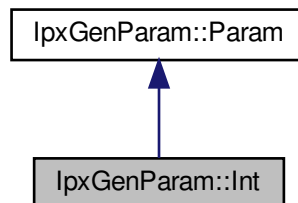
- IpxCameraGuiApi.h

## 6.12 IpxGenParam::Int Class Reference

Interface Class for GenICam [Int](#) properties.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::Int:



## Public Member Functions

- virtual [ParamType](#) [GetType](#) ()  
*This method returns the node object [Int](#) type.*
- virtual IpxCamErr [SetValue](#) (int64\_t val)=0  
*This method sets the [Int](#) node value.*
- virtual int64\_t [GetValue](#) (IpxCamErr \*err=nullptr)=0  
*This method gets the [Int](#) node value.*
- virtual int64\_t [GetMin](#) (IpxCamErr \*err=nullptr)=0  
*This method gets the minimum value.*
- virtual int64\_t [GetMax](#) (IpxCamErr \*err=nullptr)=0  
*This method gets the maximum value.*
- virtual int64\_t [GetIncrement](#) (IpxCamErr \*err=nullptr)=0  
*This method gets the Increment value.*

### 6.12.1 Detailed Description

Interface Class for GenICam [Int](#) properties.

A GenICam [Int](#) class. For example, the mapping below illustrates the ability to set the "Width" value.

### 6.12.2 Member Function Documentation

#### 6.12.2.1 [GetType](#)()

```
virtual ParamType IpxGenParam::Int::GetType ( ) [inline], [virtual]
```

This method returns the node object [Int](#) type.

#### Returns

returns the parameter type

Implements [IpxGenParam::Param](#).

#### 6.12.2.2 [SetValue](#)()

```
virtual IpxCamErr IpxGenParam::Int::SetValue (
    int64_t val ) [pure virtual]
```

This method sets the [Int](#) node value.

**Parameters**

in	val	Int node value
----	-----	----------------

**Returns**

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the Int value
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type
- `IpxCamErr::IPX_CAM_GENICAM_OUT_OF_RANGE` - the value entered is out of range

**6.12.2.3 GetValue()**

```
virtual int64_t IpxGenParam::Int::GetValue (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the Int node value.

**Parameters**

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the Int value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	---

**Returns**

returns the Int node value

**6.12.2.4 GetMin()**

```
virtual int64_t IpxGenParam::Int::GetMin (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the minimum value.



## Parameters

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the Minimum int value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	---

## Returns

Returns the minimum

## 6.12.2.5 GetMax()

```
virtual int64_t IpxGenParam::Int::GetMax (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the maximum value.

## Parameters

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the Maximum int value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	---

## Returns

Returns the maximum

## 6.12.2.6 GetIncrement()

```
virtual int64_t IpxGenParam::Int::GetIncrement (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the Increment value.

## Parameters

out	err	returns error code : <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the increment value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	--

## Returns

Returns the increment

The documentation for this class was generated from the following file:

- `IpxCameraApi.h`

## 6.13 IpxCam::Interface Class Reference

The [Interface](#) class represents a interface module in the GenTL module hierarchy.

```
#include <IpxCameraApi.h>
```

### Public Member Functions

- virtual [~Interface](#) ()  
*A destructor of the [Interface](#) class.*
- virtual [DeviceInfoList](#) \* [GetDeviceInfoList](#) ()=0  
*This method gets the device info list of [DeviceInfo](#) objects for the devices available on this [Interface](#).*
- virtual [DeviceInfo](#) \* [GetFirstDeviceInfo](#) ()=0  
*This method retrieves the [DeviceInfo](#) object for the first device available on this [Interface](#).*
- virtual [DeviceInfo](#) \* [GetDeviceInfoById](#) (const char \*deviceId)=0  
*This method retrieves the device info for the specified device identifier.*
- virtual `IpxCamErr` [ReEnumerateDevices](#) (bool \*pChanged, uint64\_t iTimeout)=0  
*This method re-enumerates the devices.*
- virtual const char \* [GetDescription](#) ()=0  
*This method gets the description of the interface object.*
- virtual [InterfaceType](#) [GetType](#) ()=0  
*This method gets the type of interface object.*
- virtual const char \* [GetId](#) ()=0  
*This method gets the identifier of the interface object.*
- virtual const char \* [GetVersion](#) ()=0

*This method gets the version of [Interface](#) driver.*

- virtual lpxCamErr [RegisterEvent](#) (uint32\_t eventType, lpxCam::EventCallback \*eventCallback, void \*pPrivate)=0

*This method registers the [Interface](#) class method as a callback method to be called when a eventType occurs.*

- virtual lpxCamErr [UnRegisterEvent](#) (uint32\_t eventType, lpxCam::EventCallback \*eventCallback, void \*pPrivate)=0

*This method unregisters the [Interface](#) class callback method for the eventType.*

- virtual lpxGenParam::Array \* [GetParameters](#) (lpxCamErr \*err=nullptr)=0

*This method returns the parameter array used to control the Imperx Camera device.*

- virtual Device \* [CreateDeviceFromConfig](#) (const char \*fileName, lpxCamErr \*err=nullptr)=0

*This method creates the device based on the information specified in the configuration file.*

### 6.13.1 Detailed Description

The [Interface](#) class represents a interface module in the GenTL module hierarchy.

It represents an individual physical interface. For example, a network interface card (NIC), a frame grabber board, U3V, and GEV in the GenTL system. This [Interface](#) class gets the enumeration and available devices on the physical interface in the system.

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 ~Interface()

```
virtual lpxCam::Interface::~Interface ( ) [inline], [virtual]
```

A destructor of the [Interface](#) class.

Destructor. Destroys the [Interface](#) and all its descendants.

### 6.13.3 Member Function Documentation

### 6.13.3.1 GetDeviceInfoList()

```
virtual DeviceInfoList* IpxCam::Interface::GetDeviceInfoList ( ) [pure virtual]
```

This method gets the device info list of [DeviceInfo](#) objects for the devices available on this [Interface](#).

#### Returns

returns the pointer to DeviceInfoList object

For example,

```
// Get the Device Info List for the Interface
// List has to be released, let us use unique pointer
auto del = [] (IpxCam::DeviceInfoList *l) { l->Release(); };
std::unique_ptr<IpxCam::DeviceInfoList, decltype(del)> list(iface->GetDeviceInfoList(), del);

if (list->GetCount() == 0)
{
    std::cout << "No Interface Available. " << endl;
    exit(1);
}

IpxCam::Device *device = nullptr;
for (auto devInfo = list->GetFirst(); devInfo; devInfo = list->GetNext())
{
    if (std::string("Test camera") == devInfo->GetModel())
    {
        device = IpxCam::IpxCam_CreateDevice(devInfo);
        break;
    }
}
```

### 6.13.3.2 GetFirstDeviceInfo()

```
virtual DeviceInfo* IpxCam::Interface::GetFirstDeviceInfo ( ) [pure virtual]
```

This method retrieves the [DeviceInfo](#) object for the first device available on this [Interface](#).

#### Returns

returns the pointer to [DeviceInfo](#) object or nullptr if no device found

For example,

```
//Retrieve the first device available for the specified interface.
lDeviceInfo = iface->GetFirstDeviceInfo();

std::cout << "First Device Info ModelName" << lDeviceInfo->GetModel() << endl;
```

### 6.13.3.3 GetDeviceInfoById()

```
virtual DeviceInfo* IpxCam::Interface::GetDeviceInfoById (
    const char * deviceId ) [pure virtual]
```

This method retrieves the device info for the specified device identifier.

**Parameters**

in	<i>device↔ Id</i>	Device identifier
----	-----------------------	-------------------

**Returns**

returns the pointer to [DeviceInfo](#) object or nullptr if no such device found

**6.13.3.4 ReEnumerateDevices()**

```
virtual IpxCamErr IpxCam::Interface::ReEnumerateDevices (
    bool * pChanged,
    uint64_t iTimeout ) [pure virtual]
```

This method re-enumerates the devices.

The ReEnumerateDevices method allows the user to re-enumerate the devices connected to the [Interface](#) and update the DeviceInfoList object returned by subsequent [GetDeviceInfoList\(\)](#) method calls.

**Parameters**

in	<i>pChanged</i>	Change in <a href="#">Device</a>
in	<i>iTimeout</i>	Timeout allowed to search for available devices

**Returns**

returns error code

**6.13.3.5 GetDescription()**

```
virtual const char* IpxCam::Interface::GetDescription ( ) [pure virtual]
```

This method gets the description of the interface object.

Returns the user readable description of the interface

**Returns**

returns the Description of the interface

#### 6.13.3.6 GetType()

```
virtual InterfaceType IpxCam::Interface::GetType ( ) [pure virtual]
```

This method gets the type of interface object.

Returns the [Interface](#) Type (Transport Layer Technology) of this interface

##### Returns

returns [Interface](#) Type

The interface type return can be the following:

```
enum InterfaceType
{
    USB3Vision    = 1,
    GigEVision    = 2,
    CameraLink    = 3,
    CoaxPress     = 4,
    HdSdi         = 5,
    AllInterfaces = 0xff,
};
```

#### 6.13.3.7 GetId()

```
virtual const char* IpxCam::Interface::GetId ( ) [pure virtual]
```

This method gets the identifier of the interface object.

Returns the interface identifier that could be used to instantiate the interface

##### Returns

returns interface identifier

#### 6.13.3.8 GetVersion()

```
virtual const char* IpxCam::Interface::GetVersion ( ) [pure virtual]
```

This method gets the version of [Interface](#) driver.

Returns the pointer to the string with the version of the interface driver

##### Returns

returns the version of the interface driver

#### 6.13.3.9 RegisterEvent()

```
virtual IpxCamErr IpxCam::Interface::RegisterEvent (
    uint32_t eventType,
    IpxCam::EventCallback * eventCallback,
    void * pPrivate ) [pure virtual]
```

This method registers the [Interface](#) class method as a callback method to be called when a eventType occurs.

**Parameters**

in	<i>eventType</i>	Event Type TODO - define event ids here!!!!
in	<i>eventCallback</i>	pointer to event CallBack method
in	<i>pPrivate</i>	pointer to user's data

**Returns**

returns Error code

**6.13.3.10 UnRegisterEvent()**

```
virtual IpxCamErr IpxCam::Interface::UnRegisterEvent (
    uint32_t eventType,
    IpxCam::EventCallback * eventCallback,
    void * pPrivate ) [pure virtual]
```

This method unregisters the [Interface](#) class callback method for the eventType.

**Parameters**

in	<i>eventType</i>	Event Type
in	<i>eventCallback</i>	pointer to event CallBack method
in	<i>pPrivate</i>	pointer to user's data

**Returns**

returns Error code

**6.13.3.11 GetParameters()**

```
virtual IpxGenParam::Array* IpxCam::Interface::GetParameters (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method returns the parameter array used to control the Imperx Camera device.

**Parameters**

out	<i>err</i>	returns error code
-----	------------	--------------------

**Returns**

returns the parameter array used to control the Imperx Camera device

**6.13.3.12 CreateDeviceFromConfig()**

```
virtual Device* IpxCam::Interface::CreateDeviceFromConfig (
    const char * fileName,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method creates the device based on the information specified in the configuration file.

TODO make a description

**Parameters**

in	<i>fileName</i>	Configuration file to open
out	<i>err</i>	returns error code

**Returns**

returns [Device](#) or nullptr if device cannot be instantiated

The documentation for this class was generated from the following file:

- IpxCameraApi.h

**6.14 IpxCam::List<\_T> Class Template Reference**

The [List](#) class is used to list the specified template type objects.

```
#include <IpxCameraApi.h>
```

**Public Types**

- typedef [\\_T](#) [elem\\_type](#)



## Public Member Functions

- virtual `~List()`  
*A destructor of the `List` class.*
- virtual void `Release()`=0  
*This method releases the instance of the list of the specified object.*
- virtual `size_t GetCount()`=0  
*This functions gets the number of items in the specified list object.*
- virtual `elem_type * GetFirst()`=0  
*This method retrieves the first element in the specified list object.*
- virtual `elem_type * GetNext()`=0  
*This method retrieves the next element in the specified list object.*

### 6.14.1 Detailed Description

```
template<class _T>
class lpxCam::List<_T>
```

The `List` class is used to list the specified template type objects.

The supported template type objects are `Interface`, `Device`, `DeviceInfo`, `Stream`, and `Buffer`.

They can be declared as follows:

<code>lpxCam::List&lt;Interface&gt; *interfaceList</code>	This class represents the list of <code>Interface</code> objects.
<code>lpxCam::List&lt;Device&gt; *deviceList</code>	This class represents the list of <code>Device</code> objects.
<code>lpxCam::List&lt;DeviceInfo&gt; *deviceInfoList</code>	This class represents the list of <code>DeviceInfo</code> objects.
<code>lpxCam::List&lt;Stream&gt; *streamList</code>	This class represents the list of Data <code>Stream</code> objects
<code>lpxCam::List&lt;Buffer&gt; *bufferList</code>	This class represents the list of <code>Buffer</code> objects

Alternatively, you can also use the declared typedef(alikes for specific objects) provided in the `lpxCam` namespace as shown below:

```
typedef List<Interface>    InterfaceList;
typedef List<DeviceInfo>   DeviceInfoList;
typedef List<Device>       DeviceList;
```

They can be declared as follows:

<code>InterfaceList *interfaceList</code>	This class represents the list of <code>Interface</code> objects.
<code>DeviceInfoList *deviceInfoList</code>	This class represents the list of <code>DeviceInfo</code> objects.

This class can be used to search through the list of objects discovered.

### Example using DeviceInfoList

In this example, you will see how to use the DeviceInfoList. An example is shown below that demonstrates on how to use the list class methods. The **deviceInfoList->GetCount()** method is used retrieve the number of devices connected. We confirm that at least one device is available. Next, the for loop will loop from the first device information listed using the **deviceInfoList->GetFirst()** funtion to the end of the list. During each iteration the **deviceInfoList->GetNext()** will increment to the next deviceInfo available. In the example, you will notice that we search for a specified device model name. Once, the specified device is found, we will release the **deviceInfoList->Release()** and the create the specified device using the **IpxCam::IpxCam\_CreateDevice()** method.

```
// Get the Device Info List for the Interface
// List has to be released, let us use unique pointer
auto del = [](IpxCam::DeviceInfoList *l) { l->Release(); };
std::unique_ptr<IpxCam::DeviceInfoList, decltype(del)> list(iface->GetDeviceInfoList(), del);

if (list->GetCount() == 0)
{
    std::cout << "No Interface Available. " << endl;
    exit(1);
}

IpxCam::Device *device = nullptr;
for (auto devInfo = list->GetFirst(); devInfo; devInfo = list->GetNext())
{
    if (std::string("Test camera") == devInfo->GetModel())
    {
        device = IpxCam::IpxCam_CreateDevice(devInfo);
        break;
    }
}
```

### Example using InterfaceList

In this example, you will see how to use the InteraceList. You will retrieve the interfaces available for this system. Next, the for loop will loop from the first interface available using the **list->GetFirst()** method to the end of the list. During each iteration the **list->GetNext()** will increment to the next interface available.

```
// Used later to get chosen interface
std::vector<IpxCam::Interface*> ifaceVector;

// Get the Interface List for the System
auto list = system->GetInterfaceList();

// Get the individual Interface elements
for (auto iface = list->GetFirst(); iface; iface = list->GetNext())
{
    ifaceVector.push_back(iface);

    // Display the Interface Available
    std::cout << "[" << (ifaceVector.size() - 1) << "]" << "\t" << iface->GetDescription() << "Id " << iface
        ->GetId() << endl;
}

// List has to be released
list->Release();
```

## 6.14.2 Member Typedef Documentation

#### 6.14.2.1 elem\_type

```
template<class _T >
typedef _T IpxCam::List< _T >::elem_type
```

Element Type

### 6.14.3 Constructor & Destructor Documentation

#### 6.14.3.1 ~List()

```
template<class _T >
virtual IpxCam::List< _T >::~~List ( ) [inline], [virtual]
```

A destructor of the [List](#) class.

Destructor. Destroys the [List](#) and all its descendants.

### 6.14.4 Member Function Documentation

#### 6.14.4.1 Release()

```
template<class _T >
virtual void IpxCam::List< _T >::Release ( ) [pure virtual]
```

This method releases the instance of the list of the specified object.

##### Returns

Void.

#### 6.14.4.2 GetCount()

```
template<class _T >
virtual size_t IpxCam::List< _T >::GetCount ( ) [pure virtual]
```

This functions gets the number of items in the specified list object.

##### Returns

Returns the number of items in the specified list object.

#### 6.14.4.3 GetFirst()

```
template<class _T >
virtual elem_type* IpxCam::List< _T >::GetFirst ( ) [pure virtual]
```

This method retrieves the first element in the specified list object.

##### Returns

Returns the first element in the specified list object.

#### 6.14.4.4 GetNext()

```
template<class _T >
virtual elem_type* IpxCam::List< _T >::GetNext ( ) [pure virtual]
```

This method retrieves the next element in the specified list object.

##### Returns

Returns the next element in the specified list object.

The documentation for this class was generated from the following file:

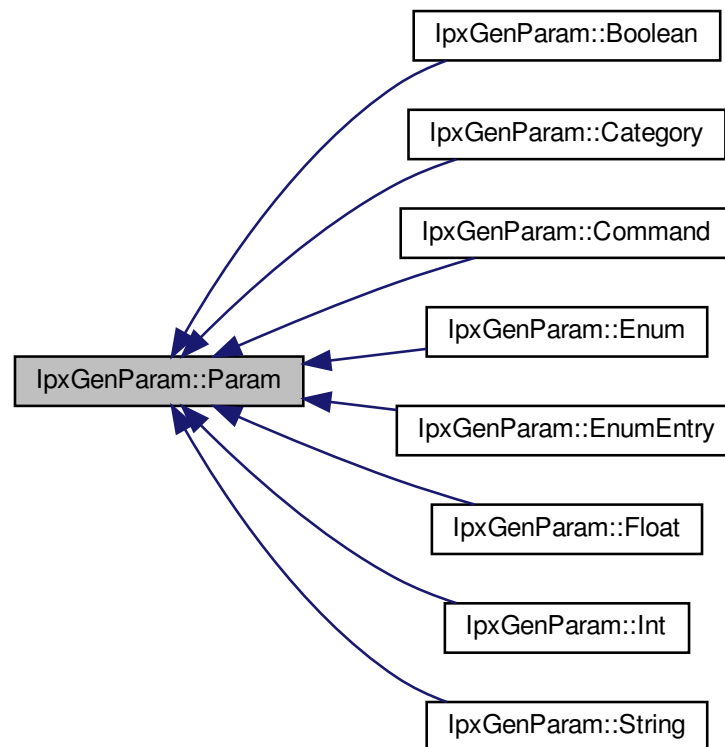
- IpxCameraApi.h

## 6.15 IpxGenParam::Param Class Reference

A Class for accessing the GenICam feature node parameters of the Camera Descriptor File.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::Param:



## Public Member Functions

- virtual `~Param()`  
*A destructor of the `Param` class.*
- virtual `ParamType GetType()`=0  
*This method returns the Parameter Node Type.*
- virtual const char \* `GetName()`=0  
*This method returns the parameter node name.*
- virtual const char \* `GetToolTip()`=0  
*This method returns a short description of the parameter node.*
- virtual const char \* `GetDescription()`=0  
*This method returns a long description of the parameter node.*
- virtual const char \* `GetDisplayName()`=0  
*This method returns the name of the display.*
- virtual `Visibility GetVisibility()`=0  
*This method returns the visibility of the node.*
- virtual bool `IsValueCached()`=0

- This method checks if the parameter node is cached.*

  - virtual bool [IsAvailable](#) ()=0

*This method checks if parameter node is available.*
- virtual bool [IsWritable](#) ()=0

*This method checks if parameter node is writeable.*
- virtual bool [IsReadable](#) ()=0

*This method checks if the parameter node is readable.*
- virtual bool [IsStreamable](#) ()=0

*This method checks if the parameter node is streamable.*
- virtual bool [IsVisible](#) ([Visibility](#) vis)=0

*This method checks if the node is visible.*
- virtual [IpxCamErr](#) [RegisterEventSink](#) ([ParamEventSink](#) \*aEventSink)=0

*This method registers the event.*
- virtual [IpxCamErr](#) [UnregisterEventSink](#) ([ParamEventSink](#) \*aEventSink)=0

*This method un-registers the event.*
- virtual [IPX\\_GENAPI\\_NS::INode](#) \* [GetNode](#) ()=0

*This method returns the callback of the node registered.*
- virtual [Category](#) \* [ToCategory](#) ()=0

*This method returns typed representation of param.*
- virtual [Boolean](#) \* [ToBoolean](#) ()=0

*This method returns typed representation of param.*
- virtual [Command](#) \* [ToCommand](#) ()=0

*This method returns typed representation of param.*
- virtual [EnumEntry](#) \* [ToEnumEntry](#) ()=0

*This method returns typed representation of param.*
- virtual [Enum](#) \* [ToEnum](#) ()=0

*This method returns typed representation of param.*
- virtual [Float](#) \* [ToFloat](#) ()=0

*This method returns typed representation of param.*
- virtual [Int](#) \* [ToInt](#) ()=0

*This method returns typed representation of param.*
- virtual [String](#) \* [ToString](#) ()=0

*This method returns typed representation of param.*

### 6.15.1 Detailed Description

A Class for accessing the GenICam feature node parameters of the Camera Descriptor File.

A Class for [Param](#) Properties.

### 6.15.2 Constructor & Destructor Documentation

#### 6.15.2.1 ~Param()

```
virtual IpxGenParam::Param::~~Param ( ) [inline], [virtual]
```

A destructor of the [Param](#) class.

Destructor. Destroys the [Param](#) and all its descendants.

### 6.15.3 Member Function Documentation

#### 6.15.3.1 GetType()

```
virtual ParamType IpxGenParam::Param::GetType ( ) [pure virtual]
```

This method returns the Parameter Node Type.

##### Returns

If the method succeeds, it will return the parameter type. Otherwise, it will return a nullptr.

Implemented in [IpxGenParam::String](#), [IpxGenParam::Int](#), [IpxGenParam::Float](#), [IpxGenParam::Enum](#), [IpxGenParam::EnumEntry](#), [IpxGenParam::Command](#), [IpxGenParam::Boolean](#), and [IpxGenParam::Category](#).

#### 6.15.3.2 GetName()

```
virtual const char* IpxGenParam::Param::GetName ( ) [pure virtual]
```

This method returns the parameter node name.

##### Returns

If the method succeeds, it will return the parameter node name. Otherwise, it will return a nullptr.

#### 6.15.3.3 GetToolTip()

```
virtual const char* IpxGenParam::Param::GetToolTip ( ) [pure virtual]
```

This method returns a short description of the parameter node.

##### Returns

If the method succeeds, it will return a short description of the parameter node. Otherwise, it will return a nullptr.

#### 6.15.3.4 GetDescription()

```
virtual const char* IpxGenParam::Param::GetDescription ( ) [pure virtual]
```

This method returns a long description of the parameter node.

##### Returns

If the method succeeds, it will return a long description of the parameter node. Otherwise, it will return a nullptr.

#### 6.15.3.5 GetDisplayName()

```
virtual const char* IpxGenParam::Param::GetDisplayName ( ) [pure virtual]
```

This method returns the name of the display.

##### Returns

If the method succeeds, it will return the name of the display. Otherwise, it will return a nullptr.

#### 6.15.3.6 GetVisibility()

```
virtual Visibility IpxGenParam::Param::GetVisibility ( ) [pure virtual]
```

This method returns the visibility of the node.

##### Returns

It will return the visibility setting of the parameter node. It will be either Basic, Expert, or Guru.

#### 6.15.3.7 IsValueCached()

```
virtual bool IpxGenParam::Param::IsValueCached ( ) [pure virtual]
```

This method checks if the parameter node is cached.

##### Returns

True if the value is cached. Otherwise, the value is not cached.



#### 6.15.3.8 IsAvailable()

```
virtual bool IpxGenParam::Param::IsAvailable ( ) [pure virtual]
```

This method checks if parameter node is available.

##### Returns

True if the parameter node is available. Otherwise, it is not available.

#### 6.15.3.9 IsWritable()

```
virtual bool IpxGenParam::Param::IsWritable ( ) [pure virtual]
```

This method checks if parameter node is writeable.

##### Returns

True if the parameter node is writeable. Otherwise, it is not writeable.

#### 6.15.3.10 IsReadable()

```
virtual bool IpxGenParam::Param::IsReadable ( ) [pure virtual]
```

This method checks if the parameter node is readable.

##### Returns

True if the parameter node is readable. Otherwise, it is not readable.

#### 6.15.3.11 IsStreamable()

```
virtual bool IpxGenParam::Param::IsStreamable ( ) [pure virtual]
```

This method checks if the parameter node is streamable.

##### Returns

True if the parameter node is streamable. Otherwise, it is not streamable.

#### 6.15.3.12 IsVisible()

```
virtual bool IpxGenParam::Param::IsVisible (
    Visibility vis ) [pure virtual]
```

This method checks if the node is visible.

**Parameters**

in	vis	Visibility of the parameter node
----	-----	----------------------------------

**Returns**

True if the parameter node is visible. Otherwise, it is not visible.

**6.15.3.13 RegisterEventSink()**

```
virtual IpxCamErr IpxCamParam::Param::RegisterEventSink (
    ParamEventSink * aEventSink ) [pure virtual]
```

This method registers the event.

**Parameters**

in	aEventSink	pointer to Parameter Event Sink
----	------------	---------------------------------

**Returns**

Returns the Error code:

- IpxCamErr::IPX\_CAM\_ERR\_OK - Successfully registers event sink

**6.15.3.14 UnregisterEventSink()**

```
virtual IpxCamErr IpxCamParam::Param::UnregisterEventSink (
    ParamEventSink * aEventSink ) [pure virtual]
```

This method un-registers the event.

**Parameters**

in	aEventSink	pointer to Paramet Event Sink
----	------------	-------------------------------

**Returns**

returns the Error code:

- IpxCamErr::IPX\_CAM\_ERR\_OK - Successfully unregisters event sink

#### 6.15.3.15 GetNode()

```
virtual IPX_GENAPI_NS::INode* IpxGenParam::Param::GetNode ( ) [pure virtual]
```

This method returns the callback of the node registered.

##### Returns

If the method succeeds, it will return the pointer to the node of the callback that is registered. Otherwise, it will return a value of nullptr.

#### 6.15.3.16 ToCategory()

```
virtual Category* IpxGenParam::Param::ToCategory ( ) [pure virtual]
```

This method returns typed representation of param.

##### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

#### 6.15.3.17 ToBoolean()

```
virtual Boolean* IpxGenParam::Param::ToBoolean ( ) [pure virtual]
```

This method returns typed representation of param.

##### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

#### 6.15.3.18 ToCommand()

```
virtual Command* IpxGenParam::Param::ToCommand ( ) [pure virtual]
```

This method returns typed representation of param.

##### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

#### 6.15.3.19 ToEnumEntry()

```
virtual EnumEntry* IpxGenParam::Param::ToEnumEntry ( ) [pure virtual]
```

This method returns typed representation of param.

##### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

#### 6.15.3.20 ToEnum()

```
virtual Enum* IpxGenParam::Param::ToEnum ( ) [pure virtual]
```

This method returns typed representation of param.

##### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

#### 6.15.3.21 ToFloat()

```
virtual Float* IpxGenParam::Param::ToFloat ( ) [pure virtual]
```

This method returns typed representation of param.

##### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

#### 6.15.3.22 ToInt()

```
virtual Int* IpxGenParam::Param::ToInt ( ) [pure virtual]
```

This method returns typed representation of param.

##### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

### 6.15.3.23 ToString()

```
virtual String* IpxGenParam::Param::ToString ( ) [pure virtual]
```

This method returns typed representation of param.

#### Returns

If the method succeeds, it will return pointer to typed param. Otherwise, it will return a value of nullptr

The documentation for this class was generated from the following file:

- IpxCameraApi.h

## 6.16 IpxGenParam::ParamEventSink Class Reference

An Event Sink class designed to receive incoming events from Parameter Node Updates.

```
#include <IpxCameraApi.h>
```

### Public Member Functions

- virtual [~ParamEventSink](#) ()  
*A destructor of the [ParamEventSink](#) class.*
- virtual void [OnParameterUpdate](#) ([Param](#) \*param)=0  
*Update Parameter Node.*

### 6.16.1 Detailed Description

An Event Sink class designed to receive incoming events from Parameter Node Updates.

A Class for [ParamEventSink](#).

### 6.16.2 Constructor & Destructor Documentation

#### 6.16.2.1 ~ParamEventSink()

```
virtual IpxGenParam::ParamEventSink::~~ParamEventSink ( ) [inline], [virtual]
```

A destructor of the [ParamEventSink](#) class.

Destructor. Destroys the [ParamEventSink](#) and all its descendants.

### 6.16.3 Member Function Documentation

#### 6.16.3.1 OnParameterUpdate()

```
virtual void IpxGenParam::ParamEventSink::OnParameterUpdate (  
    Param * param ) [pure virtual]
```

Update Parameter Node.

## Parameters

in	param	The pointer to the <a href="#">Param</a> class node return Void.
----	-------	--

The documentation for this class was generated from the following file:

- IpxCameraApi.h

## 6.17 IpxCam::Stream Class Reference

The [Stream](#) class represents the data stream module in the GenTL module hierarchy.

```
#include <IpxCameraApi.h>
```

### Public Member Functions

- virtual [~Stream](#) ()  
*A destructor of the [Stream](#) class.*
- virtual void [Release](#) ()=0  
*This method releases the instance of the stream object.*
- virtual [IpxCam::Buffer](#) \* [CreateBuffer](#) (size\_t iSize, void \*pPrivate, IpxCamErr \*err)=0  
*This method is used to create the buffer in the data stream object.*
- virtual [IpxCam::Buffer](#) \* [SetBuffer](#) (void \*pBuffer, size\_t iSize, void \*pPrivate, IpxCamErr \*err)=0  
*This method is used to set the buffer to allocate and announce to the data stream.*
- virtual IpxCamErr [RevokeBuffer](#) ([IpxCam::Buffer](#) \*hBuffer)=0  
*This method revokes any announced buffers.*
- virtual IpxCamErr [QueueBuffer](#) ([IpxCam::Buffer](#) \*hBuffer)=0  
*This method queues specified buffers.*
- virtual [IpxCam::Buffer](#) \* [GetBuffer](#) (uint64\_t iTimeout, IpxCamErr \*err=nullptr)=0  
*This method retrieves the buffer object.*
- virtual IpxCamErr [CancelBuffer](#) ()=0  
*This method cancels any previously registered buffer events that have been waiting to be performed.*
- virtual IpxCamErr [FlushBuffers](#) ([FlushOperation](#) operation)=0  
*This method flushes the buffers of the data stream object.*
- virtual IpxCamErr [StartAcquisition](#) (uint64\_t iNumFramesToAcquire=UINT64\_MAX, uint32\_t flags=0)=0  
*This method sends the start command to start the acquisition of image data.*
- virtual IpxCamErr [StopAcquisition](#) (uint32\_t flags=0)=0  
*This method sends the stop command to stop the acquisition of the any more image data.*
- virtual IpxCamErr [AllocBufferQueue](#) (void \*pPrivate, size\_t iNum)=0  
*This method allocates the buffer queue of the data stream object.*
- virtual IpxCamErr [ReleaseBufferQueue](#) ()=0  
*This method releases the buffer queue of the data stream object.*
- virtual size\_t [GetBufferQueueSize](#) ()=0  
*This functions get the buffer queue size of the data stream object.*

- virtual `IpxCamErr RegisterEvent` (`uint32_t eventType`, `IpxCam::EventCallback *eventCallback`, `void *pPrivate`)=0  
*This method registers the data [Stream](#) class method as a callback method to be called when a `eventType` occurs.*
- virtual `IpxCamErr UnRegisterEvent` (`uint32_t eventType`, `IpxCam::EventCallback *eventCallback`, `void *pPrivate`)=0  
*This method un-registers the data [Stream](#) class callback method for the `eventType`.*
- virtual `IpxGenParam::Array * GetParameters` (`IpxCamErr *err`=`nullptr`)=0  
*This method returns an array of parameters for the data stream object.*
- virtual `uint64_t GetNumDelivered` ()=0  
*This method returns the number of delivered buffers since the start of the last acquisition.*
- virtual `uint64_t GetNumUnderrun` ()=0  
*This method returns the number of lost frames due to the queue being underrun.*
- virtual `size_t GetNumAnnounced` ()=0  
*This method returns the number of announced buffers in the data stream object.*
- virtual `size_t GetNumQueued` ()=0  
*This method returns the number of queued buffers in the data stream object.*
- virtual `size_t GetNumAwaitDelivery` ()=0  
*This method returns the number of buffers awaiting delivery of the data stream object.*
- virtual `size_t GetBufferSize` ()=0  
*This method returns the buffer size of the data stream object.*
- virtual `bool IsGrabbing` ()=0  
*This method returns a flag indicating if the data stream is grabbing or not.*
- virtual `size_t GetMinNumBuffers` ()=0  
*This method returns the minimum number of buffers.*
- virtual `size_t GetBufferAlignment` ()=0  
*This method returns the alignment size of the stream object.*

### 6.17.1 Detailed Description

The [Stream](#) class represents the data stream module in the GenTL module hierarchy.

This data stream class provides buffer methods. This data stream class purpose is to access the buffer data acquirement from the Acquisition engine.

### 6.17.2 Constructor & Destructor Documentation

#### 6.17.2.1 `~Stream()`

```
virtual IpxCam::Stream::~Stream ( ) [inline], [virtual]
```

A destructor of the [Stream](#) class.

Destructor. Destroys the [Stream](#) and all its descendants.



### 6.17.3 Member Function Documentation

#### 6.17.3.1 Release()

```
virtual void IpxCam::Stream::Release ( ) [pure virtual]
```

This method releases the instance of the stream object.

##### Returns

Void. This method is used to release the stream object.

#### 6.17.3.2 CreateBuffer()

```
virtual IpxCam::Buffer* IpxCam::Stream::CreateBuffer (
    size_t iSize,
    void * pPrivate,
    IpxCamErr * err ) [pure virtual]
```

This method is used to create the buffer in the data stream object.

Allocates the memory for a buffer and announces this buffer to the data stream

##### Parameters

in	<i>iSize</i>	Size of the buffer
in	<i>pPrivate</i>	pointer to private data (user's data) which will be passed to the GenTL Consumer
out	<i>err</i>	returns Error code

##### Returns

returns [Buffer](#) object pointer of the announced buffer

#### 6.17.3.3 SetBuffer()

```
virtual IpxCam::Buffer* IpxCam::Stream::SetBuffer (
    void * pBuffer,
    size_t iSize,
```

```
void * pPrivate,  
IpxCamErr * err ) [pure virtual]
```

This method is used to set the buffer to allocate and announce to the data stream.

Sets buffers to allocate and announce to the data stream

#### Parameters

in	<i>pBuffer</i>	buffer
in	<i>iSize</i>	size of <a href="#">Buffer</a>
in	<i>pPrivate</i>	pointer to user's data
out	<i>err</i>	returns Error code

#### Returns

returns [Buffer](#) object pointer

#### 6.17.3.4 RevokeBuffer()

```
virtual IpxCamErr IpxCam::Stream::RevokeBuffer (  
    IpxCam::Buffer * hBuffer ) [pure virtual]
```

This method revokes any announced buffers.

This method removes the specified announced buffer from the acquisition engine's queue

#### Parameters

in	<i>hBuffer</i>	handle of buffer
----	----------------	------------------

#### Returns

returns Error code

#### 6.17.3.5 QueueBuffer()

```
virtual IpxCamErr IpxCam::Stream::QueueBuffer (  
    IpxCam::Buffer * hBuffer ) [pure virtual]
```

This method queues specified buffers.

During acquisition, this method queues the specified buffer.

**Parameters**

in	<i>hBuffer</i>	handle of buffer
----	----------------	------------------

**Returns**

returns Error code

**6.17.3.6 GetBuffer()**

```
virtual IpxCam::Buffer* IpxCam::Stream::GetBuffer (
    uint64_t iTimeout,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method retrieves the buffer object.

Retrieve the next event data entry from the event data queued

**Parameters**

in	<i>iTimeout</i>	timeout in ms
in	<i>err</i>	error code

**Returns**

returns the pointer of the buffer

**6.17.3.7 CancelBuffer()**

```
virtual IpxCamErr IpxCam::Stream::CancelBuffer ( ) [pure virtual]
```

This method cancels any previously registered buffer events that have been waiting to be performed.

Terminates the waiting operation on a previously registered [Buffer](#) Event

**Returns**

returns Error code

### 6.17.3.8 FlushBuffers()

```
virtual IpxCamErr IpxCam::Stream::FlushBuffers (
    FlushOperation operation ) [pure virtual]
```

This method flushes the buffers of the data stream object.

Flushes the Flush Operation specified internal buffer pool or queue. Operations defined in FlushOperations enum.

**Parameters**

in	<i>operation</i>	FlushOperation
----	------------------	----------------

**Returns**

returns Error code

**6.17.3.9 StartAcquisition()**

```
virtual IpxCamErr IpxCam::Stream::StartAcquisition (
    uint64_t iNumFramesToAcquire = UINT64_MAX,
    uint32_t flags = 0 ) [pure virtual]
```

This method sends the start command to start the acquisition of image data.

Starts the Acquisition Engine

**Parameters**

in	<i>iNumFramesToAcquire</i>	number of Frames to Acquire
in	<i>flags</i>	flags

**Returns**

returns Error code

**6.17.3.10 StopAcquisition()**

```
virtual IpxCamErr IpxCam::Stream::StopAcquisition (
    uint32_t flags = 0 ) [pure virtual]
```

This method sends the stop command to stop the acquisition of the any more image data.

The acquisition on the remote device is stopped after finishing acquiring image data

**Parameters**

in	<i>flags</i>	flags
----	--------------	-------

**Returns**

returns Error code

**6.17.3.11 AllocBufferQueue()**

```
virtual IpxCamErr IpxCam::Stream::AllocBufferQueue (
    void * pPrivate,
    size_t iNum ) [pure virtual]
```

This method allocates the buffer queue of the data stream object.

**Parameters**

in	<i>pPrivate</i>	pointer to user's data
in	<i>iNum</i>	number of Buffers to allocate

**Returns**

returns Error code

**6.17.3.12 ReleaseBufferQueue()**

```
virtual IpxCamErr IpxCam::Stream::ReleaseBufferQueue ( ) [pure virtual]
```

This method releases the buffer queue of the data stream object.

Release the [Buffer](#) Queue

**Returns**

returns Error code

**6.17.3.13 GetBufferQueueSize()**

```
virtual size_t IpxCam::Stream::GetBufferQueueSize ( ) [pure virtual]
```

This functions get the buffer queue size of the data stream object.

**Returns**

returns the [Buffer](#) Queue size

## 6.17.3.14 RegisterEvent()

```
virtual IpxCamErr IpxCam::Stream::RegisterEvent (
    uint32_t eventType,
    IpxCam::EventCallback * eventCallback,
    void * pPrivate ) [pure virtual]
```

This method registers the data [Stream](#) class method as a callback method to be called when a eventType occurs.

## Parameters

in	<i>eventType</i>	Event Type TODO - define event type ids here!!!!
in	<i>eventCallback</i>	event CallBack
in	<i>pPrivate</i>	pointer to user's data

## Returns

returns Error code

## 6.17.3.15 UnRegisterEvent()

```
virtual IpxCamErr IpxCam::Stream::UnRegisterEvent (
    uint32_t eventType,
    IpxCam::EventCallback * eventCallback,
    void * pPrivate ) [pure virtual]
```

This method un-registers the data [Stream](#) class callback method for the eventType.

## Parameters

in	<i>eventType</i>	Event Type
in	<i>eventCallback</i>	event CallBack
in	<i>pPrivate</i>	pointer to user's data

## Returns

returns Error code

## 6.17.3.16 GetParameters()

```
virtual IpxGenParam::Array* IpxCam::Stream::GetParameters (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method returns an array of parameters for the data stream object.

**Parameters**

out	err	returns the error code
-----	-----	------------------------

**Returns**

returns the data stream parameters array

**6.17.3.17 GetNumDelivered()**

```
virtual uint64_t IpxCam::Stream::GetNumDelivered ( ) [pure virtual]
```

This method returns the number of delivered buffers since the start of the last acquisition.

**Returns**

returns the number of delivered buffers since the start of the last acquisition

**6.17.3.18 GetNumUnderrun()**

```
virtual uint64_t IpxCam::Stream::GetNumUnderrun ( ) [pure virtual]
```

This method returns the number of lost frames due to the queue being underrun.

**Returns**

returns the number of lost frames due to queue underrun

**6.17.3.19 GetNumAnnounced()**

```
virtual size_t IpxCam::Stream::GetNumAnnounced ( ) [pure virtual]
```

This method returns the number of announced buffers in the data stream object.

**Returns**

returns number of announced buffers



#### 6.17.3.20 GetNumQueued()

```
virtual size_t IpxCam::Stream::GetNumQueued ( ) [pure virtual]
```

This method returns the number of queued buffers in the data stream object.

##### Returns

returns the number of buffers in the input pool and the number of buffers currently being filled

#### 6.17.3.21 GetNumAwaitDelivery()

```
virtual size_t IpxCam::Stream::GetNumAwaitDelivery ( ) [pure virtual]
```

This method returns the number of buffers awaiting delivery of the data stream object.

##### Returns

returns the number of buffers in the output buffer queue

#### 6.17.3.22 GetBufferSize()

```
virtual size_t IpxCam::Stream::GetBufferSize ( ) [pure virtual]
```

This method returns the buffer size of the data stream object.

##### Returns

returns the buffer size

#### 6.17.3.23 IsGrabbing()

```
virtual bool IpxCam::Stream::IsGrabbing ( ) [pure virtual]
```

This method returns a flag indicating if the data stream is grabbing or not.

##### Returns

Flag indicating the state of the acquisition engine. If true, acquisition engine has started. Otherwise, the acquisition engine is off.

#### 6.17.3.24 GetMinNumBuffers()

```
virtual size_t IpxCam::Stream::GetMinNumBuffers ( ) [pure virtual]
```

This method returns the minimum number of buffers.

##### Returns

returns the minimum number of buffers to announce

#### 6.17.3.25 GetBufferAlignment()

```
virtual size_t IpxCam::Stream::GetBufferAlignment ( ) [pure virtual]
```

This method returns the alignment size of the stream object.

##### Returns

returns the alignment size in bytes of the buffer passed

The documentation for this class was generated from the following file:

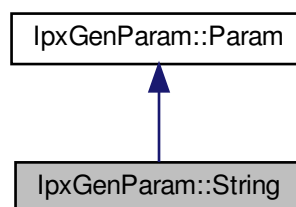
- IpxCameraApi.h

## 6.18 IpxGenParam::String Class Reference

Interface Class for GenICam [String](#) properties.

```
#include <IpxCameraApi.h>
```

Inheritance diagram for IpxGenParam::String:



## Public Member Functions

- virtual [ParamType](#) [GetType](#) ()  
*This method returns the node object [Command](#) type.*
- virtual [size\\_t](#) [GetMaxLength](#) ([IpxCamErr](#) \*err=nullptr)=0  
*This method gets the Maximum Length of the string.*
- virtual const char \* [GetValue](#) ([size\\_t](#) \*len=nullptr, [IpxCamErr](#) \*err=nullptr)=0  
*This method gets the value of the string node.*
- virtual [IpxCamErr](#) [SetValue](#) (const char \*val)=0  
*This method sets the value of the string node.*

### 6.18.1 Detailed Description

Interface Class for GenICam [String](#) properties.

A GenICam [String](#) class. For example, mapping to an edit box showing a string

### 6.18.2 Member Function Documentation

#### 6.18.2.1 [GetType](#)()

```
virtual ParamType IpxGenParam::String::GetType ( ) [inline], [virtual]
```

This method returns the node object [Command](#) type.

#### Returns

The parameter type

Implements [IpxGenParam::Param](#).

#### 6.18.2.2 [GetMaxLength](#)()

```
virtual size\_t IpxGenParam::String::GetMaxLength (
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the Maximum Length of the string.

**Parameters**

out	err	returns error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the maximum length value</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>
-----	-----	--

**Returns**

gets the maximum length of the string

**6.18.2.3 GetValue()**

```
virtual const char* IpxGenParam::String::GetValue (
    size_t * len = nullptr,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method gets the value of the string node.

**Parameters**

out	len	return the length of the string
out	err	returns the error code: <ul style="list-style-type: none"> <li>• <code>IpxCamErr::IPX_CAM_ERR_OK</code> - Successfully gets the string</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR</code> - Unable to access genicam specified node</li> <li>• <code>IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR</code> - Unable to access genicam specified node type</li> </ul>

**Returns**

returns the value

**6.18.2.4 SetValue()**

```
virtual IpxCamErr IpxGenParam::String::SetValue (
    const char * val ) [pure virtual]
```

This method sets the value of the string node.

## Parameters

in	val	Set the value of the string node
----	-----	----------------------------------

## Returns

returns the error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully sets the string
- `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node
- `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type

The documentation for this class was generated from the following file:

- `IpxCameraApi.h`

## 6.19 IpxCam::System Class Reference

The [System](#) class represents an abstraction of the system module of the GenTL module hierarchy. The [System](#) class is the entry point to the GenTL Producer software driver.

```
#include <IpxCameraApi.h>
```

### Public Member Functions

- virtual [~System](#) ()  
*A destructor of the [System](#) class.*
- virtual void [Release](#) ()=0  
*This method releases the instance of the system object.*
- virtual [InterfaceList](#) \* [GetInterfaceList](#) ([InterfaceType](#) type=[AllInterfaces](#))=0  
*This method returns all the interface list of the system object.*
- virtual [Interface](#) \* [GetInterfaceById](#) (const char \*ifaceld)=0  
*This method returns the interface by unique string identifier of the system object.*
- virtual const char \* [GetDisplayName](#) ()=0  
*This method displays the name of the GenTL Producer.*
- virtual const char \* [GetVersion](#) ()=0  
*This method returns the version of the GenTL Producer of the system object.*
- virtual [Device](#) \* [CreateDeviceFromConfig](#) (const char \*fileName, [IpxCamErr](#) \*err=nullptr)=0  
*This method configures and sets up the device using the information retrieved from the configuration file.*
- virtual [IpxCamErr](#) [RegisterGenTLProvider](#) (const char \*fileName)=0  
*This method registers the 3rd party GenTL provider CTI library in the [System](#).*

### 6.19.1 Detailed Description

The [System](#) class represents an abstraction of the system module of the GenTL module hierarchy. The [System](#) class is the entry point to the GenTL Producer software driver.

This class provides member functions to enumerate and instantiate the available interfaces reachable. It also provides a method for the configuration of the device module. This system module is the root of the GenTL Module hierarchy. [IpxCam::System](#) class has member functions to find all the interfaces, display the user readable name and producer version of the GenTL system. The [IpxCam::System](#) class can return [IpxCam::InterfaceList](#), [IpxCam::Interface](#), and [IpxCam::Device](#) objects.

The following is an example on how to use some of the public Member Functions.

```
//Get System
IpxCam::System *system = IpxCam::IpxCam_GetSystem();
IpxCam::DeviceInfo *lDeviceInfo = nullptr;

if (system)
{
    //Retrieve the System Name
    const char* displayname_str = system->GetDisplayName();
    std::cout << "DisplayName " << displayname_str;

    //Retrieve the Version of the System
    const char* verion_str = system->GetVersion();
    std::cout << "Version " << system->GetVersion();

    IpxCam::Interface *iface = nullptr;
    IpxCam::Interface *iface2 = nullptr;
    std::cout << "Interfaces Available:" << endl;

    std::vector<IpxCam::Interface*> ifaceVector;

    //Get the Interface List for the System
    IpxCam::InterfaceList* list = system->GetInterfaceList();
    for(IpxCam::List<IpxCam::Interface*>::elem_type* iface = list
        ->GetFirst(); iface; iface = list->GetNext())
    {
        ifaceVector.push_back(iface);
        //Display the Interface Available
        std::cout << "[" << (ifaceVector.size() - 1) << "]" << "\t" << iface->
            GetDescription() << "Id " << iface->GetId() << endl;
    }

    //List the number of Interfaces in the System
    std::cout << "Number of Interfaces in the System: " << list->GetCount() << endl;

    //Example of sending Interface By Id
    iface2 = system->GetInterfaceById(ifaceVector[0]->GetId());

    std::cout << "Interface Description: " << iface2->GetDescription() << endl;
    lDeviceInfo = iface2->GetFirstDeviceInfo();
    std::cout << "ModelName" << lDeviceInfo->GetModel() << endl;

    std::cout << "Releasing system" << endl;
    list->Release();
    system->Release();
}
```

### 6.19.2 Constructor & Destructor Documentation

### 6.19.2.1 ~System()

```
virtual IpxCam::System::~~System ( ) [inline], [virtual]
```

A destructor of the [System](#) class.

Destructor. Destroys the [System](#) and all its descendants. Here is the call graph for this function:



## 6.19.3 Member Function Documentation

### 6.19.3.1 Release()

```
virtual void IpxCam::System::Release ( ) [pure virtual]
```

This method releases the instance of the system object.

#### Returns

Void.

The following shows an example on how to use the **Release** method to release the system object instantiated.

```
//Get the GenTL System
IpxCam::System *system = IpxCam::IpxCam_GetSystem();

if (system)
{
    //Add Code Here

    //Release the GenTL System
    system->Release();
}
```

### 6.19.3.2 GetInterfaceList()

```
virtual InterfaceList* IpxCam::System::GetInterfaceList (
    InterfaceType type = AllInterfaces ) [pure virtual]
```

This method returns all the interface list of the system object.

It lists all the available hardware interfaces with the transport layers technologies that are supported.



**Parameters**

in	<i>type</i>	interface type
----	-------------	----------------

**Returns**

returns the interface list

The following is an example on how to use the **GetInterfaceList** method.

```
// Used later to get chosen interface
std::vector<IpxCam::Interface*> ifaceVector;

// Get the Interface List for the System
auto list = system->GetInterfaceList();

// Get the individual Interface elements
for (auto iface = list->GetFirst(); iface; iface = list->GetNext())
{
    ifaceVector.push_back(iface);

    // Display the Interface Available
    std::cout << "[" << (ifaceVector.size() - 1) << "]" << "\t" << iface->
        GetDescription() << "Id " << iface->GetId() << endl;
}

// List has to be released
list->Release();
```

**6.19.3.3 GetInterfaceById()**

```
virtual Interface* IpxCam::System::GetInterfaceById (
    const char * ifaceId ) [pure virtual]
```

This method returns the interface by unique string identifier of the system object.

Get interface specified by interface identifier.

**Parameters**

in	<i>ifaceId</i>	Interface identifier
----	----------------	----------------------

**Returns**

returns the **Interface** or nullptr if no such interface is found

For example, the const char \*ifaceId interface identification name could be as shown below:

```
//Get System
IpxCam::System *system = IpxCam::IpxCam_GetSystem();

const char *ifaceId = "\\?\\u
sb#vid_20f7&pid_30b3&mi_00#7&16f3afad&0&0000#{ff958afd-fce7-4264-994c-8fa230d5a524}";

auto iface = system->GetInterface(ifaceId);
```

This method will retrieve the available interface list of the system.

#### 6.19.3.4 GetDisplayName()

```
virtual const char* IpxCam::System::GetDisplayName ( ) [pure virtual]
```

This method displays the name of the GenTL Producer.

This method returns the User readable name of the GenTL Producer of the system object.

##### Returns

returns the Display Name

The following is an example on how to use the GetDisplayName method

```
//Get System
IpxCam::System *system = IpxCam::IpxCam_GetSystem();

if (system)
{
    //Retrieve the System Name
    const char* displayname_str = system->GetDisplayName();
    std::cout << "DisplayName " << displayname_str;

    // some code here

    system->Release();
}
```

#### 6.19.3.5 GetVersion()

```
virtual const char* IpxCam::System::GetVersion ( ) [pure virtual]
```

This method returns the version of the GenTL Producer of the system object.

This method returns the GenTL Producer version.

##### Returns

returns the Version

The following is an example on how to use the GetVersion method

```
//Get System
IpxCam::System *system = IpxCam::IpxCam_GetSystem();

if (system)
{
    //Retrieve the Version of the System
    const char* verion_str = system->GetVersion();
    std::cout << "Version " << system->GetVersion();

    // some code here

    system->Release();
}
```

### 6.19.3.6 CreateDeviceFromConfig()

```
virtual Device* IpxCam::System::CreateDeviceFromConfig (
    const char * fileName,
    IpxCamErr * err = nullptr ) [pure virtual]
```

This method configures and sets up the device using the information retrieved from the configuration file.

Creates the [Device](#) object from configuraion file

#### Parameters

in	<i>fileName</i>	Configuration file to open
out	<i>err</i>	returns the error code

#### Returns

returns [Device](#) or nullptr if device cannot be instantiated

### 6.19.3.7 RegisterGenTLProvider()

```
virtual IpxCamErr IpxCam::System::RegisterGenTLProvider (
    const char * fileName ) [pure virtual]
```

This method registers the 3rd party GenTL provider CTI library in the [System](#).

Registers the GenTL CTI library

#### Parameters

in	<i>fileName</i>	path to GenTL CTI file to add
----	-----------------	-------------------------------

#### Returns

returns the error code

The documentation for this class was generated from the following file:

- IpxCameraApi.h

## 6.20 IpxCam::Device::UploadEventData Struct Reference

A structure representing data for uploading to a device.

```
#include <IpxCameraApi.h>
```

### 6.20.1 Detailed Description

A structure representing data for uploading to a device.

The documentation for this struct was generated from the following file:

- `lpxCameraApi.h`

# Index

- ~Array
  - IpxGenParam::Array, [31](#)
- ~Buffer
  - IpxCam::Buffer, [48](#)
- ~Device
  - IpxCam::Device, [60](#)
- ~DeviceInfo
  - IpxCam::DeviceInfo, [67](#)
- ~IpxGenParamTreeView
  - IpxGui::IpxGenParamTreeView, [84](#)
- ~Interface
  - IpxCam::Interface, [93](#)
- ~List
  - IpxCam::List, [101](#)
- ~Param
  - IpxGenParam::Param, [104](#)
- ~ParamEventSink
  - IpxGenParam::ParamEventSink, [111](#)
- ~Stream
  - IpxCam::Stream, [114](#)
- ~System
  - IpxCam::System, [129](#)
- AllocBufferQueue
  - IpxCam::Stream, [120](#)
- CancelBuffer
  - IpxCam::Stream, [117](#)
- clearParams
  - IpxGui::IpxGenParamTreeView, [86](#)
- CreateBuffer
  - IpxCam::Stream, [115](#)
- CreateDeviceFromConfig
  - IpxCam::Interface, [98](#)
  - IpxCam::System, [132](#)
- CreateGenParamTreeViewForArrayA
  - IpxGui, [17](#)
- CreateGenParamTreeViewForArrayW
  - IpxGui, [18](#)
- CreateGenParamTreeViewForNodemapA
  - IpxGui, [19](#)
- CreateGenParamTreeViewForNodemapW
  - IpxGui, [20](#)
- DestroyGenParamTreeView
  - IpxGui, [21](#)

- DeviceAccess
  - IpxCam, [12](#)
- DeviceInfoList
  - IpxCam, [10](#)
- DeviceList
  - IpxCam, [10](#)
- elem\_type
  - IpxCam::List, [100](#)
- Endianness
  - IpxCam::Device, [59](#)
- EventCallback2
  - IpxCam, [11](#)
- Execute
  - IpxGenParam::Command, [56](#)
- ExecuteCommand
  - IpxGenParam::Array, [43](#)
- FlushBuffers
  - IpxCam::Stream, [117](#)
- FlushOperation
  - IpxCam, [11](#)
- ForceIP
  - IpxCam::DeviceInfo, [70](#)
- GetAccessStatus
  - IpxCam::DeviceInfo, [69](#)
- GetBoolean
  - IpxGenParam::Array, [31](#)
- GetBooleanValue
  - IpxGenParam::Array, [37](#)
- GetBuffer
  - IpxCam::Stream, [117](#)
- GetBufferAlignment
  - IpxCam::Stream, [124](#)
- GetBufferPtr
  - IpxCam::Buffer, [48](#)
- GetBufferQueueSize
  - IpxCam::Stream, [120](#)
- GetBufferSize
  - IpxCam::Buffer, [49](#)
  - IpxCam::Stream, [123](#)
- GetCameraParameters
  - IpxCam::Device, [64](#)
- GetCommand
  - IpxGenParam::Array, [32](#)

- GetCount
  - IpxCam::List, [101](#)
  - IpxGenParam::Array, [36](#)
  - IpxGenParam::Category, [54](#)
- GetDeliveredHeight
  - IpxCam::Buffer, [52](#)
- GetDescription
  - IpxCam::Interface, [95](#)
  - IpxGenParam::Param, [105](#)
- GetDeviceInfoById
  - IpxCam::Interface, [94](#)
- GetDeviceInfoList
  - IpxCam::Interface, [93](#)
- GetDisplayName
  - IpxCam::DeviceInfo, [68](#)
  - IpxCam::System, [132](#)
  - IpxGenParam::Param, [106](#)
- GetEndianness
  - IpxCam::Device, [65](#)
- GetEnum
  - IpxGenParam::Array, [33](#)
- GetEnumEntriesCount
  - IpxGenParam::Enum, [72](#)
- GetEnumEntryByIndex
  - IpxGenParam::Enum, [73](#)
- GetEnumEntryByName
  - IpxGenParam::Enum, [73](#)
- GetEnumEntryByValue
  - IpxGenParam::Enum, [74](#)
- GetEnumValue
  - IpxGenParam::Array, [39](#)
- GetEnumValueStr
  - IpxGenParam::Array, [38](#)
- GetFirst
  - IpxCam::List, [101](#)
- GetFirstDeviceInfo
  - IpxCam::Interface, [94](#)
- GetFloat
  - IpxGenParam::Array, [33](#)
- GetFloatValue
  - IpxGenParam::Array, [40](#)
- GetFrameID
  - IpxCam::Buffer, [50](#)
- GetHeight
  - IpxCam::Buffer, [50](#)
- GetID
  - IpxCam::DeviceInfo, [67](#)
- GetId
  - IpxCam::Interface, [96](#)
- GetImage
  - IpxCam::Buffer, [48](#)
- GetImageOffset
  - IpxCam::Buffer, [48](#)
- GetIncrement
  - IpxGenParam::Int, [91](#)
- GetInfo
  - IpxCam::Device, [61](#)
- GetInt
  - IpxGenParam::Array, [34](#)
- GetIntegerValue
  - IpxGenParam::Array, [41](#)
- GetInterface
  - IpxCam::DeviceInfo, [67](#)
- GetInterfaceById
  - IpxCam::System, [131](#)
- GetInterfaceList
  - IpxCam::System, [130](#)
- GetMax
  - IpxGenParam::Float, [81](#)
  - IpxGenParam::Int, [91](#)
- GetMaxLength
  - IpxGenParam::String, [125](#)
- GetMin
  - IpxGenParam::Float, [81](#)
  - IpxGenParam::Int, [90](#)
- GetMinNumBuffers
  - IpxCam::Stream, [123](#)
- GetModel
  - IpxCam::DeviceInfo, [68](#)
- GetName
  - IpxGenParam::Param, [105](#)
- GetNext
  - IpxCam::List, [102](#)
- GetNode
  - IpxGenParam::Param, [108](#)
- GetNodeMap
  - IpxGenParam::Array, [35](#)
- GetNumAnnounced
  - IpxCam::Stream, [122](#)
- GetNumAwaitDelivery
  - IpxCam::Stream, [123](#)
- GetNumDelivered
  - IpxCam::Stream, [122](#)
- GetNumQueued
  - IpxCam::Stream, [122](#)
- GetNumStreams
  - IpxCam::Device, [60](#)
- GetNumUnderrun
  - IpxCam::Stream, [122](#)
- GetParam
  - IpxGenParam::Array, [31](#)
- GetParamByIndex
  - IpxGenParam::Array, [36](#)
  - IpxGenParam::Category, [54](#)
- GetParameters
  - IpxCam::Interface, [97](#)
  - IpxCam::Stream, [121](#)
- GetPixelFormat

- IpxCam::Buffer, 49
- getPollingTime
  - IpxGui::IpxGenParamTreeView, 88
- GetRootCategory
  - IpxGenParam::Array, 35
- GetSerialNumber
  - IpxCam::DeviceInfo, 68
- GetStreamById
  - IpxCam::Device, 61
- GetStreamByIndex
  - IpxCam::Device, 60
- GetString
  - IpxGenParam::Array, 34
- GetStringValue
  - IpxGenParam::Array, 42
- GetTimestamp
  - IpxCam::Buffer, 49
- GetToolTip
  - IpxGenParam::Param, 105
- GetTransportParameters
  - IpxCam::Device, 64
- GetType
  - IpxCam::Interface, 95
  - IpxGenParam::Boolean, 45
  - IpxGenParam::Category, 53
  - IpxGenParam::Command, 56
  - IpxGenParam::Enum, 72
  - IpxGenParam::EnumEntry, 77
  - IpxGenParam::Float, 79
  - IpxGenParam::Int, 89
  - IpxGenParam::Param, 105
  - IpxGenParam::String, 125
- GetUSB3HostInfo
  - IpxCam::DeviceInfo, 69
- GetUnit
  - IpxGenParam::Float, 82
- GetUserDefinedName
  - IpxCam::DeviceInfo, 68
- GetUserPtr
  - IpxCam::Buffer, 49
- GetValue
  - IpxGenParam::Boolean, 46
  - IpxGenParam::Enum, 74
  - IpxGenParam::EnumEntry, 77
  - IpxGenParam::Float, 80
  - IpxGenParam::Int, 90
  - IpxGenParam::String, 126
- GetValueStr
  - IpxGenParam::Enum, 74
  - IpxGenParam::EnumEntry, 78
- GetVendor
  - IpxCam::DeviceInfo, 67
- GetVersion
  - IpxCam::DeviceInfo, 69
- IpxCam::Interface, 96
  - IpxCam::System, 132
- GetVisibility
  - IpxGenParam::Param, 106
- GetWidth
  - IpxCam::Buffer, 50
- GetXOffset
  - IpxCam::Buffer, 51
- GetXPadding
  - IpxCam::Buffer, 51
- GetYOffset
  - IpxCam::Buffer, 51
- GetYPadding
  - IpxCam::Buffer, 51
- InterfaceList
  - IpxCam, 10
- InterfaceType
  - IpxCam, 11
- IpxCam, 9
  - DeviceAccess, 12
  - DeviceInfoList, 10
  - DeviceList, 10
  - EventCallback2, 11
  - FlushOperation, 11
  - InterfaceList, 10
  - InterfaceType, 11
  - IpxCam\_GetSystem, 12
  - ServiceFileType, 12
- IpxCam::Buffer, 46
  - ~Buffer, 48
  - GetBufferPtr, 48
  - GetBufferSize, 49
  - GetDeliveredHeight, 52
  - GetFrameID, 50
  - GetHeight, 50
  - GetImage, 48
  - GetImageOffset, 48
  - GetPixelFormat, 49
  - GetTimestamp, 49
  - GetUserPtr, 49
  - GetWidth, 50
  - GetXOffset, 51
  - GetXPadding, 51
  - GetYOffset, 51
  - GetYPadding, 51
  - IsIncomplete, 50
  - IsKacFrameB, 52
- IpxCam::Device, 57
  - ~Device, 60
  - Endianness, 59
  - GetCameraParameters, 64
  - GetEndianness, 65
  - GetInfo, 61

- GetNumStreams, 60
- GetStreamById, 61
- GetStreamByIndex, 60
- GetTransportParameters, 64
- LoadConfiguration, 65
- ReadMem, 61
- RegisterEvent, 63
- SaveConfiguration, 65
- UnRegisterEvent, 63
- UploadEventCallback, 59
- UploadEventType, 59
- UploadFile, 62
- WriteMem, 62
- IpxCam::Device::UploadEventData, 133
- IpxCam::DeviceInfo, 66
  - ~DeviceInfo, 67
  - ForceIP, 70
  - GetAccessStatus, 69
  - GetDisplayName, 68
  - GetID, 67
  - GetInterface, 67
  - GetModel, 68
  - GetSerialNumber, 68
  - GetUSB3HostInfo, 69
  - GetUserDefinedName, 68
  - GetVendor, 67
  - GetVersion, 69
- IpxCam::Interface, 92
  - ~Interface, 93
  - CreateDeviceFromConfig, 98
  - GetDescription, 95
  - GetDeviceInfoById, 94
  - GetDeviceInfoList, 93
  - GetFirstDeviceInfo, 94
  - GetId, 96
  - GetParameters, 97
  - GetType, 95
  - GetVersion, 96
  - ReEnumerateDevices, 95
  - RegisterEvent, 96
  - UnRegisterEvent, 97
- IpxCam::List
  - ~List, 101
  - elem\_type, 100
  - GetCount, 101
  - GetFirst, 101
  - GetNext, 102
  - Release, 101
- IpxCam::List<\_T>, 98
- IpxCam::Stream, 113
  - ~Stream, 114
  - AllocBufferQueue, 120
  - CancelBuffer, 117
  - CreateBuffer, 115
  - FlushBuffers, 117
  - GetBuffer, 117
  - GetBufferAlignment, 124
  - GetBufferQueueSize, 120
  - GetBufferSize, 123
  - GetMinNumBuffers, 123
  - GetNumAnnounced, 122
  - GetNumAwaitDelivery, 123
  - GetNumDelivered, 122
  - GetNumQueued, 122
  - GetNumUnderrun, 122
  - GetParameters, 121
  - IsGrabbing, 123
  - QueueBuffer, 116
  - RegisterEvent, 120
  - Release, 115
  - ReleaseBufferQueue, 120
  - RevokeBuffer, 116
  - SetBuffer, 115
  - StartAcquisition, 119
  - StopAcquisition, 119
  - UnRegisterEvent, 121
- IpxCam::System, 128
  - ~System, 129
  - CreateDeviceFromConfig, 132
  - GetDisplayName, 132
  - GetInterfaceById, 131
  - GetInterfaceList, 130
  - GetVersion, 132
  - RegisterGenTLProvider, 133
  - Release, 130
- IpxCam\_GetSystem
  - IpxCam, 12
- IpxGenParam, 13
  - Namespace, 14
  - ParamType, 14
  - Visibility, 15
- IpxGenParam::Array, 29
  - ~Array, 31
  - ExecuteCommand, 43
  - GetBoolean, 31
  - GetBooleanValue, 37
  - GetCommand, 32
  - GetCount, 36
  - GetEnum, 33
  - GetEnumValue, 39
  - GetEnumValueStr, 38
  - GetFloat, 33
  - GetFloatValue, 40
  - GetInt, 34
  - GetIntegerValue, 41
  - GetNodeMap, 35
  - GetParam, 31
  - GetParamByIndex, 36



- GetRootCategory, [35](#)
- GetString, [34](#)
- GetStringValue, [42](#)
- IsCommandDone, [43](#)
- Poll, [44](#)
- SetBooleanValue, [36](#)
- SetEnumValue, [38](#)
- SetEnumValueStr, [37](#)
- SetFloatValue, [39](#)
- SetIntegerValue, [40](#)
- SetStringValue, [42](#)
- IpxGenParam::Boolean, [44](#)
  - GetType, [45](#)
  - GetValue, [46](#)
  - SetValue, [45](#)
- IpxGenParam::Category, [53](#)
  - GetCount, [54](#)
  - GetParamByIndex, [54](#)
  - GetType, [53](#)
- IpxGenParam::Command, [55](#)
  - Execute, [56](#)
  - GetType, [56](#)
  - IsDone, [56](#)
- IpxGenParam::Enum, [71](#)
  - GetEnumEntriesCount, [72](#)
  - GetEnumEntryByIndex, [73](#)
  - GetEnumEntryByName, [73](#)
  - GetEnumEntryByValue, [74](#)
  - GetType, [72](#)
  - GetValue, [74](#)
  - GetValueStr, [74](#)
  - SetValue, [75](#)
  - SetValueStr, [75](#)
- IpxGenParam::EnumEntry, [76](#)
  - GetType, [77](#)
  - GetValue, [77](#)
  - GetValueStr, [78](#)
- IpxGenParam::Float, [78](#)
  - GetMax, [81](#)
  - GetMin, [81](#)
  - GetType, [79](#)
  - GetUnit, [82](#)
  - GetValue, [80](#)
  - SetValue, [80](#)
- IpxGenParam::Int, [88](#)
  - GetIncrement, [91](#)
  - GetMax, [91](#)
  - GetMin, [90](#)
  - GetType, [89](#)
  - GetValue, [90](#)
  - SetValue, [89](#)
- IpxGenParam::Param, [102](#)
  - ~Param, [104](#)
  - GetDescription, [105](#)
  - GetDisplayName, [106](#)
  - GetName, [105](#)
  - GetNode, [108](#)
  - GetToolTip, [105](#)
  - GetType, [105](#)
  - GetVisibility, [106](#)
  - IsAvailable, [106](#)
  - IsReadable, [107](#)
  - IsStreamable, [107](#)
  - IsValueCached, [106](#)
  - IsVisible, [107](#)
  - IsWritable, [107](#)
  - RegisterEventSink, [108](#)
  - ToBoolean, [109](#)
  - ToCategory, [109](#)
  - ToCommand, [109](#)
  - ToEnum, [110](#)
  - ToEnumEntry, [109](#)
  - ToFloat, [110](#)
  - ToInt, [110](#)
  - ToString, [110](#)
  - UnregisterEventSink, [108](#)
- IpxGenParam::ParamEventSink, [111](#)
  - ~ParamEventSink, [111](#)
  - OnParameterUpdate, [112](#)
- IpxGenParam::String, [124](#)
  - GetMaxLength, [125](#)
  - GetType, [125](#)
  - GetValue, [126](#)
  - SetValue, [126](#)
- IpxGui, [15](#)
  - CreateGenParamTreeViewForArrayA, [17](#)
  - CreateGenParamTreeViewForArrayW, [18](#)
  - CreateGenParamTreeViewForNodemapA, [19](#)
  - CreateGenParamTreeViewForNodemapW, [20](#)
  - DestroyGenParamTreeView, [21](#)
  - SelectCameraA, [21](#)
  - SelectCameraW, [22](#)
  - ShowCamConfigDialog, [23](#)
  - ShowColorDialog, [27](#)
  - ShowFrameABDialog, [24](#)
  - ShowOutputDialog, [26](#)
  - ShowPulseDialog, [25](#)
  - ShowStrobeDialog, [25](#)
  - ShowTriggerDialog, [24](#)
  - Visibility, [17](#)
- IpxGui::IpxGenParamTreeView, [83](#)
  - ~IpxGenParamTreeView, [84](#)
  - clearParams, [86](#)
  - getPollingTime, [88](#)
  - loadState, [87](#)
  - saveState, [87](#)
  - setParams, [85](#)
  - setPollingTime, [87](#)

- setVisibility, [86](#)
  - visibility, [86](#)
- IsAvailable
  - IpxGenParam::Param, [106](#)
- IsCommandDone
  - IpxGenParam::Array, [43](#)
- IsDone
  - IpxGenParam::Command, [56](#)
- IsGrabbing
  - IpxCam::Stream, [123](#)
- IsIncomplete
  - IpxCam::Buffer, [50](#)
- IsKacFrameB
  - IpxCam::Buffer, [52](#)
- IsReadable
  - IpxGenParam::Param, [107](#)
- IsStreamable
  - IpxGenParam::Param, [107](#)
- IsValueCached
  - IpxGenParam::Param, [106](#)
- IsVisible
  - IpxGenParam::Param, [107](#)
- IsWritable
  - IpxGenParam::Param, [107](#)
- LoadConfiguration
  - IpxCam::Device, [65](#)
- loadState
  - IpxGui::IpxGenParamTreeView, [87](#)
- Namespace
  - IpxGenParam, [14](#)
- OnParameterUpdate
  - IpxGenParam::ParamEventSink, [112](#)
- ParamType
  - IpxGenParam, [14](#)
- Poll
  - IpxGenParam::Array, [44](#)
- QueueBuffer
  - IpxCam::Stream, [116](#)
- ReEnumerateDevices
  - IpxCam::Interface, [95](#)
- ReadMem
  - IpxCam::Device, [61](#)
- RegisterEvent
  - IpxCam::Device, [63](#)
  - IpxCam::Interface, [96](#)
  - IpxCam::Stream, [120](#)
- RegisterEventSink
  - IpxGenParam::Param, [108](#)
- RegisterGenTLProvider
  - IpxCam::System, [133](#)
- Release
  - IpxCam::List, [101](#)
  - IpxCam::Stream, [115](#)
  - IpxCam::System, [130](#)
- ReleaseBufferQueue
  - IpxCam::Stream, [120](#)
- RevokeBuffer
  - IpxCam::Stream, [116](#)
- SaveConfiguration
  - IpxCam::Device, [65](#)
- saveState
  - IpxGui::IpxGenParamTreeView, [87](#)
- SelectCameraA
  - IpxGui, [21](#)
- SelectCameraW
  - IpxGui, [22](#)
- ServiceFileType
  - IpxCam, [12](#)
- SetBooleanValue
  - IpxGenParam::Array, [36](#)
- SetBuffer
  - IpxCam::Stream, [115](#)
- SetEnumValue
  - IpxGenParam::Array, [38](#)
- SetEnumValueStr
  - IpxGenParam::Array, [37](#)
- SetFloatValue
  - IpxGenParam::Array, [39](#)
- SetIntegerValue
  - IpxGenParam::Array, [40](#)
- setParams
  - IpxGui::IpxGenParamTreeView, [85](#)
- setPollingTime
  - IpxGui::IpxGenParamTreeView, [87](#)
- SetStringValue
  - IpxGenParam::Array, [42](#)
- SetValue
  - IpxGenParam::Boolean, [45](#)
  - IpxGenParam::Enum, [75](#)
  - IpxGenParam::Float, [80](#)
  - IpxGenParam::Int, [89](#)
  - IpxGenParam::String, [126](#)
- SetValueStr
  - IpxGenParam::Enum, [75](#)
- setVisibility
  - IpxGui::IpxGenParamTreeView, [86](#)
- ShowCamConfigDialog
  - IpxGui, [23](#)
- ShowColorDialog
  - IpxGui, [27](#)
- ShowFrameABDialog
  - IpxGui, [24](#)

- ShowOutputDialog
  - IpxGui, [26](#)
- ShowPulseDialog
  - IpxGui, [25](#)
- ShowStrobeDialog
  - IpxGui, [25](#)
- ShowTriggerDialog
  - IpxGui, [24](#)
- StartAcquisition
  - IpxCam::Stream, [119](#)
- StopAcquisition
  - IpxCam::Stream, [119](#)
  
- ToBoolean
  - IpxGenParam::Param, [109](#)
- ToCategory
  - IpxGenParam::Param, [109](#)
- ToCommand
  - IpxGenParam::Param, [109](#)
- ToEnum
  - IpxGenParam::Param, [110](#)
- ToEnumEntry
  - IpxGenParam::Param, [109](#)
- ToFloat
  - IpxGenParam::Param, [110](#)
- ToInt
  - IpxGenParam::Param, [110](#)
- ToString
  - IpxGenParam::Param, [110](#)
  
- UnRegisterEvent
  - IpxCam::Device, [63](#)
  - IpxCam::Interface, [97](#)
  - IpxCam::Stream, [121](#)
- UnregisterEventSink
  - IpxGenParam::Param, [108](#)
- UploadEventCallback
  - IpxCam::Device, [59](#)
- UploadEventType
  - IpxCam::Device, [59](#)
- UploadFile
  - IpxCam::Device, [62](#)
  
- Visibility
  - IpxGenParam, [15](#)
  - IpxGui, [17](#)
- visibility
  - IpxGui::IpxGenParamTreeView, [86](#)
  
- WriteMem
  - IpxCam::Device, [62](#)