# Imperx Camera SDK

1.5.0.54

Generated by Doxygen 1.8.13

# Contents

Chapter 1

# Imperx Camera C++ SDK

## 1.1 General Information

The Imperx Camera C++ SDK is designed to provide software developers with C++ API functionality for ease of integrating Imperx cameras into their software applications. The API implemented in two libraries: IpxCameraApi and IpxCameraGuiApi. IpxCameraApi includes two namespaces: IpxCam and IpxGenParam. IpxCameraGuiApi includes IpxGui namespace.

The IpxCam namespace provides the scope to the API of GenICam GenTL transport layer to acquire images with an Imperx Camera. The IpxGenParam namespace provides the scope to the API to control the GenICam camera parameters, like image Width, Height, Pixel Format, Gain, Exposure, Trigger settings, etc. IpxGui namespace provides the scope for the user interface features, like windows and panels.

## 1.2 IpxCameraApi library

IpxCameraApi library includes classes, functions and types of IpxCam and IpxGenParam namespaces. It uses Imperx GenTL Producer library IpxCTI.cti to communicate with the cameras

### 1.2.1 IpxCam namespace

The IpxCam namespace consist of several main classes that represent the GenTL modules. The main classes are

- IpxCam::System - The System class is the entry point to the GenTL Producer software driver.
- IpxCam::Interface - The Interface class provides method to represents an individual physical interface, like GigE or USB3
- IpxCam::Device - The Device class provides methods to enable the communication with the camera device and enumerate/instantiate the video data streams.

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

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

**Example of GenTL System Hierarchy**

### 1.2.2 IpxGenParam namespace

The IpxGenParam namespace consist of the following main classes to access the GenICam parameters features. The main classes are

- IpxGenParam::Param - General class for accessing the GenICam feature node of the Camera parameters.
- IpxGenParam::Boolean - Class representing the Boolean GenICam camera parameter.
- IpxGenParam::Command - Class representing the Command GenICam camera parameter.
- IpxGenParam::Enum - Class representing the Enumeration GenICam camera parameter.
- IpxGenParam::Float - Class representing the Float GenICam camera parameter.
- IpxGenParam::Int - Class representing the Integer GenICam camera parameter.
- IpxGenParam::String - Class representing the String GenICam camera parameter.

## 1.3 IpxCameraGuiApi library

IpxCameraGuiApi library includes classes, functions and types of IpxGui namespace. The IpxGui namespace consist of the following GUI API classes and functions:

- IpxGui::SelectCameraA - function to show the modal dialog window of camera selection
- IpxGui::SelectCameraW - unicode version of IpxGui::SelectCameraA
- IpxGui::CreateGenParamTreeViewForArrayA - function to show the modeless dialog window of the camera Gen↩ ICam parameters
- IpxGui::CreateGenParamTreeViewForArrayW - unicode version of IpxGui::CreateGenParamTreeViewForArrayA
- IpxGui::DestroyGenParamTreeView - function to destroy the modeless dialog window of the camera GenICam parameters, created with IpxGui::CreateGenParamTreeViewForArrayA function call
- IpxGui::IIpxGenParamTreeView - Interface class for the modeless dialog window of the camera GenICam parameters. This class provides methods to set visibility level and parameters tree state.
- IpxGui::IpxGenParamTreeView - QT class, based on QWidget for the modeless window of the camera GenICam parameters.
- IpxGui::IpxCameraSelectorDialog - QT class, based on QDialog for the modal dialog window of camera selection

# Chapter 2

# Deprecated List

**Member IpxCam::Device::RegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗p←Private)=0**

Use Device::RegisterEvent2 instead

**Member IpxCam::Device::UnRegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗pPrivate)=0**

Use Device::UnRegisterEvent2 instead

**Member IpxCam::EventCallback (const void ∗eventData, size_t eventSize, void ∗pPrivate)**

Use EventCallback2 instead

**Member IpxCam::Interface::RegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗pPrivate)=0**

Use RegisterEvent2 instead

**Member IpxCam::Interface::UnRegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗pPrivate)=0**

Use UnRegisterEvent2 instead

# Chapter 3

# Namespace Index

## 3.1 Namespace List

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

# Chapter 4

# Hierarchical Index

## 4.1 Class Hierarchy

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

# Chapter 5

# Class Index

## 5.1 Class List

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

# Chapter 6

# Namespace Documentation

## 6.1 IpxCam Namespace Reference

A namespace providing scope to the GenICam GenTL transport layer interface to acquire images with an Imperx Camera.

### 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 camera device.*
- class Interface

  *The Interface class represents a interface module in the GenTL module hierarchy.*
- class List

  *The List class is used as list-like container for 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.*

### Typedefs

- typedef List< Interface > InterfaceList
- typedef List< DeviceInfo > DeviceInfoList
- typedef List< Device > DeviceList
- typedef void IPXCAM_CALL EventCallback(const void ∗eventData, size_t eventSize, void ∗pPrivate)
- typedef void IPXCAM_CALL EventCallback2(uint32_t eventType, const void ∗eventData, size_t eventSize, void ∗pPrivate)

  *EventCallback2.*

## 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 DeviceAccess : uint32_t { ReadOnly = 0, Control = 1, Exclusive = 2 }

  *An enum of Device Access.*

## Functions

- IPXCAM_EXTERN_C IPX_CAMERA_API System ∗ IpxCam_GetSystem ()

  *Returns the System object pointer.*

### 6.1.1 Detailed Description

A namespace providing scope to the GenICam GenTL transport layer interface to acquire images with an Imperx Camera.

IpxCam namespace includes classes that represent the base GenTLtransport layer modules: System, Interface, Device, Stream, Buffer. These modules can be used to enumerate the interfaces in the system, enumerate the cameras, connected to each interface, connect to necessary camera, allocate the memory buffers for images, and run the video acquisition.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 InterfaceList

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

List of Interface objects

#### 6.1.2.2 DeviceInfoList

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

List of DeviceInfo objects

#### 6.1.2.3 DeviceList

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

List of Device objects

#### 6.1.2.4 EventCallback

```
typedef void IPXCAM_CALL IpxCam::EventCallback(const void *eventData, size_t eventSize, void *p↩
Private)
```

EventCallback

**Deprecated** Use EventCallback2 instead

#### 6.1.2.5 EventCallback2

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

EventCallback2.

Callback function type for Event handling param[in] eventType type of the arrived event param[in] eventData pointer to event Data param[in] eventSize event Size param[in] pPrivate pointer to the context Data

### 6.1.3 Enumeration Type Documentation

#### 6.1.3.1 InterfaceType

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

An enum of Interface Types. Interface Node Types representing physical interface in the system.

**Enumerator**

| | |
|---|---|
| USB3Vision | Enum value for USB3Vision camera interface. |
| GigEVision | Enum value for GigEVision camera interface |
| CameraLink | Enum value for CameraLink camera interface |
| CoaxPress | Enum value for CoaxPress camera interface |
| HdSdi | Enum value for HD-SDI camera interface |
| AllInterfaces | Enum value AllInterfaces. |

#### 6.1.3.2 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. |

#### 6.1.3.3 DeviceAccess

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

An enum of Device Access.

**Enumerator**

| | |
|---|---|
| ReadOnly | Enum value ReadOnly. |
| Control | Enum value Control. |
| Exclusive | Enum value Exclusive. |

### 6.1.4 Function Documentation

#### 6.1.4.1 IpxCam_GetSystem()

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

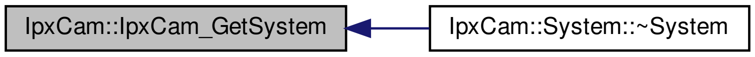
Returns the System object pointer.

This method returns the System module object. System object is being created as soon as API library is loaded. It is the entry point to the GenTL Module hierarchy.

**Returns**

Returns the pointer to system.

Here is the caller graph for this function:



## 6.2 IpxGenParam Namespace Reference

A namespace provides the scope to the API to access the GenICam parameters.

### Classes

- class Array

  *An Array class contains methods to access all GenICam camera parameters.*
- class Boolean

  *A class containing methods for Boolean GenICam camera parameter.*
- class Category

  *A class containing methods for GenICam Category.*
- class Command

  *A class containing methods for Command GenICam camera parameter.*
- class Enum

  *A class containing methods for Enumeration GenICam camera parameter.*
- class EnumEntry

  *EnumEntry class represents the entry of GenICam Enum parameter.*
- class Float

  *A class containing methods for Float GenICam camera parameter.*
- class Int

  *A class containing methods for Integer GenICam camera parameter.*
- class Param

  *General class for GenICam parameter.*
- class ParamEventSink

  *A Class for ParamEventSink notifications handling.*
- class String

  *A class containing methods for String GenICam camera parameter.*

## Enumerations

- enum ParamType : uint32_t {
  ParamUnknown, ParamInt, ParamFloat, ParamString,
  ParamEnum, ParamEnumEntry, ParamBoolean, ParamCommand,
  ParamCategory }

  *An enumeration 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 enumeration of GenICam NameSpace. Parameter Node Namespace.*
- enum Visibility : uint32_t {
  VisBeginner = 0, VisExpert, VisGuru, VisInvisible,
  VisUndefined = 99 }

  *An enumeration of Visibility. This element defines the level of user that has access to the feature.*

## 6.2.1 Detailed Description

A namespace provides the scope to the API to access the GenICam parameters.

The IpxGenParam namespace provides the scope to the API to control the GenICam camera parameters of types: Boolean, Enumeration, String, Float, Integer, Commands and Categories. Such parameters may include image Width, Height, Pixel Format, Gain, Exposure, Trigger, I/O settings, etc. Parameters are described in camera GenICam XML file, and documented in appropriate camera user's manual.

## 6.2.2 Enumeration Type Documentation

### 6.2.2.1 ParamType

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

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

**Enumerator**

| | |
|---|---|
| ParamUnknown | Enum value ParamUnknown. Unknown Parameter. |
| ParamInt | Enum value ParamInt will access node object's of IInteger interface. |
| ParamFloat | Enum value ParamFloat will access node object's of IFloat interface. |
| ParamString | Enum value ParamString will access node object's of IString interface. |
| ParamEnum | Enum value ParamEnum will access node object's of IEnumeration interface. |
| ParamEnumEntry | Enum value ParamEnumEntry will access the entry of Enum parameter. |
| ParamBoolean | Enum value ParamBoolean will access node object's of IBoolean interface. |
| ParamCommand | Enum value ParamCommand will access node object's of ICommand interface. |
| ParamCategory | Enum value ParamCategory will access node object's of ICategory interface. |

#### 6.2.2.2 NameSpace

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

An enumeration of GenICam NameSpace. Parameter Node Namespace.

**Enumerator**

| | |
|---|---|
| NameSpaceStandard | Enum value NameSpaceStandard. Identifies the standard namespace used in the file. |
| NameSpaceCustom | Enum value NameSpaceCustom. Identifies the custom namespace used in the file. |
| NameSpaceUndefined | Enum value NameSpaceUndefined. Unknown namespace. |

#### 6.2.2.3 Visibility

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

An enumeration of Visibility. This element defines the level of user that has access to the feature.

**Enumerator**

| | |
|---|---|
| VisBeginner | Enum value VisBeginner. User has visibility to all the basic features of the device. |
| VisExpert | Enum value VisExpert. User has visibility to more advance features of the device. |
| VisGuru | Enum value VisGuru. User has visibility to even more advance features that if set improperly can cause device to be in an improper state. |
| VisInvisible | Enum value VisInvisible. Not visible. |
| VisUndefined | Enum value VisUndefined. Unknown visibility. |

## 6.3 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 IIpxGenParamTreeView

  *IIpxGenParamTreeView class represents the GenICam parameters node tree panel.*

## 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 IIpxGenParamTreeView ∗ CreateGenParamTreeViewFor↩ ArrayA (IpxGenParam::Array ∗genParam, const char ∗title, uintptr_t parentWindow=0)

    *Creates the panel of the camera GenICam parameters for IpxGenParam::Array object.*
- IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IIpxGenParamTreeView ∗ CreateGenParamTreeViewFor↩ ArrayW (IpxGenParam::Array ∗genParam, const wchar_t ∗title, uintptr_t parentWindow=0)

    *Creates the panel of the camera GenICam parameters for IpxGenParam::Array object. Unicode version.*
- IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IIpxGenParamTreeView ∗ CreateGenParamTreeViewFor↩ NodemapA (IPX_GENAPI_NS::INodeMap ∗nodemap, const char ∗title, uintptr_t parentWindow=0)

    *Creates the panel of the camera GenICam parameters for GenApi::INodeMap object.*
- IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IIpxGenParamTreeView ∗ CreateGenParamTreeViewFor↩ NodemapW (IPX_GENAPI_NS::INodeMap ∗nodemap, const wchar_t ∗title, uintptr_t parentWindow=0)

    *Creates the panel of the camera GenICam parameters for GenApi::INodeMap object. Unicode version.*
- IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void DestroyGenParamTreeView (IIpxGenParamTreeView ∗view)

    *Destroys the GenICam parameters panel.*
- IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IpxCam::DeviceInfo ∗ SelectCameraA (IpxCam::System ∗pSystem, const char ∗title, uintptr_t parentWindow=0, bool poll=true)

    *Pops-up the camera selection dialog.*
- 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)

    *Pops-up the camera selection dialog. Unicode version.*
- 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.*

### 6.3.1 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: IIpxGenParamTreeView, SelectCameraA SelectCameraW IpxGenParamTreeView, IpxCameraSelectorDialog

### 6.3.2 Enumeration Type Documentation

#### 6.3.2.1 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. |

### 6.3.3 Function Documentation

#### 6.3.3.1 CreateGenParamTreeViewForArrayA()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IIpxGenParamTreeView* IpxGui::CreateGenParamTreeViewFor←↩
ArrayA (
                IpxGenParam::Array * genParam,
                const char * title,
                uintptr_t parentWindow = 0 )
```

Creates the panel of the camera GenICam parameters for IpxGenParam::Array object.

This function returns the pointer to the IIpxGenParamTreeView class that was created using information extracted from the IpxGenParam::Array class.

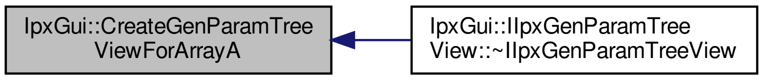
**Parameters**

| | | |
|---|---|---|
| in | *genParam* | The pointer to the IpxGenParam::Array class. |
| in | *title* | The title of the IIpxGenParamTreeView class as a const char. |
| in | *parentWindow* | 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 function succeeds, the return value is the pointer to the IIpxGenParamTreeView class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:


IpxGui::CreateGenParamTree ViewForArrayA
IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView

#### 6.3.3.2 CreateGenParamTreeViewForArrayW()

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

Creates the panel of the camera GenICam parameters for IpxGenParam::Array object. Unicode version.

This function returns the pointer to the IIpxGenParamTreeView class that was created using information extracted from the IpxGenParam::Array.

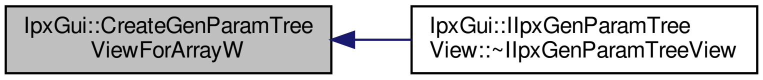
**Parameters**

| | | |
|---|---|---|
| in | *genParam* | The pointer to the IpxGenParam::Array class. |
| in | *title* | The title of the IIpxGenParamTreeView class as a wchar_t variable. |
| in | *parentWindow* | 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 function succeeds, the return value is the pointer to the IIpxGenParamTreeView class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:



#### 6.3.3.3 CreateGenParamTreeViewForNodemapA()

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

Creates the panel of the camera GenICam parameters for GenApi::INodeMap object.

This function returns the pointer to the IIpxGenParamTreeView class that was created using information extracted from the GenApi::INodeMap class.

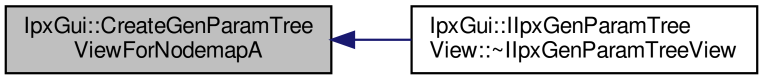
**Parameters**

| | | |
|---|---|---|
| in | *nodemap* | The pointer to the GenApi::INodeMap class. |
| in | *title* | The title of the IIpxGenParamTreeView class as a wchar_t variable. |
| in | *parentWindow* | 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 function succeeds, the return value is the pointer to the IIpxGenParamTreeView class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:


IpxGui::CreateGenParamTree ViewForNodemapA
IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView

#### 6.3.3.4 CreateGenParamTreeViewForNodemapW()

```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API IIpxGenParamTreeView* IpxGui::CreateGenParamTreeViewFor←↩
NodemapW (
            IPX_GENAPI_NS::INodeMap * nodemap,
            const wchar_t * title,
            uintptr_t parentWindow = 0 )
```

Creates the panel of the camera GenICam parameters for GenApi::INodeMap object. Unicode version.

This function returns the pointer to the IIpxGenParamTreeView that was created using information extracted from the GenApi::INodeMap class.

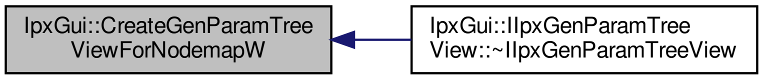
**Parameters**

| | | |
|---|---|---|
| in | *nodemap* | The pointer to the GenApi::INodeMap class. |
| in | *title* | The title of the IIpxGenParamTreeView as a wchar_t variable. |
| in | *parentWindow* | 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 function succeeds, the return value is the pointer to the IIpxGenParamTreeView class created. Otherwise, the return value is nullptr.

Here is the caller graph for this function:


IpxGui::CreateGenParamTree ViewForNodemapW
IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView

### 6.3.3.5 DestroyGenParamTreeView()

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

Destroys the GenICam parameters panel.

This function closes the camera GenICam parameters panel and destroys the IIpxGenParamTreeView object previously created with CreateGenParamTreeViewForNodemap∗ or CreateGenParamTreeViewForArray∗ function

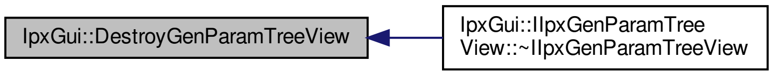
**Parameters**

| | | |
|---|---|---|
| in | *view* | A pointer to the IIpxGenParamTreeView class. |

**Returns**

void

Here is the caller graph for this function:


IpxGui::DestroyGenParamTreeView
IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView

#### 6.3.3.6 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 )
```

Pops-up the camera selection dialog.

This function 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

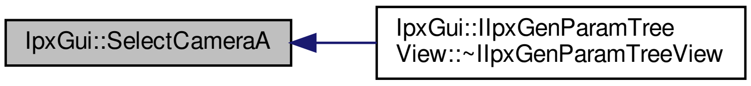
**Parameters**

| | | |
|---|---|---|
| in | *pSystem* | The pointer to the IpxCam::System class. |
| in | *title* | The title of the selected Camera as a const char variable. |
| in | *parentWindow* | 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 | *poll* | Specifies if poll check box should be checked by default, so the System will be polled for new devices to appear |

**Returns**

If the function 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:



#### 6.3.3.7 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 )
```

Pops-up the camera selection dialog. Unicode version.

This function 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.

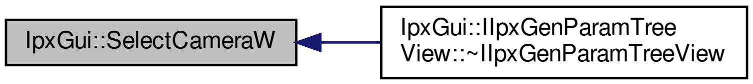
**Parameters**

| | | |
|---|---|---|
| in | *pSystem* | The pointer to the IpxCam::System class. |
| in | *title* | The title of the IIpxGenParamTreeView as a wchar_t variable. |
| in | *parentWindow* | 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 | *poll* | Specifies if poll check box should be checked by default, so the System will be polled for new devices to appear |

**Returns**

If the function 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:



#### 6.3.3.8 ShowCamConfigDialog()

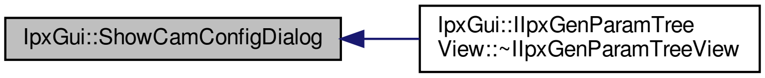
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowCamConfigDialog (
            IpxCam::Device * device,
            uintptr_t parentWindow = 0 )
```

Show Camera Configuration Dialog.

**Parameters**

| | | |
|---|---|---|
| in | *device* | The pointer to the IpxCam::Device class. |
| in | *parentWindow* | 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:


IpxGui::ShowCamConfigDialog
IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView

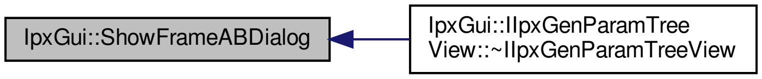
#### 6.3.3.9 ShowFrameABDialog()

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

**Parameters**

| | | |
|---|---|---|
| in | *device* | The pointer to the IpxCam::Device class. |
| in | *parentWindow* | 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:


IpxGui::ShowFrameABDialog
IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView

#### 6.3.3.10 ShowTriggerDialog()

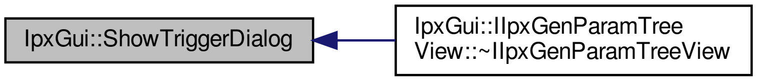
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowTriggerDialog (
            IpxCam::Device * device,
            uintptr_t parentWindow = 0 )
```

Show Trigger Dialog.

**Parameters**

| | | |
|---|---|---|
| in | *device* | The pointer to the IpxCam::Device class. |
| in | *parentWindow* | 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:


IpxGui::ShowTriggerDialog
IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView

#### 6.3.3.11 ShowPulseDialog()

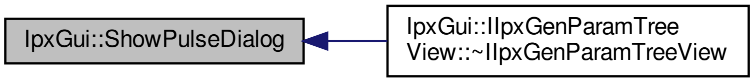
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowPulseDialog (
            IpxCam::Device * device,
            uintptr_t parentWindow = 0 )
```

Show Pulse Dialog.

**Parameters**

| | | |
|---|---|---|
| in | *device* | The pointer to the IpxCam::Device class. |
| in | *parentWindow* | 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:



#### 6.3.3.12 ShowStrobeDialog()

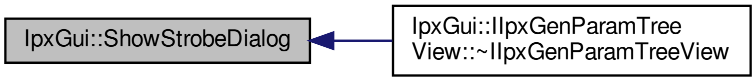
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowStrobeDialog (
            IpxCam::Device * device,
            uintptr_t parentWindow = 0 )
```

Show Strobe Dialog.

**Parameters**

| | | |
|---|---|---|
| in | *device* | The pointer to the IpxCam::Device class. |
| in | *parentWindow* | 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:



#### 6.3.3.13 ShowOutputDialog()

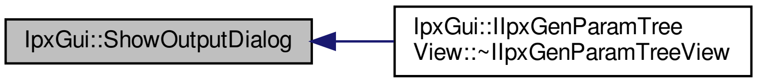
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowOutputDialog (
            IpxCam::Device * device,
            uintptr_t parentWindow = 0 )
```

Show Output Data Dialog.

**Parameters**

| | | |
|---|---|---|
| in | *device* | The pointer to the IpxCam::Device class. |
| in | *parentWindow* | 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:



#### 6.3.3.14 ShowColorDialog()

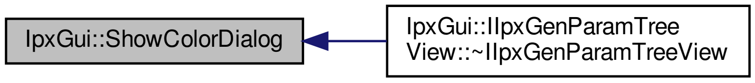
```
IPXCAM_GUI_EXTERN_C IPX_CAMERA_GUI_API void IpxGui::ShowColorDialog (
             IpxCam::Device * device,
             uintptr_t parentWindow = 0 )
```

Show Color Dialog.

**Parameters**

| | | |
|---|---|---|
| in | *device* | The pointer to the IpxCam::Device class. |
| in | *parentWindow* | 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 7

# Class Documentation

## 7.1 IpxGenParam::Array Class Reference

An Array class contains methods to access all GenICam camera parameters.

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

### Public Member Functions

- virtual ∼Array ()

  *Array class destructor.*
- virtual Param ∗ GetParam (const char ∗name, IpxCamErr ∗err)=0

  *This method gets the pointer to the Param class object 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 object 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 object 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 object 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 object 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 object 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 object 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 category node object. 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, IpxCamErr ∗err)=0

  *This method gets the parameter by index.*
- virtual IpxCamErr SetBooleanValue (const char ∗name, bool aValue)=0

  *This method sets the Boolean value of the Boolean node.*
- virtual bool GetBooleanValue (const char ∗name, IpxCamErr ∗err=nullptr)=0

  *This method gets the Boolean value of the Boolean node.*
- virtual IpxCamErr 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 IpxCamErr 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, IpxCamErr ∗err=nullptr)=0

  *This method gets the Enum value string of the current set Enum value entry.*
- virtual int64_t GetEnumValue (const char ∗name, IpxCamErr ∗err=nullptr)=0

  *This method gets the Enum value of the Enum node.*
- virtual IpxCamErr SetFloatValue (const char ∗name, double val)=0

  *This method sets the Float value of the Float node.*
- virtual double GetFloatValue (const char ∗name, IpxCamErr ∗err=nullptr)=0

  *This method gets the Float value of the Float node.*
- virtual IpxCamErr 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, IpxCamErr ∗err=nullptr)=0

  *This method gets the Integer value of the Integer node.*
- virtual IpxCamErr 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, IpxCamErr ∗err=nullptr)=0

  *This method gets the String value of the String node.*
- virtual IpxCamErr ExecuteCommand (const char ∗name)=0

  *This method executes/submits the command.*
- virtual bool IsCommandDone (const char ∗name, IpxCamErr ∗err=nullptr)=0

  *This method polls the corresponding executed command to see if the executed command is done or not.*
- virtual IpxCamErr Poll (int64_t elapsedTime)=0

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

### 7.1.1 Detailed Description

An Array class contains methods to access all GenICam camera parameters.

This class contains methods that can access each node from the GenICam camera description XML file by parameters type and name.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 ∼Array()

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

Array class destructor.

Array class destructor. Destroys the Array object and all its descendants.

### 7.1.3 Member Function Documentation

#### 7.1.3.1 GetParam()

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

This method gets the pointer to the Param class object for the specified node name from the node map declared in the camera descriptor XML file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Unique name of a node in node map. |
| out | *err* | Returns an error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Param class of the specified node name<br>• `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - specified node name not found in camera descriptor XML file |

**Returns**

If the method succeeds, it returns the pointer to the Param class for the specific node name. Otherwise, it returns a nullptr.

### 7.1.3.2 GetBoolean()

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

This method gets the pointer to the Boolean class object for the specified node name of the camera descriptor XML file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | A unique name of Boolean type node in the camera descriptor XML file. |
| out | *err* | Returns an error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Boolean class of the specified node name<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - specified node name not found in camera descriptor XML file |

**Returns**

If the method succeeds, it returns the pointer to the Boolean class for the specific node name. Otherwise, it returns a nullptr.

### 7.1.3.3 GetCommand()

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

This method gets the pointer to the Command class object for the specified node name of the camera descriptor XML file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Unique name of Command type node in the camera descriptor XML file. |
| out | *err* | returns an error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Command class of the specified node name<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - specified node name not found in camera descriptor XML file |

**Returns**

If method succeeds, it returns the pointer to the Command class for the specific node name. Otherwise, it returns a nullptr.

#### 7.1.3.4 GetEnum()

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

This method gets the pointer to the Enum class object for the specified node name of the camera descriptor XML file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Unique name of Enumeration type node in the camera descriptor XML file. |
| out | *err* | returns an error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Enum class of the specified node name<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - specified node name not found in camera descriptor XML file |

**Returns**

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

#### 7.1.3.5 GetFloat()

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

This method gets the pointer to the Float class object for the specified node name of the camera descriptor XML file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Unique name of Float type node in the camera descriptor XML file. |
| out | *err* | returns an error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Float class of the specified node name<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - specified node name not found in camera descriptor XML file |

**Returns**

If the method succeeds, it returns the pointer to the Float parameter class for the specific node name

### 7.1.3.6 GetInt()

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

This method gets the pointer to the Int class object for the specified node name of the camera descriptor XML file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Unique name of Integer type node in the camera descriptor XML file. |
| out | *err* | returns an error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Int class of the specified node name <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - specified node name not found in camera descriptor XML file |

**Returns**

If the method succeeds, it returns the pointer to the Int class for the specific node name

### 7.1.3.7 GetString()

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

This method gets the pointer to the String class object for the specified node name of the camera descriptor XML file.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Unique name of String type node in the camera descriptor XML file. |
| out | *err* | returns an error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to String class of the specified node name <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - specified node name not found in camera descriptor XML file |

**Returns**

If the method succeeds, it returns the pointer to the String class for the specific node name

#### 7.1.3.8 GetRootCategory()

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

This method gets the pointer to the root category node object. 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Category class<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - specified Root node name not found in camera descriptor XML file |

**Returns**

Returns the pointer to the Category (root node) class

#### 7.1.3.9 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to GenApi::INodeMap class<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - the node map does not exist |

**Returns**

nodemap returns the pointer to the NodeMap interface

### 7.1.3.10 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.

### 7.1.3.11 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to Param class<br>• `IpxCamErr::IPX_CAM_ERR_INVALID_INDEX` - entered invalid index |

**Returns**

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

### 7.1.3.12 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` | *name* | Unique name of Boolean node to set |
| `in` | *aValue* | Boolean 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

### 7.1.3.13 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` | *name* | Unique name of Boolean node to get |
| `out` | *err* | returns the error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Boolean value<br>• `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node |

**Returns**

Returns the Boolean Value

### 7.1.3.14 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 | *name* | Name of Enum entry node to set |
| in | *val* | Enum node string value to 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 value entered is out of range

#### 7.1.3.15 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 | *name* | Unique name of Enum entry to set |
| in | *val* | Enum entry integer 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

#### 7.1.3.16 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 | *name* | Unique name of Enum entry |
| out | *err* | returns error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Enum string value <br>• `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Get the Enum value String of the current set Enum Value Entry

### 7.1.3.17 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 | *name* | Unique name of Enum type node in the camera descriptor XML file. |
| out | *err* | returns error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Enum value <br>• `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the Enum Value

### 7.1.3.18 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` | *name* | Unique name of Float type node in the camera descriptor XML file. |
| `in` | *val* | Float 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

#### 7.1.3.19 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` | *name* | Unique name of Float type node in the camera descriptor XML file. |
| `out` | *err* | returns the error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Float value<br>• `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the Float value

#### 7.1.3.20 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 | *name* | Unique name of Integer type node in the camera descriptor XML file. |
| in | *val* | 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

#### 7.1.3.21 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 | *name* | Unique name of Integer type node in the camera descriptor XML file. |
| out | *err* | returns the error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Integer value <br>• `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the Integer value

### 7.1.3.22 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 | *name* | Unique name of String type node in the camera descriptor XML file. |
| in | *val* | String 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

### 7.1.3.23 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 | *name* | Unique name of String type node in the camera descriptor XML file. |
| out | *err* | returns the error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the String value<br>• `IpxCamErr::IPX_CAM_GENICAM_UNKNOWN_PARAM` - unknown parameter<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |
| | | |

**Returns**

Returns the String value

#### 7.1.3.24 ExecuteCommand()

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

This method executes/submits the command.

**Parameters**

| | | |
|---|---|---|
| `in` | *name* | Unique name of Command 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

#### 7.1.3.25 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` | *name* | Unique name of Command type node in the camera descriptor XML file. |
| `out` | *err* | 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 |

**Returns**

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

#### 7.1.3.26 Poll()

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

This method fires nodes which have a polling time.

**Parameters**

| | | |
|---|---|---|
| in | *elapsedTime* | 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:

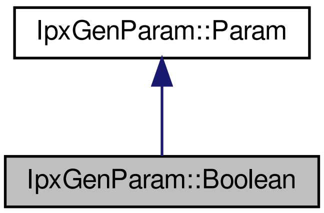
- IpxCameraApi.h

## 7.2 IpxGenParam::Boolean Class Reference

A class containing methods for Boolean GenICam camera parameter.

```
#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.*

### 7.2.1 Detailed Description

A class containing methods for Boolean GenICam camera parameter.

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

For example, the mapping below will illustrate the IBoolean interfaces of a **LUTEnable** feature.

### 7.2.2 Member Function Documentation

#### 7.2.2.1 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.

#### 7.2.2.2 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

#### 7.2.2.3 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: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the value of the Boolean node <br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

The node value read.

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

- IpxCameraApi.h

## 7.3 IpxCam::Buffer Class Reference

The Buffer class represents the buffer module in the GenTL module hierarchy.

`#include <IpxCameraApi.h>`

## Public Member Functions

- virtual ∼Buffer ()

    *Buffer class destructor.*
- virtual IpxImage ∗ GetImage ()=0

    *Returns the pointer to the IpxImage structure.*
- virtual void ∗ GetBufferPtr ()=0

    *Returns the pointer to the image data.*
- virtual size_t GetImageOffset ()=0

    *Returns the offset of the actual image data start.*
- virtual size_t GetBufferSize ()=0

    *This method returns the size of the allocated memory buffer in bytes.*
- 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, associated with the buffer object.*
- virtual uint64_t GetTimestamp ()=0

    *This method returns the timestamp of the acquired buffer.*
- virtual uint64_t GetFrameID ()=0

    *This method returns the identificator of the image stream block of the buffer object.*
- virtual bool IsIncomplete ()=0

    *This method returns a flag indicating if the buffer data has been fully transferred or incompleted.*
- virtual size_t GetWidth ()=0

    *Returns the image width.*
- virtual size_t GetHeight ()=0

    *Returns the image height.*
- virtual size_t GetXOffset ()=0

    *Returns the horizontal offset of the image data in the buffer.*
- virtual size_t GetYOffset ()=0

    *Returns the vertical offset of the image data in the buffer.*
- virtual size_t GetXPadding ()=0

    *This method returns the number of extra bytes padded in the horizontal direction.*
- virtual size_t GetYPadding ()=0

    *This method returns the number of extra bytes padded in the vertical 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.*

### 7.3.1 Detailed Description

The Buffer class represents the buffer module in the GenTL module hierarchy.

The Buffer class contains the methods that can be used to get the pointer to the acquired image data memory and / or retrieve the information about the received image data such as timestamp, image size, pixel format, etc

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 ∼Buffer()

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

Buffer class destructor.

Destroys the Buffer object and all its descendants.

**Returns**

none

### 7.3.3 Member Function Documentation

#### 7.3.3.1 GetImage()

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

Returns the pointer to the IpxImage structure.

This method returns the pointer to the IpxImage structure. See IpxTools user's manual for IpxImage structure description.

**Returns**

Returns the pointer to the IpxImage structure.

#### 7.3.3.2 GetBufferPtr()

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

Returns the pointer to the image data.

This method returns the pointer to the memory of the acquired image data.

**Returns**

Returns the pointer to the image data

#### 7.3.3.3 GetImageOffset()

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

Returns the offset of the actual image data start.

This method returns the offset of the actual image data start in the acquired data buffer memory.

**Returns**

Returns the offset of the actual image data start

#### 7.3.3.4 GetBufferSize()

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

This method returns the size of the allocated memory buffer in bytes.

**Returns**

Returns the buffer size in bytes

#### 7.3.3.5 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 image in the buffer object. This value equals to **PixeFormat** GenICam parameter

#### 7.3.3.6 GetUserPtr()

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

This method returns the user data buffer pointer, associated with the buffer object.

**Returns**

Returns the user data buffer pointer

#### 7.3.3.7 GetTimestamp()

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

This method returns the timestamp of the acquired buffer.

This method returns the timestamp of the acquired buffer. Imperx USB3 and GEV cameras have 10ns timestamp granularity. GEV cameras timestamp clock frequency can be obtained from **GevTimestampTickFrequency** GenICam parameter

**Returns**

Returns the timestamp of the acquired buffer.

#### 7.3.3.8 GetFrameID()

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

This method returns the identificator of the image stream block of the buffer object.

**Returns**

Returns the identificator of the image stream block of the buffer object.

#### 7.3.3.9 IsIncomplete()

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

This method returns a flag indicating if the buffer data has been fully transferred or incompleted.

**Returns**

Returns True, if buffer transfer was incompleted, False, if transfer was successful

#### 7.3.3.10 GetWidth()

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

Returns the image width.

This method returns the image width of the buffer data in number of pixels. Usually the return value equals to **Width** GenICam parameter value

**Returns**

Returns the image width

#### 7.3.3.11 GetHeight()

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

Returns the image height.

This method returns the image height of the buffer data in number of lines. Usually the return value equals to **Height** GenICam parameter value

**Returns**

Returns the image height

#### 7.3.3.12 GetXOffset()

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

Returns the horizontal offset of the image data in the buffer.

This method returns the horizontal offset of the image data in the buffer in number of pixels from the image origin. Usually the return value equals to **OffsetX** GenICam parameter value

**Returns**

Returns the horizontal offset in number of pixels

#### 7.3.3.13 GetYOffset()

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

Returns the vertical offset of the image data in the buffer.

This method returns the vertical offset of the image data in the buffer in number of lines from the image origin. Usually the return value equals to **OffsetY** GenICam parameter value

**Returns**

Returns the vertical offset of the data in the buffer in number of lines from the image origin

#### 7.3.3.14 GetXPadding()

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

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

**Returns**

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

#### 7.3.3.15 GetYPadding()

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

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

**Returns**

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

#### 7.3.3.16 GetDeliveredHeight()

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

This method returns the actual height of delivered data.

This method returns the actual height of delivered data. Can be different than value returned by GetHeight() method, if image transfer was incompleted.

**Returns**

Returns the actual height of delivered data

#### 7.3.3.17 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:

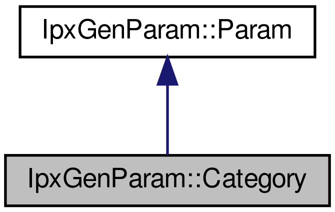
- IpxCameraApi.h

## 7.4 IpxGenParam::Category Class Reference

A class containing methods for GenICam 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 parameters in the category.*
- virtual Param ∗ GetParamByIndex (uint32_t idx, IpxCamErr ∗err)=0

  *This method returns the Parameter by Index.*

### 7.4.1 Detailed Description

A class containing methods for GenICam Category.

A class containing methods that the user can access the categories of 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.

For example, the mapping below will illustrate the ICategory interfaces categories such as DeviceControl and Event↩ Control.

### 7.4.2 Member Function Documentation

#### 7.4.2.1 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.

#### 7.4.2.2 GetCount()

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

This method returns the number of parameters in the category.

**Returns**

Returns the number of parameters in the category

#### 7.4.2.3 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully returns pointer to the parameter for specified index<br>• `IpxCamErr::IPX_CAM_ERR_INVALID_INDEX` - an invalid index for node |

**Returns**

Returns the pointer to the parameter object

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

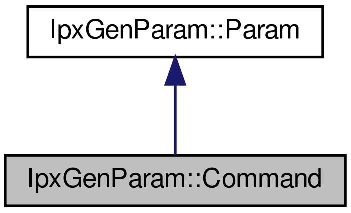
- IpxCameraApi.h

## 7.5 IpxGenParam::Command Class Reference

A class containing methods for Command GenICam camera parameter.

`#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 and completed.*

### 7.5.1 Detailed Description

A class containing methods for Command GenICam camera parameter.

A class for GenICam Command contains methods that allow the user submit a command for execution as well as poll the command status.

For example, the mapping below will illustrate the ICommand interface for AcquisitionStart. This feature starts the Acquisition of the device.

### 7.5.2 Member Function Documentation

#### 7.5.2.1 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.

#### 7.5.2.2 Execute()

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

This method executes the command.

**Returns**

the error code

#### 7.5.2.3 IsDone()

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

This method queries whether the command is executed and completed.

**Parameters**

|  |  |  |
|---|---|---|
| `out` | *err* | returns error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully determined that state of execute command <br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node <br>• `IpxCamErr::IPX_CAM_GENICAM_TREE_ERROR` - Unable to access tree |

**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

## 7.6 IpxCam::Device Class Reference

The Device class represents the device module in the GenTL module hierarchy.

`#include <IpxCameraApi.h>`

### Public Types

- enum UploadEventType : uint32_t { FlashSectorErase, FlashPageWrite, FlashPageRead }
- enum Endianness : uint8_t { BigEndian, LittleEndian }

  *An enum of endianness types of underlying protocol.*

### 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 IpxCamErr 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 IpxCamErr 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 IpxCamErr RegisterEvent2 (uint32_t eventType, IpxCam::EventCallback2 ∗eventCallback, void ∗p←Private)=0

  *This method registers the Device class method as a callback method to be called when a eventType occurs.*
- virtual IpxCamErr RegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗pPrivate)=0

  *RegisterEvent.*
- virtual IpxCamErr UnRegisterEvent2 (uint32_t eventType, IpxCam::EventCallback2 ∗eventCallback, void ∗p←Private)=0

  *This event occurs, when the camera was disconnected from the System.*
- virtual IpxCamErr UnRegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗p←Private)=0

  *UnRegisterEvent.*
- virtual IpxGenParam::Array ∗ GetTransportParameters (IpxCamErr ∗err=nullptr)=0

  *This method returns the transport parameters IpxGenParam::Array object of the camera device object.*
- virtual IpxGenParam::Array ∗ GetCameraParameters (IpxCamErr ∗err=nullptr)=0

  *This method returns the camera parameters IpxGenParam::Array object of the device object.*
- virtual IpxCamErr SaveConfiguration (const char ∗fileName)=0

  *This method saves the camera parameters to the configuration file.*
- virtual IpxCamErr LoadConfiguration (const char ∗fileName)=0

  *This method loads the configuration from file, and configures the camera with the parameter values, saved to this file.*
- virtual Endianness GetEndianness () const =0

  *This method returns endianness of underlying protocol for this camera device.*

### Static Public Attributes

- static const uint32_t CameraConnected = 1003

  *This event occurs, if GenICam event was triggered by the camera device.*
- static const uint32_t CameraDisconnected = 1004

  *This event occurs, when the camera was connected to the System.*

## 7.6.1 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 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.

## 7.6.2 Member Enumeration Documentation

#### 7.6.2.1 UploadEventType

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

**Enumerator**

| | |
|---|---|
| FlashSectorErase | Enum value FlashSectorErase. |
| FlashPageWrite | Enum value FlashPagewrite. |
| FlashPageRead | Enum value FlashPageRead. |

#### 7.6.2.2 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 |

### 7.6.3 Constructor & Destructor Documentation

#### 7.6.3.1 ∼Device()

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

A destructor of the Device class.

Destructor. Destroys the Device and all its descendants.

### 7.6.4 Member Function Documentation

#### 7.6.4.1 GetNumStreams()

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

This method retrieves the number of the data streams, provided by the Device.

**Returns**

returns the number of the data streams

#### 7.6.4.2 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 | *idx* | stream index value |

**Returns**

Returns the pointer to the Stream object

#### 7.6.4.3 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 | *id* | pointer to the string representing the stream identifier |

**Returns**

Returns the pointer to the Stream object

#### 7.6.4.4 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

#### 7.6.4.5 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 | *addr* | Byte address to read from |
| in | *data* | pointer to a user allocated byte data buffer |
| in | *len* | size of the amount of bytes to read from the register map address |

**Returns**

Returns ErrorCode

#### 7.6.4.6 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 | *addr* | Byte address to read from |
| in | *data* | pointer to a user allocated byte data buffer |
| in | *len* | size of the amount of bytes to write to the register map address |
| out | *written* | size of bytes written |

**Returns**

Returns ErrorCode

#### 7.6.4.7 RegisterEvent2()

```
virtual IpxCamErr IpxCam::Device::RegisterEvent2 (
             uint32_t eventType,
             IpxCam::EventCallback2 * 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 | *eventType* | Event Type, can receive one of the following values:<br>• **`GenICamEvent`** [1002] - this event occurs, if GenICam event was triggered by the camera<br>• **`CameraConnected`** [1003] - this event occurs, when camera was connected to the System<br>• **`CameraDisconnected`** [1004] - this event occurs, when camera was disconnected from the System |
| in | *eventCallback* | event CallBack |
| in | *pPrivate* | pointer to user's data |

**Returns**

Returns Error code

#### 7.6.4.8 RegisterEvent()

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

RegisterEvent.

**Deprecated** Use Device::RegisterEvent2 instead

#### 7.6.4.9 UnRegisterEvent2()

```
virtual IpxCamErr IpxCam::Device::UnRegisterEvent2 (
             uint32_t eventType,
             IpxCam::EventCallback2 * eventCallback,
             void * pPrivate )  [pure virtual]
```

This event occurs, when the camera was disconnected from the System.

This method unregisters the Interface class callback method for the eventType.

**Parameters**

| | | |
|---|---|---|
| in | *eventType* | Event Type, can receive one of the following values: • **GenICamEvent** [1002] - this event occurs, if GenICam event was triggered by the camera • **CameraConnected** [1003] - this event occurs, when camera was connected to the System • **CameraDisconnected** [1004] - this event occurs, when camera was disconnected from the System |
| in | *eventCallback* | event CallBack |
| in | *pPrivate* | pointer to user's data |

**Returns**

Returns Error code

### 7.6.4.10 UnRegisterEvent()

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

UnRegisterEvent.

**Deprecated** Use Device::UnRegisterEvent2 instead

### 7.6.4.11 GetTransportParameters()

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

This method returns the transport parameters IpxGenParam::Array object of the camera device object.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code |

**Returns**

Returns the Transport parameters object pointer

#### 7.6.4.12 GetCameraParameters()

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

This method returns the camera parameters IpxGenParam::Array object of the device object.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code |

**Returns**

Returns the Camera Parameters array object pointer

#### 7.6.4.13 SaveConfiguration()

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

This method saves the camera parameters to the configuration file.

**Parameters**

| | | |
|---|---|---|
| in | *fileName* | Configuration file name |

**Returns**

Returns Error code

#### 7.6.4.14 LoadConfiguration()

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

This method loads the configuration from file, and configures the camera with the parameter values, saved to this file.

**Parameters**

| | | |
|---|---|---|
| in | *fileName* | Configuration file name |

**Returns**

Returns Error code

#### 7.6.4.15 GetEndianness()

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

This method returns endianness of underlying protocol for this camera device.

**Returns**

Returns endianness

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

- IpxCameraApi.h

## 7.7 IpxCam::DeviceInfo Class Reference

DeviceInfo class provides the information about the camera device.

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

### Public Member Functions

- virtual ∼DeviceInfo ()

  *DeviceInfo class destructor.*
- 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 vendor name of the camera device object.*
- virtual const char ∗ GetModel ()=0

  *This method returns the model name of the camera device object.*
- virtual const char ∗ GetDisplayName ()=0

*This method returns the user readable 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

  *Returns the device access status.*
- virtual const char ∗ GetUSB3HostInfo ()=0

  *Returns the information about USB3 host controller.*
- virtual const char ∗ GetIPAddress (IpxCamErr ∗err)=0

  *Returns the IP address of the GEV camera.*
- virtual const char ∗ GetIPMask (IpxCamErr ∗err)=0

  *Returns the IP subnet mask of the GEV camera.*
- virtual const char ∗ GetIPGateway (IpxCamErr ∗err)=0

  *Returns the IP gateway of GEV camera.*
- virtual IpxCamErr GetIP (uint32_t ∗addr, uint32_t ∗netmask, uint32_t ∗gateway)=0

  *Gets IP information from the GEV camera.*
- virtual IpxCamErr ForceIP (const char ∗addr, const char ∗netmask, const char ∗gateway)=0

  *Set the IP address to GEV camera.*
- virtual IpxCamErr ForceIP (uint32_t addr, uint32_t netmask, uint32_t gateway)=0

  *Set IP address to GEV camera.*

### 7.7.1 Detailed Description

DeviceInfo class provides the information about the camera device.

The DeviceInfo class can be used to retrieve the information about the device, and to create the IpxCam::Device object by IpxCam_CreateDevice() call

### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 ∼DeviceInfo()

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

DeviceInfo class destructor.

Destroys the DeviceInfo object and all its descendants.

### 7.7.3 Member Function Documentation

#### 7.7.3.1 GetInterface()

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

This method returns the interface of the device object.

Returns the IpxCam::Interface object pointer for the camera device, associated with the DeviceInfo object

**Returns**

Returns the Interface

#### 7.7.3.2 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

#### 7.7.3.3 GetVendor()

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

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

**Returns**

Returns the camera device vendor name

#### 7.7.3.4 GetModel()

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

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

**Returns**

Returns the Camera device model name

#### 7.7.3.5 GetDisplayName()

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

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

**Returns**

Returns the name of the Camera device

#### 7.7.3.6 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

#### 7.7.3.7 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

#### 7.7.3.8 GetVersion()

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

This method returns the device version of the device object.

**Returns**

Returns the Device version

#### 7.7.3.9 GetAccessStatus()

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

Returns the device access status.

This method returns the information about the current access status of the Camera device

**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.
- **`IpSubnetMismatch`** [1001] - The device is available, but IP address does not match to the host subnet mask.

#### 7.7.3.10 GetUSB3HostInfo()

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

Returns the information about USB3 host controller.

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

**Returns**

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

#### 7.7.3.11 GetIPAddress()

```
virtual const char* IpxCam::DeviceInfo::GetIPAddress (
            IpxCamErr * err )  [pure virtual]
```

Returns the IP address of the GEV camera.

This method returns the IP address of the GEV camera, retrieved from DISCOVERY_ACK packet, received from the camera device

**Parameters**

|  |  |  |
|---|---|---|
| out | *err* | Error code |

**Returns**

Returns IP Address string or nullptr for non-GEV camera

#### 7.7.3.12 GetIPMask()

```
virtual const char* IpxCam::DeviceInfo::GetIPMask (
            IpxCamErr * err )  [pure virtual]
```

Returns the IP subnet mask of the GEV camera.

This method returns the IP subnet mask of the GEV camera, retrieved from DISCOVERY_ACK packet, received from the camera device

**Parameters**

|  |  |  |
|---|---|---|
| out | *err* | Error code |

**Returns**

Returns IP subnet mask string or nullptr for non-GEV camera

#### 7.7.3.13 GetIPGateway()

```
virtual const char* IpxCam::DeviceInfo::GetIPGateway (
            IpxCamErr * err )  [pure virtual]
```

Returns the IP gateway of GEV camera.

This method returns the IP gateway of the GEV camera, retrieved from DISCOVERY_ACK packet, received from the camera device

**Parameters**

|  |  |  |
|---|---|---|
| out | *err* | Error code |

**Returns**

Returns IP gateway string or nullptr for non-GEV camera

#### 7.7.3.14 GetIP()

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

Gets IP information from the GEV camera.

This method returns the IP address, netmask, and gateway of the GEV camera, from DISCOVERY_ACK packet, received from the camera

**Parameters**

| | | |
|---|---|---|
| out | *addr* | IP Address |
| out | *netmask* | IP Address subnet mask |
| out | *gateway* | Gateway address |

**Returns**

Returns Error code

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

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

Set the IP address to GEV camera.

This method sets the specified IP address to the GEV camera, using ForceIP GVCP command

**Parameters**

| | | |
|---|---|---|
| in | *addr* | IP Address string to set |
| in | *netmask* | IP Address subnet mask string |
| in | *gateway* | Gateway address string |

**Returns**

Returns Error code

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

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

Set IP address to GEV camera.

This method sets the specified IP address to the GEV camera, using ForceIP GVCP command

**Parameters**

| | | |
|---|---|---|
| in | *addr* | IP Address to set (host byte order) |
| in | *netmask* | IP Address subnet mask (host byte order) |
| in | *gateway* | Gateway address (host byte order) |

**Returns**

Returns Error code

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

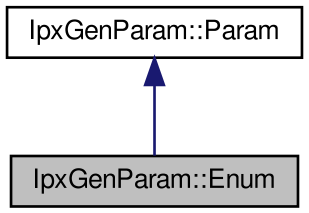
- IpxCameraApi.h

## 7.8 IpxGenParam::Enum Class Reference

A class containing methods for Enumeration GenICam camera parameter.

```
#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 as Integer.*
- virtual const char ∗ GetValueStr (IpxCamErr ∗err=nullptr)=0

  *This method gets the Enum Entry node value as String.*
- virtual IpxCamErr SetValue (int64_t val)=0

  *This method sets the Enum Entry node value as Integer.*
- virtual IpxCamErr SetValueStr (const char ∗val)=0

  *This method sets the Enum Entry node as String.*

### 7.8.1 Detailed Description

A class containing methods for Enumeration GenICam camera parameter.

A class containing methods to access the Enumeration GenICam camera parameter, using Integer or String value.

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

### 7.8.2 Member Function Documentation

#### 7.8.2.1 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.

#### 7.8.2.2 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the number of EnumEntries<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the number of enum entry nodes.

#### 7.8.2.3 GetEnumEntryByIndex()

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

This method gets the Enum Entry node by the Index number.

**Parameters**

|  |  |  |
|---|---|---|
| in | *aIndex* | Index number |

**Returns**

If the method succeeds, it returns the Enum Entry node.

#### 7.8.2.4 GetEnumEntryByName()

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

This method gets the Enum Entry node by Name.

**Parameters**

|  |  |  |
|---|---|---|
| in | *name* | Entry Name |

**Returns**

If the method succeeds, it returns the Enum Entry node.

#### 7.8.2.5 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.

#### 7.8.2.6 GetValue()

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

This method gets the Enum Entry node value as Integer.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Enum Entry node value<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

If the method succeeds, it returns the Enum Entry node value.

#### 7.8.2.7 GetValueStr()

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

This method gets the Enum Entry node value as String.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully get the Enum Entry node string<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

If the method succeeds, it returns the Enum Entry node string.

#### 7.8.2.8 SetValue()

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

This method sets the Enum Entry node value as Integer.

**Parameters**

| | | |
|---|---|---|
| in | *val* | Enum 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

#### 7.8.2.9 SetValueStr()

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

This method sets the Enum Entry node as String.

**Parameters**

| | | |
|---|---|---|
| in | *val* | Enum Entry node String |

**Returns**

Returns the error code

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

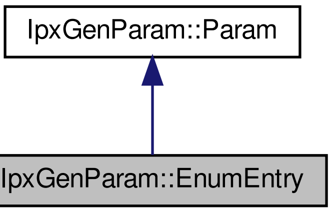
- IpxCameraApi.h

## 7.9 IpxGenParam::EnumEntry Class Reference

EnumEntry class represents the entry of GenICam Enum parameter.

```
#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.*

## 7.9.1 Detailed Description

EnumEntry class represents the entry of GenICam Enum parameter.

A Class for GenICam Enum Entries has methods to access the Enumeration GenICam parameter entry.

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".

## 7.9.2 Member Function Documentation

#### 7.9.2.1 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.

#### 7.9.2.2 GetValue()

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

This method gets the EnumEntry numerical value.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully indicates EnumEntry value was retrieved<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

If the method succeeds, it returns the value read of the EnumEntry.

#### 7.9.2.3 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully indicates EnumEntry string value was retrieved<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**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:

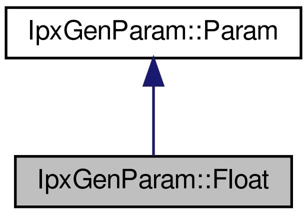
- IpxCameraApi.h

## 7.10 IpxGenParam::Float Class Reference

A class containing methods for Float GenICam camera parameter.

```
#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.*

### 7.10.1 Detailed Description

A class containing methods for Float GenICam camera parameter.

A class containing methods to access the Float GenICam camera parameter as floating point value.

For example, the picture below illustrates the float "ExposureTime".

### 7.10.2 Member Function Documentation

#### 7.10.2.1 GetType()

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

This method returns the node object Float type.

**Returns**

Returns the parameter type

Implements IpxGenParam::Param.

#### 7.10.2.2 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

#### 7.10.2.3 GetValue()

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

This method gets the Float node value.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully get the Float value<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Gets the Float node value

#### 7.10.2.4 GetMin()

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

This method gets the minimum value.

**Parameters**

|  |  |  |
|---|---|---|
| out | *err* | returns error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Minimum float value <br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the minimum

#### 7.10.2.5 GetMax()

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

This method gets the maximum value.

**Parameters**

|  |  |  |
|---|---|---|
| out | *err* | returns error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Maximum float value <br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the maximum

#### 7.10.2.6 GetUnit()

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

This method gets the Unit.

**Parameters**

| | | |
|---|---|---|
| `out` | *err* | returns error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the units<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the measurement unit string

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

- IpxCameraApi.h

## 7.11 IpxGui::IIpxGenParamTreeView Class Reference

IIpxGenParamTreeView class represents the GenICam parameters node tree panel.

`#include <IpxCameraGuiApi.h>`

### Public Member Functions

- virtual ∼IIpxGenParamTreeView ()
  *A destructor of the IIpxGenParamTreeView class.*
- virtual void setParams (IpxGenParam::Array ∗genParam)=0
  *Sets the IpxGenParam::Array object to the node tree GUI.*
- virtual void setParams (IPX_GENAPI_NS::INodeMap ∗nodemap)=0
  *Sets the GenApi::INodeMap object to the node tree GUI.*
- virtual void clearParams ()=0
  *Clears the parameters of the node tree GUI.*
- 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
  *Saves the current state of the Tree View.*
- virtual void loadState (const char ∗state)=0
  *Loads the state of the Tree View.*
- virtual void setPollingTime (uint64_t pollingTime)=0
  *Sets the polling time.*
- virtual uint64_t getPollingTime ()=0
  *Retrieves current polling time.*
- virtual void enablePolling (bool enable)=0
  *Enables the polling.*
- virtual bool isPollingEnabled ()=0
  *Retrieves current polling state.*

### 7.11.1 Detailed Description

IIpxGenParamTreeView class represents the GenICam parameters node tree panel.

The IIpxGenParamTreeView class is composed of functions to set and clear parameters of the GenICam parameters node tree of the camera. The node tree can be set with the current parameters stored in the IpxGenParam::Array and GenApi::INodeMap class.

For example, we can declare the instance of IpxGui::IIpxGenParamTreeView class as m_parameterView as shown below:

```
IpxGui::IIpxGenParamTreeView* m_ParameterView;
```

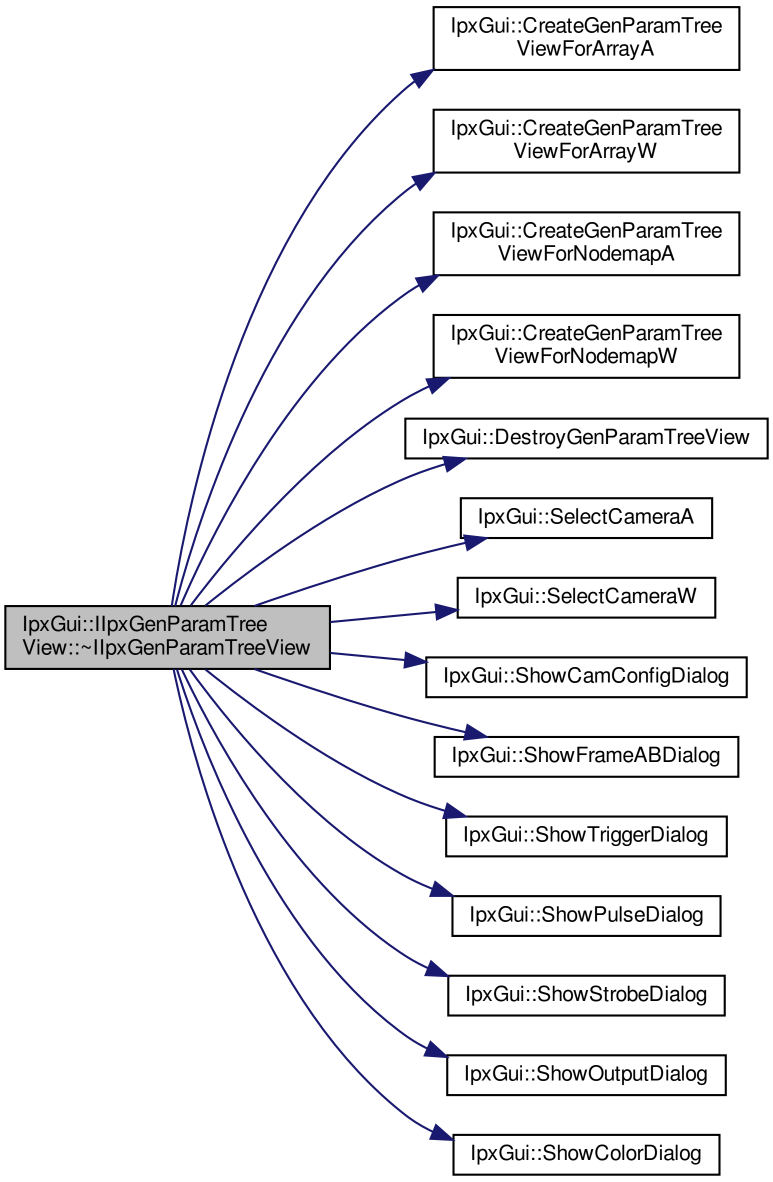
### 7.11.2 Constructor & Destructor Documentation

#### 7.11.2.1 ∼IIpxGenParamTreeView()

```
virtual IpxGui::IIpxGenParamTreeView::∼IIpxGenParamTreeView ( )  [inline], [virtual]
```

A destructor of the IIpxGenParamTreeView class.

Destroys the IIpxGenParamTreeView object and all its descendants. Here is the call graph for this function:


IpxGui::IIpxGenParamTree View::~IIpxGenParamTreeView
IpxGui::CreateGenParamTree ViewForArrayA
IpxGui::CreateGenParamTree ViewForArrayW
IpxGui::CreateGenParamTree ViewForNodemapA
IpxGui::CreateGenParamTree ViewForNodemapW
IpxGui::DestroyGenParamTreeView
IpxGui::SelectCameraA
IpxGui::SelectCameraW
IpxGui::ShowCamConfigDialog
IpxGui::ShowFrameABDialog
IpxGui::ShowTriggerDialog
IpxGui::ShowPulseDialog
IpxGui::ShowStrobeDialog
IpxGui::ShowOutputDialog
IpxGui::ShowColorDialog

### 7.11.3 Member Function Documentation

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

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

Sets the IpxGenParam::Array object to the node tree GUI.

This method sets the parameters of the node tree by the information extracted from the IpxGenParam::Array class

**Parameters**

| | | |
|---|---|---|
| in | *genParam* | The pointer to the IpxGenParam::Array 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());
}
```

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

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

Sets the GenApi::INodeMap object to the node tree GUI.

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 | *nodemap* | The pointer to the GenApi::INodeMap 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);
...
```

### 7.11.3.3 clearParams()

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

Clears the parameters of the node tree GUI.

This method clears the parameters of the node tree that have been set by the instance of the IpxGui::IIpxGenParam←TreeView 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();
```

### 7.11.3.4 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 visibility level for the feature

**Returns**

Visibility value

#### 7.11.3.5 setVisibility()

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

This method sets visibility mode.

It sets the current visibility level for the feature.

**Parameters**

| | | |
|---|---|---|
| in | *visibility* | The visibility mode value to set |

**Returns**

Void.

#### 7.11.3.6 saveState()

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

Saves the current state of the Tree View.

This method creates the string, representing the current state of the Tree View, and returns the pointer to this string.

**Returns**

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

#### 7.11.3.7 loadState()

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

Loads the state of the Tree View.

This method loads the state of the Tree View using the string, created by saveState() method. The individual node can be in expanded or collapse state.

**Parameters**

| | | |
|---|---|---|
| in | *state* | State string to be loaded. The string consists of sub-string values separated by the token. |

#### 7.11.3.8 setPollingTime()

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

Sets the polling time.

This method sets the value of the parameters pooling time. Polling should be enabled by enablePolling() function

**Parameters**

| | | |
|---|---|---|
| in | *pollingTime* | time in msec to be set |

#### 7.11.3.9 getPollingTime()

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

Retrieves current polling time.

This method retrieves the value of the parameters 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:

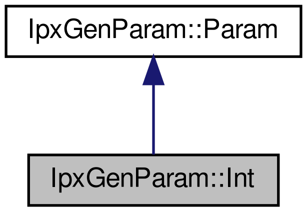
- IpxCameraGuiApi.h

## 7.12 IpxGenParam::Int Class Reference

A class containing methods for Integer GenICam camera parameter.

```
#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.*

### 7.12.1 Detailed Description

A class containing methods for Integer GenICam camera parameter.

A class containing methods to access the Integer GenICam camera parameter as integer value.

For example, the mapping below illustrates "Width" Integer parameter.

### 7.12.2 Member Function Documentation

#### 7.12.2.1 GetType()

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

This method returns the node object Int type.

**Returns**

Returns the parameter type

Implements IpxGenParam::Param.

#### 7.12.2.2 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

#### 7.12.2.3 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: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Int value <br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the Int node value

#### 7.12.2.4 GetMin()

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

This method gets the minimum value.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code: <br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Minimum int value <br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node <br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the minimum

#### 7.12.2.5 GetMax()

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

This method gets the maximum value.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the Maximum int value<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the maximum

#### 7.12.2.6 GetIncrement()

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

This method gets the Increment value.

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns error code :<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the increment value<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the increment

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

- IpxCameraApi.h

## 7.13 IpxCam::Interface Class Reference

The Interface class represents a interface module in the GenTL module hierarchy.

`#include <IpxCameraApi.h>`

### Public Member Functions

- virtual ∼Interface ()

  *Interface class destructor.*
- virtual DeviceInfoList ∗ GetDeviceInfoList ()=0

  *This method retrieves the list of DeviceInfo objects for the camera 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 DeviceInfo object pointer 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 returns the description of the interface.*
- virtual InterfaceType GetType ()=0

  *This method gets the type of interface.*
- virtual const char ∗ GetId ()=0

  *This method gets the identifier of the interface .*
- virtual const char ∗ GetVersion ()=0

  *This method gets the version of Interface driver.*
- virtual IpxCamErr RegisterEvent2 (uint32_t eventType, IpxCam::EventCallback2 ∗eventCallback, void ∗p←Private)=0

  *This method registers the Interface class method as a callback method to be called when a eventType occurs.*
- virtual IpxCamErr RegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗pPrivate)=0

  *This method registers the Interface class method as a callback method to be called when a eventType occurs.*
- virtual IpxCamErr UnRegisterEvent2 (uint32_t eventType, IpxCam::EventCallback2 ∗eventCallback, void ∗p←Private)=0

  *This method unregisters the Interface class callback method for the eventType.*
- virtual IpxCamErr UnRegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗p←Private)=0

  *This method unregisters the Interface class callback method for the eventType.*
- virtual IpxGenParam::Array ∗ GetParameters (IpxCamErr ∗err=nullptr)=0

  *This method returns the parameter array used to control the Imperx Camera device.*
- virtual Device ∗ CreateDeviceFromConfig (const char ∗fileName, IpxCamErr ∗err=nullptr)=0

  *Creates the Device object from configuration file.*

## 7.13.1 Detailed Description

The Interface class represents a interface module in the GenTL module hierarchy.

This class represents an individual physical interface in the System. For example, a network interface card (NIC) for GigE Vision connection, CXP or Camera Link frame grabber board, or USB3 Vision driver in the GenTL system. The Interface class includes methods to enumerate the available devices on the physical interface in the system.

## 7.13.2 Constructor & Destructor Documentation

#### 7.13.2.1 ∼Interface()

`virtual IpxCam::Interface::∼Interface ( ) [inline], [virtual]`

Interface class destructor.

Destroys the Interface object and all its descendants.

### 7.13.3 Member Function Documentation

#### 7.13.3.1 GetDeviceInfoList()

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

This method retrieves the list of DeviceInfo objects for the camera 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;
   }
}
```

#### 7.13.3.2 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;
```

#### 7.13.3.3 GetDeviceInfoById()

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

This method retrieves the DeviceInfo object pointer for the specified device identifier.

**Parameters**

| | | |
|---|---|---|
| in | *device↩ Id* | Device identifier |

**Returns**

Returns the pointer to DeviceInfo object or nullptr if no such device found

#### 7.13.3.4 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 | *pChanged* | Change in Device |
| in | *iTimeout* | Timeout allowed to search for available camera devices |

**Returns**

Returns error code

### 7.13.3.5 GetDescription()

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

This method returns the description of the interface.

The GetDescription method gets the user readable string description of the interface.

**Returns**

Returns the Description of the interface

### 7.13.3.6 GetType()

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

This method gets the type of interface.

The GetType method returns the Interface Type (Transport Layer Technology) of this interface object

**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,
};
```

#### 7.13.3.7 GetId()

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

This method gets the identifier of the interface .

The GetId method returns the interface identifier that could be used to instantiate the interface object

**Returns**

Returns interface identifier

#### 7.13.3.8 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

#### 7.13.3.9 RegisterEvent2()

```
virtual IpxCamErr IpxCam::Interface::RegisterEvent2 (
             uint32_t eventType,
             IpxCam::EventCallback2 * 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 | *eventType* | Event Type |
| in | *eventCallback* | pointer to event CallBack method |
| in | *pPrivate* | pointer to user's data |

**Returns**

Returns Error code

#### 7.13.3.10 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.

**Deprecated** Use RegisterEvent2 instead

#### 7.13.3.11 UnRegisterEvent2()

```
virtual IpxCamErr IpxCam::Interface::UnRegisterEvent2 (
            uint32_t eventType,
            IpxCam::EventCallback2 * eventCallback,
            void * pPrivate )  [pure virtual]
```

This method unregisters the Interface class callback method for the eventType.

**Parameters**

| | | |
|---|---|---|
| in | *eventType* | Event Type |
| in | *eventCallback* | pointer to event CallBack method |
| in | *pPrivate* | pointer to user's data |

**Returns**

Returns Error code

#### 7.13.3.12 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.

**Deprecated** Use UnRegisterEvent2 instead

#### 7.13.3.13 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 | *err* | returns error code |

**Returns**

Returns the pointer to IpxGenParam::Array object, used to control the Imperx Camera device

#### 7.13.3.14 CreateDeviceFromConfig()

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

Creates the Device object from configuration file.

This method creates, configures and sets up the device using the information retrieved from the specified configuration file

**Parameters**

| | | |
|---|---|---|
| in | *fileName* | Configuration file to open |
| out | *err* | 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

## 7.14 IpxCam::List< _T > Class Template Reference

The List class is used as list-like container for 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.*

### 7.14.1 Detailed Description

**template<class _T>**
**class IpxCam::List< _T >**

The List class is used as list-like container for the specified template type objects.

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

They can be declared as follows:

| | |
|---|---|
| **IpxCam::List<Interface> ∗interfaceList** | **This class represents the list of Interface objects.** |
| **IpxCam::List<Device> ∗deviceList** | **This class represents the list of Device objects.** |
| **IpxCam::List<DeviceInfo> ∗deviceInfoList** | **This class represents the list of DeviceInfo objects.** |
| **IpxCam::List<Stream> ∗streamList** | **This class represents the list of Data Stream objects** |
| **IpxCam::List<Buffer> ∗bufferList** | **This class represents the list of Buffer objects** |

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

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

They can be declared as follows:

| | |
|---|---|
| **InterfaceList ∗interfaceList** | **This class represents the list of Interface objects.** |
| **DeviceInfoList ∗deviceInfoList** | **This class represents the list of DeviceInfo 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()** function 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();
```

### 7.14.2 Member Typedef Documentation

#### 7.14.2.1 elem_type

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

Element Type

### 7.14.3 Constructor & Destructor Documentation

#### 7.14.3.1 ∼List()

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

A destructor of the List class.

Destructor. Destroys the List object and all its descendants.

### 7.14.4 Member Function Documentation

#### 7.14.4.1 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.

#### 7.14.4.2 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.

#### 7.14.4.3 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.

#### 7.14.4.4 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:

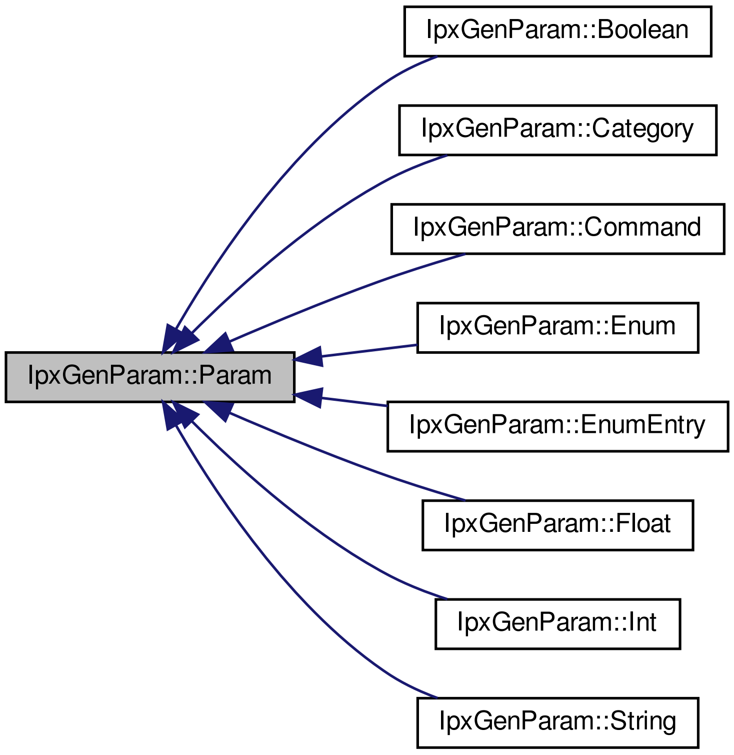
- IpxCameraApi.h

## 7.15 IpxGenParam::Param Class Reference

General class for GenICam parameter.

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

Inheritance diagram for IpxGenParam::Param:



### Public Member Functions

- virtual ∼Param ()

  *Param class destructor. Destroys the Param and all its descendants.*
- 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 string to be used for the parameter displaying.*
- 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 writable.*
- 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 unregisters 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.*

### 7.15.1 Detailed Description

General class for GenICam parameter.

Class for accessing the GenICam feature node of the Camera parameters

## 7.15.2 Constructor & Destructor Documentation

#### 7.15.2.1 ∼Param()

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

Param class destructor. Destroys the Param and all its descendants.

Param class destructor.

## 7.15.3 Member Function Documentation

#### 7.15.3.1 GetType()

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

This method returns the Parameter Node Type.

**Returns**

return the parameter type.

Implemented in IpxGenParam::String, IpxGenParam::Int, IpxGenParam::Float, IpxGenParam::Enum, IpxGenParam::↩
EnumEntry, IpxGenParam::Command, IpxGenParam::Boolean, and IpxGenParam::Category.

#### 7.15.3.2 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.

#### 7.15.3.3 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.

#### 7.15.3.4 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.

#### 7.15.3.5 GetDisplayName()

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

This method returns the string to be used for the parameter displaying.

**Returns**

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

#### 7.15.3.6 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.

#### 7.15.3.7 IsValueCached()

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

This method checks if the parameter node is cached.

**Returns**

True if the value is cached. False if the value is not cached.

#### 7.15.3.8 IsAvailable()

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

This method checks if parameter node is available.

**Returns**

True if the parameter node is available. False if it is not available.

#### 7.15.3.9 IsWritable()

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

This method checks if parameter node is writable.

**Returns**

True if the parameter node is writable. False if it is not writable.

#### 7.15.3.10 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. False if it is not readable.

#### 7.15.3.11 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. False if it is not streamable.

#### 7.15.3.12 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. False if it is not visible.

#### 7.15.3.13 RegisterEventSink()

```
virtual IpxCamErr IpxGenParam::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

#### 7.15.3.14 UnregisterEventSink()

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

This method unregisters the event.

**Parameters**

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

**Returns**

Returns the Error code:

- `IpxCamErr::IPX_CAM_ERR_OK` - Successfully unregisters event sink

#### 7.15.3.15 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.

#### 7.15.3.16 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

#### 7.15.3.17 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

#### 7.15.3.18 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

#### 7.15.3.19 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

#### 7.15.3.20 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

#### 7.15.3.21 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

#### 7.15.3.22 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

#### 7.15.3.23 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

## 7.16 IpxGenParam::ParamEventSink Class Reference

A Class for ParamEventSink notifications handling.

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

### Public Member Functions

- virtual ∼ParamEventSink ()

  *ParamEventSink class destructor. Destroys the ParamEventSink object and all its descendants.*
- virtual void OnParameterUpdate (Param ∗param)=0

  *Update Parameter Node.*

## 7.16.1 Detailed Description

A Class for ParamEventSink notifications handling.

An Event Sink class designed to receive the notifications from the GenICam parameter Node Updates

## 7.16.2 Member Function Documentation

#### 7.16.2.1 OnParameterUpdate()

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

Update Parameter Node.

**Parameters**

| | | |
|---|---|---|
| in | *param* | The pointer to the Param class node |

**Returns**

Void.

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

- IpxCameraApi.h

## 7.17 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

  *Creates the buffer in the data stream object.*
- virtual IpxCam::Buffer ∗ SetBuffer (void ∗pBuffer, size_t iSize, void ∗pPrivate, IpxCamErr ∗err)=0

  *Sets memory buffer to create the Buffer object.*
- virtual IpxCamErr RevokeBuffer (IpxCam::Buffer ∗buff)=0

  *Revokes any announced buffer.*
- virtual IpxCamErr QueueBuffer (IpxCam::Buffer ∗buff)=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

  *Terminates the waiting operation on a previously queued Buffer.*
- 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

  *Starts the Acquisition Engine.*
- virtual IpxCamErr StopAcquisition (uint32_t flags=0)=0

  *Stops the stream's acquisition engine.*
- virtual IpxCamErr AllocBufferQueue (void ∗pPrivate, size_t iNum)=0

  *Allocates the Buffer Queue.*
- virtual IpxCamErr ReleaseBufferQueue ()=0

  *Releases the Buffer Queue.*
- virtual size_t GetBufferQueueSize ()=0

  *Retrieves the Buffer Queue size.*
- virtual IpxCamErr RegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗pPrivate)=0

  *Registers the EventCallback.*
- virtual IpxCamErr UnRegisterEvent (uint32_t eventType, IpxCam::EventCallback ∗eventCallback, void ∗p←
  Private)=0

  *Unregisters the EventCallback.*
- virtual IpxGenParam::Array ∗ GetParameters (IpxCamErr ∗err=nullptr)=0

  *Returns the GenICam parameters array.*
- virtual uint64_t GetNumDelivered ()=0

  *Returns the number of the delivered buffers.*
- virtual uint64_t GetNumUnderrun ()=0

  *Returns the number under-run frames.*
- virtual size_t GetNumAnnounced ()=0

  *Returns the number of announced buffers.*
- virtual size_t GetNumQueued ()=0

  *Returns the number of queued buffers.*
- virtual size_t GetNumAwaitDelivery ()=0

  *Returns the number of buffers awaiting delivery.*

- virtual size_t GetBufferSize ()=0

  *Returns the buffer size.*
- virtual bool IsGrabbing ()=0

  *This method returns a flag indicating if the data stream is grabbing or not.*
- virtual size_t GetMinNumBuffers ()=0

  *Returns the minimum number of buffers to be announced.*
- virtual size_t GetBufferAlignment ()=0

  *Returns the buffer alignment size.*

### 7.17.1 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.

### 7.17.2 Constructor & Destructor Documentation

#### 7.17.2.1 ∼Stream()

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

A destructor of the Stream class.

Destroys the Stream object and all its descendants.

### 7.17.3 Member Function Documentation

#### 7.17.3.1 Release()

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

This method releases the instance of the stream object.

**Returns**

void

#### 7.17.3.2 CreateBuffer()

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

Creates the buffer in the data stream object.

This method allocates the memory for a buffer and announces this buffer to the data stream

**Parameters**

| | | |
|---|---|---|
| in | *iSize* | Size of the buffer |
| in | *pPrivate* | pointer to private data (user's data) which will be passed to the GenTL Consumer |
| out | *err* | returns Error code |

**Returns**

Returns Buffer object pointer of the announced buffer

### 7.17.3.3 SetBuffer()

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

Sets memory buffer to create the Buffer object.

This method is used to set the user-allocated memory buffer to create the Buffer object and announce it to the data stream.

**Parameters**

| | | |
|---|---|---|
| in | *pBuffer* | buffer |
| in | *iSize* | size of Buffer |
| in | *pPrivate* | pointer to user's data |
| out | *err* | returns Error code |

**Returns**

returns Buffer object pointer

### 7.17.3.4 RevokeBuffer()

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

Revokes any announced buffer.

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

**Parameters**

| | | |
|---|---|---|
| in | *buff* | Buffer object pointer |

**Returns**

Returns Error code

#### 7.17.3.5 QueueBuffer()

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

This method queues specified buffers.

During the acquisition, this method is used to return the specified buffer to the acquisition engine's queue

**Parameters**

| | | |
|---|---|---|
| in | *buff* | Buffer object pointer |

**Returns**

Returns Error code

#### 7.17.3.6 GetBuffer()

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

This method retrieves the buffer object.

Retrieves the next acquired buffer entry from the acquisition engine's queue and returns the acquired Buffer object

**Parameters**

| | | |
|---|---|---|
| in | *iTimeout* | timeout in ms |
| in | *err* | error code |

**Returns**

Returns the pointer to the acquired Buffer object

### 7.17.3.7 CancelBuffer()

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

Terminates the waiting operation on a previously queued Buffer.

This method cancels the waiting operation on a previously queued Buffer in the acquisition engine's queue

**Returns**

Returns Error code

### 7.17.3.8 FlushBuffers()

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

This method flushes the buffers of the data stream object.

Performs the specified Flush Operation on the acquisition engine's queue. Operations type is defined in FlushOperations enum.

**Parameters**

| | | |
|---|---|---|
| in | *operation* | FlushOperation |

**Returns**

Returns Error code

### 7.17.3.9 StartAcquisition()

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

Starts the Acquisition Engine.

This method starts the acquisition engine of the stream to acquire the image data frames to the queued buffers

**Parameters**

| | | |
|---|---|---|
| in | *iNumFramesToAcquire* | number of Frames to Acquire. Set UINT64_MAX for the infinite acquisition |
| in | *flags* | flags. Set to 0 by default |

**Returns**

Returns Error code

#### 7.17.3.10 StopAcquisition()

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

Stops the stream's acquisition engine.

This method stops the acquisition engine of the stream and terminates the image data frames acquisition

**Parameters**

| | | |
|---|---|---|
| in | *flags* | flags:<br>• ACQ_STOP_FLAGS_DEFAULT=0, Stop the acquisition engine when the currently running tasks like filling a buffer are completed (default behavior).<br>• ACQ_STOP_FLAGS_KILL=1, Stop the acquisition engine immediately and leave buffers currently being filled in the Input Buffer Pool. |

**Returns**

Returns Error code

#### 7.17.3.11 AllocBufferQueue()

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

Allocates the Buffer Queue.

This method allocates the buffers in the queue of the acquisition engine of the data stream object.

**Parameters**

| | | |
|---|---|---|
| in | *pPrivate* | pointer to user's data |
| in | *iNum* | number of the buffers to allocate |

**Returns**

Returns Error code

#### 7.17.3.12 ReleaseBufferQueue()

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

Releases the Buffer Queue.

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

**Returns**

Returns Error code

#### 7.17.3.13 GetBufferQueueSize()

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

Retrieves the Buffer Queue size.

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

**Returns**

Returns the Buffer Queue size

#### 7.17.3.14 RegisterEvent()

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

Registers the EventCallback.

This method registers the data Stream class method as a callback method to be called when event of the specified type occurs.

**Parameters**

| | | |
|---|---|---|
| in | *eventType* | Event Type |
| in | *eventCallback* | event CallBack function pointer |
| in | *pPrivate* | pointer to the user's data |

**Returns**

Returns Error code

### 7.17.3.15 UnRegisterEvent()

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

Unregisters the EventCallback.

This method unregisters the data Stream class callback method for the specified event type

**Parameters**

| | | |
|---|---|---|
| in | *eventType* | Event Type |
| in | *eventCallback* | event CallBack function pointer |
| in | *pPrivate* | pointer to the user's data |

**Returns**

Returns Error code

### 7.17.3.16 GetParameters()

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

Returns the GenICam parameters array.

This method returns the pointer to IpxGenParam::Array object of the GenICam parameters array for the data stream object

**Parameters**

| | | |
|---|---|---|
| out | *err* | returns the error code |

**Returns**

Returns the data stream GenICam parameters array

#### 7.17.3.17 GetNumDelivered()

```
virtual uint64_t IpxCam::Stream::GetNumDelivered ( ) [pure virtual]
```

Returns the number of the delivered buffers.

This method returns the number of the delivered buffers since the start of the last acquisition

**Returns**

Returns the number of the delivered buffers

#### 7.17.3.18 GetNumUnderrun()

```
virtual uint64_t IpxCam::Stream::GetNumUnderrun ( ) [pure virtual]
```

Returns the number under-run frames.

This method returns the number of the lost frames due to the acquisition queue being under-run.

**Returns**

Returns the number of lost frames due to queue under-run

#### 7.17.3.19 GetNumAnnounced()

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

Returns the number of announced buffers.

This method returns the number of announced buffers in the data stream acquisition queue

**Returns**

Returns number of announced buffers

#### 7.17.3.20 GetNumQueued()

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

Returns the number of queued buffers.

This method returns the number of queued buffers in the data stream object acquisition queue

**Returns**

Returns the number of buffers in the input pool and the number of buffers currently being filled

#### 7.17.3.21 GetNumAwaitDelivery()

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

Returns the number of buffers awaiting delivery.

This method returns the number of buffers awaiting the delivery from the data stream object acquisition queue to the client application

**Returns**

Returns the number of buffers in the output buffer queue

#### 7.17.3.22 GetBufferSize()

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

Returns the buffer size.

This method returns the buffer size of the data stream object.

**Returns**

Returns the buffer size

#### 7.17.3.23 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 stared. Otherwise, the acquisition engine is off.

#### 7.17.3.24 GetMinNumBuffers()

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

Returns the minimum number of buffers to be announced.

This method returns the minimum number of buffers to be announced in the data stream object acquisition queue to perform the grabbing

**Returns**

Returns the minimum number of buffers to announce

#### 7.17.3.25 GetBufferAlignment()

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

Returns the buffer alignment size.

This method returns the alignment size of the buffers in the stream object acquisition queue

**Returns**

Returns the alignment size in bytes of the stream buffers

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

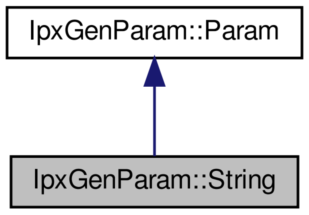
- IpxCameraApi.h

## 7.18 IpxGenParam::String Class Reference

A class containing methods for String GenICam camera parameter.

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

Inheritance diagram for IpxGenParam::String:



### Public Member Functions

- virtual ParamType GetType ()

  *This method returns the node object String 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.*

### 7.18.1 Detailed Description

A class containing methods for String GenICam camera parameter.

A class containing methods to access the String GenICam camera parameter as zero-terminated array of characters

For example, the image below illustrates "DeviceModelName" parameter.

### 7.18.2 Member Function Documentation

#### 7.18.2.1 GetType()

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

This method returns the node object String type.

**Returns**

The parameter type

Implements IpxGenParam::Param.

#### 7.18.2.2 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the maximum length value<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

gets the maximum length of the string

#### 7.18.2.3 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:<br>• `IpxCamErr::IPX_CAM_ERR_OK` - Successfully gets the string<br>• `IpxCamErr::IPX_CAM_GENICAM_ACCESS_ERROR` - Unable to access genicam specified node<br>• `IpxCamErr::IPX_CAM_GENICAM_TYPE_ERROR` - Unable to access genicam specified node type |

**Returns**

Returns the value

#### 7.18.2.4 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

## 7.19 IpxCam::System Class Reference

The System class represents an abstraction of the System module of the GenTL module hierarchy.

`#include <IpxCameraApi.h>`

### Public Member Functions

- virtual ∼System ()

  *System class Destructor.*
- virtual void Release ()=0

  *This method releases the instance of the system object.*
- virtual InterfaceList ∗ GetInterfaceList (InterfaceType type=AllInterfaces)=0

  *This method returns the list of all the interfaces of the system object.*
- virtual Interface ∗ GetInterfaceById (const char ∗ifaceId)=0

  *Retrieves the interface specified by interface identifier.*
- virtual const char ∗ GetDisplayName ()=0

  *Retrieves the name of the GenTL Producer.*
- virtual const char ∗ GetVersion ()=0

  *Returns the GenTL Producer version.*
- virtual Device ∗ CreateDeviceFromConfig (const char ∗fileName, IpxCamErr ∗err=nullptr)=0

  *Creates the Device object from configuration file.*
- virtual IpxCamErr RegisterGenTLProvider (const char ∗fileName)=0

  *Registers the GenTL CTI library.*

## 7.19.1 Detailed Description

The System class represents an abstraction of the System module of the GenTL module hierarchy.

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 be used to obtain **IpxCam::InterfaceList**, then get the list **IpxCam::DeviceInfo** objects on the **IpxCam::Interface**, and create **IpxCam::Device** object, representing the camera device .

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* version_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();
}
```

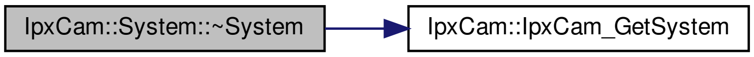
## 7.19.2 Constructor & Destructor Documentation

### 7.19.2.1 ∼System()

`virtual IpxCam::System::∼System ( ) [inline], [virtual]`

System class Destructor.

Destroys the System object and all its descendants. Here is the call graph for this function:



## 7.19.3 Member Function Documentation

### 7.19.3.1 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();
}
```

### 7.19.3.2 GetInterfaceList()

```
virtual InterfaceList* IpxCam::System::GetInterfaceList (
              InterfaceType type = AllInterfaces )  [pure virtual]
```

This method returns the list of all the interfaces of the system object.

GetInterfaceList method lists all the available hardware interfaces with the transport layers technologies, supported by GenTL producer library

**Parameters**

| | | |
|---|---|---|
| in | *type* | 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();
```

#### 7.19.3.3 GetInterfaceById()

```
virtual Interface* IpxCam::System::GetInterfaceById (
              const char * ifaceId )  [pure virtual]
```

Retrieves the interface specified by interface identifier.

This method returns the interface by unique string identifier of the system object.

**Parameters**

| | | |
|---|---|---|
| in | *iface←Id* | 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.

#### 7.19.3.4 GetDisplayName()

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

Retrieves 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 string

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();
}
```

### 7.19.3.5 GetVersion()

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

Returns the GenTL Producer version.

This method returns the version of the GenTL Producer of the system object.

**Returns**

Returns the Version string

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* version_str = system->GetVersion();
  std::cout << "Version " << system->GetVersion();

  // some code here

  system->Release();
}
```

### 7.19.3.6 CreateDeviceFromConfig()

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

Creates the Device object from configuration file.

This method creates, configures and sets up the device using the information retrieved from the specified configuration file

**Parameters**

| | | |
|---|---|---|
| in | *fileName* | Configuration file to open |
| out | *err* | returns the error code |

**Returns**

Returns Device or nullptr if device cannot be instantiated

#### 7.19.3.7 RegisterGenTLProvider()

```
virtual IpxCamErr IpxCam::System::RegisterGenTLProvider (
             const char * fileName )  [pure virtual]
```

Registers the GenTL CTI library.

This method registers the 3rd party GenTL provider CTI library in the System.

**Parameters**

| | | |
|---|---|---|
| in | *fileName* | 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

# Index