DepthSense SDK

Generated by Doxygen 1.7.6.1

Mon Oct 14 2013 14:56:13

Contents

Chapter 1

The DepthSense SDK Reference Manual

1.1 Overview

DepthSense SDK provides an interface to the various SoftKinetic DepthSense cameras. By using the DepthSense library, you can configure one or more DepthSense cameras and capture depth, color and audio frame data.

Some of the DepthSense SDK features are:

- a type system transparently combining automatic memory management and polymorphism, and providing some reflection capabilities
- an elegant event-based framework for receiving frame data and server notifications
- a property system supporting reflection
- multi-camera and multi-client support

These features will be examined in more detail in the following sections.

1.2 The DepthSense SDK type system

The DepthSense SDK type system provides automatic memory management (which is implemented with traditional smart pointer mechanisms) while preserving the syntactical advantages of C++ polymorphism.

DepthSense SDK objects reside on the stack and are passed by value (like smart pointers), but offer polymorphism abilities similar to what C++ offers with heap-based pointers. The following examples compare C++ and DepthSense with commonly-used idioms.

1.2.1 Initializing an object variable to the unset state

1.2.1.1 C++ code

```
MyClass* obj = NULL;
```

1.2.1.2 DepthSense SDK code

```
MyClass obj;
```

1.2.2 Testing if an object variable is set

1.2.2.1 C++ code

```
bool b = obj != NULL;
```

1.2.2.2 DepthSense SDK code

```
bool b = obj.isSet();
```

1.2.3 Unsetting an object variable

1.2.3.1 C++ code

```
obj = NULL;
```

1.2.3.2 DepthSense SDK code

```
obj.unset();
```

1.2.4 Performing an upcast

1.2.4.1 C++ code

```
MyDerivedClass* derived = param1;
MyBaseClass* base = derived;
```

1.2.4.2 DepthSense SDK code

```
MyDerivedClass derived = param1;
MyBaseClass base = derived;
```

1.2.5 Calling a method of the most derived class

1.2.5.1 C++ code

```
obj->derivedMethod();
```

1.2.5.2 DepthSense SDK code

```
obj.derivedMethod();
```

1.2.6 Calling a method of a base class

1.2.6.1 C++ code

```
obj->baseMethod();
```

1.2.6.2 DepthSense SDK code

```
obj.baseMethod();
```

1.2.7 Testing the runtime type of an object

1.2.7.1 C++ code

```
MyBaseClass* base = param1;
bool b = dynamic_cast<MyDerivedClass*>(base) != NULL;
```

1.2.7.2 DepthSense SDK code

```
MyBaseClass base = param1;
bool b = base.is<MyDerivedClass>();
```

1.2.8 Performing a downcast which cannot fail

1.2.8.1 C++ code

```
// if the cast fails, derived is NULL and the code crashes
MyBaseClass* base = param1;
MyDerivedClass* derived = dynamic_cast<MyDerivedClass*>(base);
derived->someMethod();
```

1.2.8.2 DepthSense SDK code

```
// if the cast fails, std::bad_cast is thrown
MyBaseClass base = param1;
MyDerivedClass derived = (MyDerivedClass) base;
derived.someMethod();
```

1.2.9 Performing a downcast which can fail

1.2.9.1 C++ code

```
MyBaseClass* base = param1;
MyDerivedClass* derived = dynamic_cast<MyDerivedClass*>(base);
bool castSucceeded = derived != NULL;
```

1.2.9.2 DepthSense SDK code

```
MyBaseClass base = param1;
MyDerivedClass derived = base.as<MyDerivedClass>();
bool castSucceeded = derived.isSet();
```

1.2.10 Obtaining the name of the runtime type of an object

1.2.10.1 C++ code

```
// the name is implementation-defined and often mangled
const char* name = typeid(*obj).name();
```

1.2.10.2 DepthSense SDK code

```
// the name is formalized and of the form DepthSense.ColorNode
std::string name = obj.getType().name();
```

1.2.11 Obtaining the list of properties of an interface

1.2.11.1 C++ code

```
// C++ provides no such facility
```

1.2.11.2 DepthSense SDK code

```
std::vector<DepthSense::PropertyBase> properties = MyClass::type().
     getProperties();
```

1.3 The DepthSense SDK event system

DepthSense SDK features an event-based framework for providing the client application with frame data and server notifications.

The following example demonstrates how to capture the data coming from the first available color sensor attached to the host system.

```
// SoftKinetic DepthSense SDK
//
// COPYRIGHT AND CONFIDENTIALITY NOTICE - SOFTKINETIC CONFIDENTIAL
// INFORMATION
//
// All rights reserved to SOFTKINETIC SENSORS NV (a
// company incorporated and existing under the laws of Belgium, with
// its principal place of business at Boulevard de la Plainelaan 15,
// 1050 Brussels (Belgium), registered with the Crossroads bank for
// enterprises under company number 0811 341 454 - "Softkinetic
// Sensors").
//
// The source code of the SoftKinetic DepthSense Camera Drivers is
// proprietary and confidential information of Softkinetic Sensors NV.
//
// For any question about terms and conditions, please contact:
// info@softkinetic.com Copyright (c) 2002-2013 Softkinetic Sensors NV

#include <stdlib.h>
#include <iostream>

#include <DepthSense.hxx>

using namespace std;
using namespace DepthSense;

static void error (const char* message)
{
    cerr << message << endl;
    exit(1);
}
```

```
static ColorNode getFirstAvailableColorNode (Context context)
{
  // obtain the list of devices attached to the host
  vector<Device> devices = context.getDevices();

  for (vector<Device>::const_iterator iter = devices.begin(); iter != devices.
      end(); iter++)
  {
      Device device = *iter;

      // obtain the list of nodes of the current device
      vector<Node> nodes = device.getNodes();

      for (vector<Node>::const_iterator nodeIter = nodes.begin(); nodeIter !=
      nodes.end(); nodeIter++)
      {
          Node node = *nodeIter;

          // if the node is a DepthSense::ColorNode, return it
          ColorNode colorNode = node.as<ColorNode>();
          if (colorNode.isSet())
              return colorNode;
      }
  }

  // return an unset color node
  return ColorNode();
}

static void onNewColorSample (ColorNode obj, ColorNode::NewSampleReceivedData
      data)
{
    cout << "New color sample received (timeOfCapture=" << data.timeOfCapture <
      < ")" << endl;
}

int main (int argc, char** argv)
{
    // create a connection to the DepthSense server at localhost
    Context context = Context::create();

    // get the first available color sensor
    ColorNode colorNode = getFirstAvailableColorNode(context);

    // if no color node was found, fail
    if (! colorNode.isSet())
        error("no color node found");

    // enable the capture of the color map
    colorNode.setEnableColorMap(true);

    // connect a callback to the newSampleReceived event of the color node
    colorNode.newSampleReceivedEvent().connect(onNewColorSample);

    // add the color node to the list of nodes that will be streamed
    context.registerNode(colorNode);

    // start streaming
    context.startNodes();

    // start the DepthSense main event loop
    context.run();
}
```

1.4 The DepthSense SDK property system

Each DepthSense class (such as DepthSense::Device or DepthSense::ColorNode) defines properties of various types.

While the most natural way of querying and modifying the value of a property is to use the relevant accessor methods, DepthSense SDK also supports property reflection. - The following example demonstrates how to discover and display object properties at runtime.

```
// SoftKinetic DepthSense SDK
//
// COPYRIGHT AND CONFIDENTIALITY NOTICE - SOFTKINETIC CONFIDENTIAL
// INFORMATION
//
// All rights reserved to SOFTKINETIC SENSORS NV (a
// company incorporated and existing under the laws of Belgium, with
// its principal place of business at Boulevard de la Plainelaan 15,
// 1050 Brussels (Belgium), registered with the Crossroads bank for
// enterprises under company number 0811 341 454 - "Softkinetic
// Sensors").
//
// The source code of the SoftKinetic DepthSense Camera Drivers is
// proprietary and confidential information of Softkinetic Sensors NV.
//
// For any question about terms and conditions, please contact:
// info@softkinetic.com Copyright (c) 2002-2013 Softkinetic Sensors NV

#include <stdlib.h>
#include <iostream>

#include <DepthSense.hxx>

using namespace std;
using namespace DepthSense;

static void error (const char* message)
{
    cerr << message << endl;
    exit(1);
}

static Node getFirstAvailableNode (Context context)
{
    // obtain the list of devices attached to the host
    vector<Device> devices = context.getDevices();

    for (vector<Device>::const_iterator iter = devices.begin(); iter != devices
      .end(); iter++)
    {
        Device device = *iter;

        // obtain the list of nodes of the current device
        vector<Node> nodes = device.getNodes();

        // return the first node if any
        if (! nodes.empty())
            return nodes[0];
    }

    // return an unset node
    return Node();
}

template <class T>
static bool tryDisplayProperty (Interface iface, PropertyBase prop)
{
    // attempt to downcast the PropertyBase to a Property<T>
    Property<T> derivedProp = prop.as< Property<T> >();
    if (! derivedProp.isSet())
        return false;

    // display the property name and its value
    cout << "Property " << prop.name() << " has value " << derivedProp.getValue
      (iface) << endl;

    return true;
}

static void displayProperty (Interface iface, PropertyBase prop)
{
    // handle a few common property types
    if (tryDisplayProperty<bool>(iface, prop))
        return;
    if (tryDisplayProperty<int32_t>(iface, prop))
        return;
    if (tryDisplayProperty<string>(iface, prop))
        return;

    // the type is unhandled
    cout << "Property " << prop.name() << " has unhandled type" << endl;
}

static void displayProperties (Interface iface)
{
    // retrieve the runtime type of the object
    Type type = iface.getType();

    // obtain the list of properties declared by that type
    vector<PropertyBase> properties = type.getProperties();

    // display the properties
    for (vector<PropertyBase>::const_iterator iter = properties.begin(); iter !
      = properties.end(); iter++)
    {
        PropertyBase prop = *iter;
        displayProperty(iface, prop);
    }
}

int main (int argc, char** argv)
{
    // create a connection to the DepthSense server at localhost
    Context context = Context::create();

    // get the first available sensor
    Node node = getFirstAvailableNode(context);

    // if no node was found, fail
    if (! node.isSet())
        error("no node found");

    // display the properties of the node
    displayProperties(node);
}
```

1.5 Multi-camera, multi-client support

DepthSense SDK can interface with multiple cameras simultaneously. The DepthSense::Context::getDevices() method returns the list of camera devices attached to the host system.

Moreover, a single DepthSense server instance can serve multiple clients simultaneously. To be notified when the server accepts a new client connection, connect to the `clientConnected` event of the DepthSense::Context class.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 DepthSense Namespace Reference

The DepthSense Software Development Kit.

Classes

- class ArgumentException

    The type of the exception thrown when an argument is unset or invalid.
- class AudioNode

    Represents an audio stream data source.
- class ColorNode

    Represents a color stream data source.
- class ConfigurationException

    The type of the exception thrown when a valid configuration failed to apply.
- class Context

    Represents an application session.
- class DepthNode

    Represents a depth stream data source.
- class Device

    Represents a camera device.
- class Exception

    The base exception class.
- struct Extended2DPoint

    A point in the cartesian space as defined by its floating point pixel coordinates, and its integral cartesian depth.
- struct ExtrinsicParameters

The extrinsic parameters of the camera system.

- struct FPExtended2DPoint

    A point in the cartesian space as defined by its floating point pixel coordinates, and its floating point cartesian depth.
- struct FPVertex

    A point in space as defined by its floating point coordinates.
- class InitializationException

    The type of the exception thrown when an initialization error has occurred.
- class Interface

    The base interface class.
- struct IntrinsicParameters

    The intrinsic parameters of the camera system.
- class InvalidOperationException

    The type of the exception thrown when the current state of an object does not support the requested operation.
- class IOException

    The type of the exception throw when a device or file I/O operation has failed.
- class Node

    Represents a stream data source.
- class NotSupportedException

    The type of the exception thrown when a unsupported operation is requested.
- struct Point2D

    A point in the cartesian space as defined by its floating point pixel coordinates.
- class Pointer

    Exposes a memory buffer.
- class ProcessingHelper
- class ProjectionHelper

    Computes the UV mapping of image points.
- class Property

    The strongly-typed property leaf class.
- class Property< std::string >

    The string property class.
- class PropertyBase

    The base property class.
- struct StereoCameraParameters

    The intrinsic and extrinsic parameters of the camera system.
- class StreamingException

    The type of the exception thrown when a streaming error has occurred.
- class TimeoutException

    The type of the exception thrown when a timeout condition occurs.
- class TransportException

    The type of the exception thrown when a network or protocol error has occurred.
- class Type

    Represents a DepthSense instance type.

- class UnauthorizedAccessException

    The type of the exception thrown when access to a privileged operation is denied.
- class UnsupportedNode

    Represents an unsupported stream data source.
- struct UV

    UV coordinates.
- struct Version

    DepthSense version information.
- struct Vertex

    A point in space as defined by its integer coordinates.

Enumerations

- enum CameraPlane { CAMERA_PLANE_COLOR = 0, CAMERA_PLANE_DEPTH = 1 }

    The camera plane to project on.
- enum CompressionType { COMPRESSION_TYPE_YUY2 = 0, COMPRESSION_TYPE_MJPEG = 1 }

    The image compression type.
- enum ExposureAuto { EXPOSURE_AUTO_MANUAL = 0, EXPOSURE_AUTO_APERTURE_PRIORITY = 1 }

    The auto exposure mode.
- enum FrameFormat { FRAME_FORMAT_UNKNOWN = 0, FRAME_FORMAT_QQVGA = 1, FRAME_FORMAT_QCIF = 2, FRAME_FORMAT_HQVGA = 3, FRAME_FORMAT_QVGA = 4, FRAME_FORMAT_CIF = 5, FRAME_FORMAT_HVGA = 6, FRAME_FORMAT_VGA = 7, FRAME_FORMAT_WXGA_H = 8, FRAME_FORMAT_DS311 = 9, FRAME_FORMAT_XGA = 10, FRAME_FORMAT_SVGA = 11, FRAME_FORMAT_OVVGA = 12, FRAME_FORMAT_WHVGA = 13, FRAME_FORMAT_NHD = 14 }

    The image resolution.
- enum PowerLineFrequency { POWER_LINE_FREQUENCY_DISABLED = 0, POWER_LINE_FREQUENCY_50HZ = 1, POWER_LINE_FREQUENCY_60HZ = 2 }

    The power line frequency.

Functions

- static std::string CameraPlane_toString (CameraPlane value)

    Converts a DepthSense::CameraPlane value to a string.
- static std::string CompressionType_toString (CompressionType value)

    Converts a DepthSense::CompressionType value to a string.
- static std::string ExposureAuto_toString (ExposureAuto value)

    Converts a DepthSense::ExposureAuto value to a string.
- FrameFormat FrameFormat_fromResolution (int32_t width, int32_t height)

Converts a resolution to a FrameFormat value.

- void FrameFormat_toResolution (FrameFormat frameFormat, int32_t ∗width, int32_t ∗height)

  Converts a FrameFormat value to a resolution.
- static std::string FrameFormat_toString (FrameFormat value)

  Converts a DepthSense::FrameFormat value to a string.
- DepthSense::Version getLibraryVersion ()

  Gets the DepthSense Library version information.
- static std::string PowerLineFrequency_toString (PowerLineFrequency value)

  Converts a DepthSense::PowerLineFrequency value to a string.

5.1.1 Enumeration Type Documentation

5.1.1.1 enum DepthSense::CameraPlane

The camera plane to project the 3D points on.

Enumerator:

CAMERA_PLANE_COLOR the color plane

CAMERA_PLANE_DEPTH the depth plane

5.1.1.2 enum DepthSense::CompressionType

A type enumerating the various compression types supported by DepthSense SDK.

Enumerator:

COMPRESSION_TYPE_YUY2 Y’UV422

COMPRESSION_TYPE_MJPEG MJPEG

5.1.1.3 enum DepthSense::ExposureAuto

The supported auto exposure modes.

Enumerator:

EXPOSURE_AUTO_MANUAL manual

EXPOSURE_AUTO_APERTURE_PRIORITY aperture priority

5.1.1.4 enum DepthSense::FrameFormat

A type enumerating the various frame formats supported by DepthSense SDK.

Enumerator:

FRAME_FORMAT_UNKNOWN	unknown
FRAME_FORMAT_QQVGA	QQVGA (160x120)
FRAME_FORMAT_QCIF	QCIF (176x144)
FRAME_FORMAT_HQVGA	HQVGA (240x160)
FRAME_FORMAT_QVGA	QVGA (320x240)
FRAME_FORMAT_CIF	CIF (352x288)
FRAME_FORMAT_HVGA	HVGA (480x320)
FRAME_FORMAT_VGA	VGA (640x480)
FRAME_FORMAT_WXGA_H	WXGA_H (1280x720)
FRAME_FORMAT_DS311	DS311 (320x120)
FRAME_FORMAT_XGA	XGA (1024x768)
FRAME_FORMAT_SVGA	SVGA (800x600)
FRAME_FORMAT_OVVGA	OVVGA (636x480)
FRAME_FORMAT_WHVGA	WHVGA (640x240)
FRAME_FORMAT_NHD	nHD (640x360)

5.1.1.5 enum DepthSense::PowerLineFrequency

The supported power line frequencies.

Enumerator:

POWER_LINE_FREQUENCY_DISABLED	disabled
POWER_LINE_FREQUENCY_50HZ	50 Hz
POWER_LINE_FREQUENCY_60HZ	60 Hz

5.1.2 Function Documentation

5.1.2.1 static std::string DepthSense::CameraPlane_toString (CameraPlane *value*) `[inline, static]`

Converts the provided enumeration value to a string.

Parameters

value	the enumeration value to convert

Returns

the name of the enumeration member whose value is `value`, or, if `value` is not a member of DepthSense::CameraPlane, its numeric representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

5.1.2.2 static std::string DepthSense::CompressionType_toString (CompressionType *value*) `[inline, static]`

Converts the provided enumeration value to a string.

Parameters

value	the enumeration value to convert

Returns

the name of the enumeration member whose value is `value`, or, if `value` is not a member of DepthSense::CompressionType, its numeric representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

5.1.2.3 static std::string DepthSense::ExposureAuto_toString (ExposureAuto *value*) `[inline, static]`

Converts the provided enumeration value to a string.

Parameters

value	the enumeration value to convert

Returns

the name of the enumeration member whose value is `value`, or, if `value` is not a member of DepthSense::ExposureAuto, its numeric representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

5.1.2.4 FrameFormat DepthSense::FrameFormat_fromResolution (int32_t *width*, int32_t *height*)

Converts a resolution to a FrameFormat enumeration value.

Parameters

width	the width
height	the height

Returns

the corresponding FrameFormat value

Exceptions

DepthSense::Argumer	the `width`, `height` pair does not match any FrameFormat value
std::bad_alloc	not enough memory to perform the requested operation

5.1.2.5 void DepthSense::FrameFormat_toResolution (FrameFormat *frameFormat*, int32_t ∗ *width*, int32_t ∗ *height*)

Converts a FrameFormat enumeration value to a resolution.

Parameters

frameFormat	the FrameFormat value to convert
width	a location to store the resulting width
height	a location to store the resulting height

Exceptions

DepthSense::Argumer	`frameFormat` is invalid
std::bad_alloc	not enough memory to perform the requested operation

5.1.2.6 static std::string DepthSense::FrameFormat_toString (FrameFormat *value*) [inline, static]

Converts the provided enumeration value to a string.

Parameters

value	the enumeration value to convert

Returns

the name of the enumeration member whose value is `value`, or, if `value` is not a member of DepthSense::FrameFormat, its numeric representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

5.1.2.7 DepthSense::Version DepthSense::getLibraryVersion ( )

Returns a structure containing the DepthSense Library version information.

Returns

the DepthSense Library version information

See also

Context::getClientVersion
Context::getServerVersion

5.1.2.8 static std::string DepthSense::PowerLineFrequency_toString (PowerLineFrequency *value*) [inline, static]

Converts the provided enumeration value to a string.

Parameters

value	the enumeration value to convert

Returns

the name of the enumeration member whose value is `value`, or, if `value` is not a member of DepthSense::PowerLineFrequency, its numeric representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

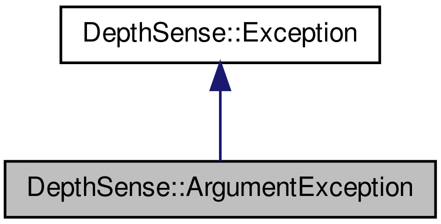
Chapter 6

Class Documentation

6.1 DepthSense::ArgumentException Class Reference

The type of the exception thrown when an argument is unset or invalid.

Inheritance diagram for DepthSense::ArgumentException:



Public Member Functions

- std::string getParameterName () const

    Returns the name of the corresponding parameter.

Protected Member Functions

- **ArgumentException** (void ∗data)

6.1.1 Detailed Description

ArgumentException is thrown when a method argument or property value is unset when it should be set or is outside of the range of allowed values.

6.1.2 Member Function Documentation

6.1.2.1 std::string DepthSense::ArgumentException::getParameterName ( ) const

Returns the name of the method parameter whose argument is invalid. For a property value, this name is always `value`.

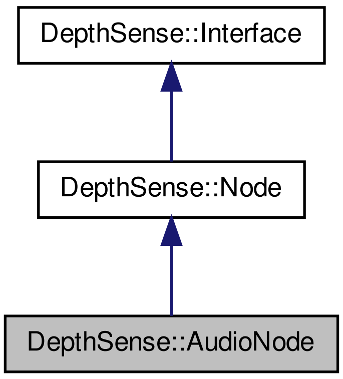
Returns

the parameter name

6.2 DepthSense::AudioNode Class Reference

Represents an audio stream data source.

Inheritance diagram for DepthSense::AudioNode:



Classes

- struct Configuration

  The configuration of an audio node.
- struct NewSampleReceivedData

  Holds the DepthSense::AudioNode::NewSampleReceivedEvent arguments.
- class NewSampleReceivedEvent

  Event raised when an audio sample is captured.

Public Member Functions

- bool configurationIsReadOnly ()

  Checks whether property AudioNode::configuration is read-only.
- DepthSense::AudioNode::Configuration getConfiguration ()

  Gets the value of the AudioNode::configuration property.
- std::vector < DepthSense::AudioNode::Configuration > getConfigurations ()

  Gets the value of the AudioNode::configurations property.
- float getInputMixerLevel ()

  Gets the value of the AudioNode::inputMixerLevel property.
- bool getMute ()

  Gets the value of the AudioNode::mute property.
- bool inputMixerLevelIsReadOnly ()

  Checks whether property AudioNode::inputMixerLevel is read-only.
- bool muteIsReadOnly ()

  Checks whether property AudioNode::mute is read-only.
- DepthSense::AudioNode::NewSampleReceivedEvent & newSampleReceivedEvent () const

  Returns the `newSampleReceived` event object.
- void setConfiguration (DepthSense::AudioNode::Configuration value)

  Sets the value of the AudioNode::configuration property.
- void setInputMixerLevel (float value)

  Sets the value of the AudioNode::inputMixerLevel property.
- void setMute (bool value)

  Sets the value of the AudioNode::mute property.

Static Public Member Functions

- static DepthSense::Type type ()

  Returns the DepthSense::AudioNode type object.

Properties

- DepthSense::AudioNode::Configuration configuration

  The node configuration.
- std::vector < DepthSense::AudioNode::Configuration > configurations

  The list of supported node configurations.
- float inputMixerLevel

  The recording level.
- bool mute

  Whether to mute the recording.

6.2.1 Detailed Description

The AudioNode class allows to capture audio data with the microphone array of a given camera device.

6.2.2 Member Function Documentation

6.2.2.1 bool DepthSense::AudioNode::configurationIsReadOnly ( )

Checks whether property AudioNode::configuration is read-only.

The AudioNode::configuration property specifies the configuration of the audio node.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

whether property AudioNode::configuration is read-only

See also

setConfiguration()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.2 DepthSense::AudioNode::Configuration DepthSense::AudioNode::getConfiguration ( )

Gets the value of the AudioNode::configuration property.

The AudioNode::configuration property specifies the configuration of the audio node.

Exceptions

DepthSense::InvalidOp	the node no longer exists

Returns

the value of the AudioNode::configuration property

See also

setConfiguration()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.3 std::vector< DepthSense::AudioNode::Configuration > DepthSense::AudioNode::getConfigurations ( ) [inline]

Gets the value of the AudioNode::configurations property.

Returns

the value of the AudioNode::configurations property

Exceptions

DepthSense::InvalidOp	the node no longer exists
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.4 float DepthSense::AudioNode::getInputMixerLevel ( )

Gets the value of the AudioNode::inputMixerLevel property.

The AudioNode::inputMixerLevel property accepts a value ranging from 0.0 to 1.0.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

the value of the AudioNode::inputMixerLevel property

See also

setInputMixerLevel()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.5 bool DepthSense::AudioNode::getMute ( )

Gets the value of the AudioNode::mute property.

The AudioNode::mute property specifies whether to mute the recording.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

the value of the AudioNode::mute property

See also

setMute()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.6 bool DepthSense::AudioNode::inputMixerLevelIsReadOnly ( )

Checks whether property AudioNode::inputMixerLevel is read-only.

The AudioNode::inputMixerLevel property accepts a value ranging from 0.0 to 1.0.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

whether property AudioNode::inputMixerLevel is read-only

See also

setInputMixerLevel()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.7 bool DepthSense::AudioNode::muteIsReadOnly ( )

Checks whether property AudioNode::mute is read-only.

The AudioNode::mute property specifies whether to mute the recording.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

whether property AudioNode::mute is read-only

See also

setMute()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.8 DepthSense::AudioNode::NewSampleReceivedEvent& DepthSense::AudioNode::newSampleReceivedEvent ( ) const

Returns a reference to the `newSampleReceived` event object, which can be used to connect handlers to that event.

Returns

the `newSampleReceived` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.2.2.9 void DepthSense::AudioNode::setConfiguration (DepthSense::AudioNode::Configuration *value*)

Sets the value of the AudioNode::configuration property.

The AudioNode::configuration property specifies the configuration of the audio node.

Exceptions

DepthSense::InvalidO	the node no longer exists

Parameters

value	the value to set

See also

getConfiguration(), configurationIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided configuration is invalid
DepthSense::Configur	the provided configuration is valid but failed to apply
DepthSense::Streamir	streaming was enabled at the time of the call and could not be restarted because of a device or software error
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.10 void DepthSense::AudioNode::setInputMixerLevel (float *value*)

Sets the value of the AudioNode::inputMixerLevel property.

The AudioNode::inputMixerLevel property accepts a value ranging from 0.0 to 1.0.

Exceptions

DepthSense::InvalidOp	the node no longer exists

Parameters

value	the value to set

See also

getInputMixerLevel(), inputMixerLevelIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.11 void DepthSense::AudioNode::setMute (bool *value*)

Sets the value of the AudioNode::mute property.

The AudioNode::mute property specifies whether to mute the recording.

Exceptions

DepthSense::InvalidOp	the node no longer exists

Parameters

value	the value to set

See also

getMute(), muteIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.2.2.12 static DepthSense::Type DepthSense::AudioNode::type ( ) [static]

Returns the DepthSense::AudioNode type object

Returns

the DepthSense::AudioNode type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented from DepthSense::Node.

6.2.3 Property Documentation

6.2.3.1 DepthSense::AudioNode::Configuration DepthSense-::AudioNode::configuration `[read, write, assign]`

The AudioNode::configuration property specifies the configuration of the audio node.

Exceptions

DepthSense::InvalidO	the node no longer exists

6.2.3.2 std::vector< DepthSense::AudioNode::Configuration > DepthSense::AudioNode::configurations `[read, assign]`

The AudioNode::configurations property specifies the list of supported node configurations.

6.2.3.3 float DepthSense::AudioNode::inputMixerLevel `[read, write, assign]`

The AudioNode::inputMixerLevel property accepts a value ranging from 0.0 to 1.0.

Exceptions

DepthSense::InvalidO	the node no longer exists

6.2.3.4 bool DepthSense::AudioNode::mute `[read, write, assign]`

The AudioNode::mute property specifies whether to mute the recording.

Exceptions

DepthSense::InvalidO	the node no longer exists

6.3 DepthSense::AudioNode::Configuration Struct Reference

The configuration of an audio node.

Public Member Functions

- Configuration (int32_t channels, int32_t bitsPerSample, int32_t sampleRate)

  Constructs a Configuration instance.
- bool operator!= (const Configuration &other) const

  Compares two Configuration instances for inequality.
- bool operator== (const Configuration &other) const

  Compares two Configuration instances for equality.

Public Attributes

- int32_t bitsPerSample

  the number of bits per sample
- int32_t channels

  the number of audio channels
- int32_t sampleRate

  the sample rate in Hz

6.3.1 Detailed Description

The Configuration struct holds the configuration of an audio node.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 DepthSense::AudioNode::Configuration::Configuration (int32_t *channels*, int32_t *bitsPerSample*, int32_t *sampleRate*)

Constructs a Configuration instance, initializing the instance fields with the provided values.

Parameters

channels	the value of the Configuration::channels field
bitsPer-Sample	the value of the Configuration::bitsPerSample field
sampleRate	the value of the Configuration::sampleRate field

6.3.3 Member Function Documentation

6.3.3.1 bool DepthSense::AudioNode::Configuration::operator!= (const Configuration & *other*) const

Checks whether the current Configuration instance is different from the Configuration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.3.3.2 bool DepthSense::AudioNode::Configuration::operator== (const Configuration & *other*) const

Checks whether the current Configuration instance is equal to the Configuration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.4 DepthSense::AudioNode::NewSampleReceivedData Struct Reference

Holds the DepthSense::AudioNode::NewSampleReceivedEvent arguments.

Public Attributes

- ::DepthSense::Pointer< uint8_t > audioData

  the audio data

- DepthSense::AudioNode::Configuration captureConfiguration

  the camera configuration that was in effect at the time of capture
- int32_t cumulativeDroppedSampleCount

  the number of dropped samples since the streaming was started
- int32_t droppedSampleCount

  the number of dropped samples since the last `newSampleReceived` *event was raised*
- uint64_t timeOfArrival

  the time of arrival of the sample in the library, expressed in µs
- uint64_t timeOfCapture

  the time of capture of the sample, expressed in µs

6.4.1 Detailed Description

The NewSampleReceivedData struct holds the DepthSense::AudioNode::NewSampleReceivedEvent parameters and is passed to callbacks connected to that event.

6.5 DepthSense::AudioNode::NewSampleReceivedEvent Class Reference

Event raised when an audio sample is captured.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleRec
  data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration capture-Configuration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t dropped-SampleCount, int32_t cumulativeDroppedSampleCount))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceived
  data, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration capture-Configuration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t dropped-SampleCount, int32_t cumulativeDroppedSampleCount, T closureData), T closureData)

  Connects a closure to the current event.

- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedDa
  data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::AudioNode obj,-::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData
  data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration capture-Configuration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t dropped-SampleCount, int32_t cumulativeDroppedSampleCount))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration capture-Configuration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t dropped-SampleCount, int32_t cumulativeDroppedSampleCount, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::AudioNode obj,-::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Disconnects a method from the current event.

6.5.1 Detailed Description

The `newSampleReceived` event is raised when an audio sample is captured.

Parameters

audioData	the audio data
capture-Configuration	the camera configuration that was in effect at the time of capture
timeOf-Capture	the time of capture of the sample, expressed in µs
timeOf-Arrival	the time of arrival of the sample in the library, expressed in µs
dropped-Sample-Count	the number of dropped samples since the last `newSample-Received` event was raised
cumulative-Dropped-Sample-Count	the number of dropped samples since the streaming was started

6.5.2 Member Function Documentation

6.5.2.1 void DepthSense::AudioNode::NewSampleReceivedEvent::connect (void(∗)(DepthSense::AudioNode obj, DepthSense::-AudioNode::NewSampleReceivedData data) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.2 void DepthSense::AudioNode::NewSampleReceivedEvent::connect (void(∗)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`audioData`	the audio data
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.3 template<class T > void DepthSense::AudioNode::NewSample-ReceivedEvent::connect (void(∗)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData data, T closureData) *closure*, T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.4 template< class T > void DepthSense::AudioNode::NewSampleReceivedEvent::connect (void(∗)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`audioData`	the audio data
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.5 template<class T > void DepthSense::AudioNode::NewSampleReceivedEvent::connect (T * *obj*, void(T::*)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData data) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.6 template<class T > void DepthSense::AudioNode::NewSampleReceivedEvent::connect (T * *obj,* void(T::*)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`audioData`	the audio data
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.7 void DepthSense::AudioNode::NewSampleReceivedEvent::disconnect (void(*)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData data) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.8 void DepthSense::AudioNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`audioData`	the audio data
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.9 template<class T > void DepthSense::AudioNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData data, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.10 template<class T > void DepthSense::AudioNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`audioData`	the audio data
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.11 template<class T > void DepthSense::AudioNode::NewSampleReceivedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::AudioNode obj, DepthSense::AudioNode::NewSampleReceivedData data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.5.2.12 template<class T > void DepthSense::AudioNode::NewSampleReceivedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::AudioNode obj,::DepthSense::Pointer< uint8_t > audioData, DepthSense::AudioNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`audioData`	the audio data
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

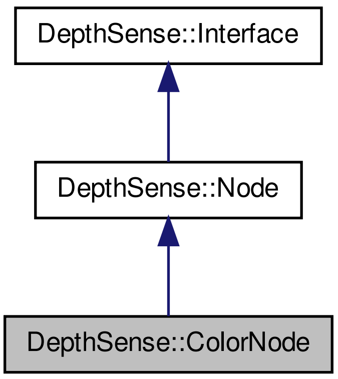
Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.6 DepthSense::ColorNode Class Reference

Represents a color stream data source.

Inheritance diagram for DepthSense::ColorNode:



Classes

- struct Configuration

  The configuration of a color node.
- struct NewSampleReceivedData

  Holds the DepthSense::ColorNode::NewSampleReceivedEvent arguments.
- class NewSampleReceivedEvent

  Event raised when a color sample is captured.

Public Member Functions

- bool brightnessIsReadOnly ()

Checks whether property ColorNode::brightness is read-only.
- bool configurationIsReadOnly ()

  Checks whether property ColorNode::configuration is read-only.
- bool contrastIsReadOnly ()

  Checks whether property ColorNode::contrast is read-only.
- bool enableColorMapIsReadOnly ()

  Checks whether property ColorNode::enableColorMap is read-only.
- bool enableCompressedDataIsReadOnly ()

  Checks whether property ColorNode::enableCompressedData is read-only.
- bool exposureAutoIsReadOnly ()

  Checks whether property ColorNode::exposureAuto is read-only.
- bool exposureAutoPriorityIsReadOnly ()

  Checks whether property ColorNode::exposureAutoPriority is read-only.
- bool exposureIsReadOnly ()

  Checks whether property ColorNode::exposure is read-only.
- bool gammaIsReadOnly ()

  Checks whether property ColorNode::gamma is read-only.
- int32_t getBrightness ()

  Gets the value of the ColorNode::brightness property.
- DepthSense::ColorNode::Configuration getConfiguration ()

  Gets the value of the ColorNode::configuration property.
- std::vector < DepthSense::ColorNode::Configuration > getConfigurations ()

  Gets the value of the ColorNode::configurations property.
- int32_t getContrast ()

  Gets the value of the ColorNode::contrast property.
- bool getEnableColorMap ()

  Gets the value of the ColorNode::enableColorMap property.
- bool getEnableCompressedData ()

  Gets the value of the ColorNode::enableCompressedData property.
- int32_t getExposure ()

  Gets the value of the ColorNode::exposure property.
- DepthSense::ExposureAuto getExposureAuto ()

  Gets the value of the ColorNode::exposureAuto property.
- bool getExposureAutoPriority ()

  Gets the value of the ColorNode::exposureAutoPriority property.
- int32_t getGamma ()

  Gets the value of the ColorNode::gamma property.
- int32_t getHue ()

  Gets the value of the ColorNode::hue property.
- int32_t getSaturation ()

  Gets the value of the ColorNode::saturation property.
- int32_t getSharpness ()

  Gets the value of the ColorNode::sharpness property.

- int32_t getWhiteBalance ()

  Gets the value of the ColorNode::whiteBalance property.
- bool getWhiteBalanceAuto ()

  Gets the value of the ColorNode::whiteBalanceAuto property.
- bool hueIsReadOnly ()

  Checks whether property ColorNode::hue is read-only.
- DepthSense::ColorNode::NewSampleReceivedEvent & newSampleReceivedEvent () const

  Returns the `newSampleReceived` event object.
- bool saturationIsReadOnly ()

  Checks whether property ColorNode::saturation is read-only.
- void setBrightness (int32_t value)

  Sets the value of the ColorNode::brightness property.
- void setConfiguration (DepthSense::ColorNode::Configuration value)

  Sets the value of the ColorNode::configuration property.
- void setContrast (int32_t value)

  Sets the value of the ColorNode::contrast property.
- void setEnableColorMap (bool value)

  Sets the value of the ColorNode::enableColorMap property.
- void setEnableCompressedData (bool value)

  Sets the value of the ColorNode::enableCompressedData property.
- void setExposure (int32_t value)

  Sets the value of the ColorNode::exposure property.
- void setExposureAuto (DepthSense::ExposureAuto value)

  Sets the value of the ColorNode::exposureAuto property.
- void setExposureAutoPriority (bool value)

  Sets the value of the ColorNode::exposureAutoPriority property.
- void setGamma (int32_t value)

  Sets the value of the ColorNode::gamma property.
- void setHue (int32_t value)

  Sets the value of the ColorNode::hue property.
- void setSaturation (int32_t value)

  Sets the value of the ColorNode::saturation property.
- void setSharpness (int32_t value)

  Sets the value of the ColorNode::sharpness property.
- void setWhiteBalance (int32_t value)

  Sets the value of the ColorNode::whiteBalance property.
- void setWhiteBalanceAuto (bool value)

  Sets the value of the ColorNode::whiteBalanceAuto property.
- bool sharpnessIsReadOnly ()

  Checks whether property ColorNode::sharpness is read-only.
- bool whiteBalanceAutoIsReadOnly ()

  Checks whether property ColorNode::whiteBalanceAuto is read-only.
- bool whiteBalanceIsReadOnly ()

  Checks whether property ColorNode::whiteBalance is read-only.

Static Public Member Functions

- static DepthSense::Type type ()

  Returns the DepthSense::ColorNode type object.

Properties

- int32_t brightness

  The brightness.
- DepthSense::ColorNode::Configuration configuration

  The node configuration.
- std::vector < DepthSense::ColorNode::Configuration > configurations

  The list of supported node configurations.
- int32_t contrast

  The contrast.
- bool enableColorMap

  Whether to enable the color map.
- bool enableCompressedData

  Whether to enable the compressed data.
- int32_t exposure

  The exposure.
- DepthSense::ExposureAuto exposureAuto

  The auto exposure mode.
- bool exposureAutoPriority

  Whether to enable the auto exposure priority mode.
- int32_t gamma

  The gamma.
- int32_t hue

  The hue.
- int32_t saturation

  The saturation.
- int32_t sharpness

  The sharpness.
- int32_t whiteBalance

  The white balance.
- bool whiteBalanceAuto

  Whether to enable automatic white balance.

6.6.1 Detailed Description

The ColorNode class allows to capture pixel data with the RGB sensor of a given camera device.

6.6.2 Member Function Documentation

6.6.2.1 bool DepthSense::ColorNode::brightnessIsReadOnly ( )

Checks whether property ColorNode::brightness is read-only.

The ColorNode::brightness property accepts a value ranging from -10 to 10 with a step of 1. This is a relative value where increasing values indicate increasing brightness.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property ColorNode::brightness is read-only

See also

setBrightness()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.2 bool DepthSense::ColorNode::configurationIsReadOnly ( )

Checks whether property ColorNode::configuration is read-only.

The ColorNode::configuration property specifies the configuration of the color node.

Returns

whether property ColorNode::configuration is read-only

See also

setConfiguration()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.3 bool DepthSense::ColorNode::contrastIsReadOnly ( )

Checks whether property ColorNode::contrast is read-only.

The ColorNode::contrast property accepts a value ranging from 1 to 32 with a step of 1. This is a relative value where increasing values indicate increasing contrast.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property ColorNode::contrast is read-only

See also

setContrast()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.4 bool DepthSense::ColorNode::enableColorMapIsReadOnly ( )

Checks whether property ColorNode::enableColorMap is read-only.

The ColorNode::enableColorMap property specifies whether to capture the color stream and make it available through the `colorMap` argument of the `newSample-Received` event.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

whether property ColorNode::enableColorMap is read-only

See also

setEnableColorMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.5 bool DepthSense::ColorNode::enableCompressedDataIsReadOnly ( )

Checks whether property ColorNode::enableCompressedData is read-only.

The ColorNode::enableCompressedData property specifies whether to capture the compressed data and make it available through the `compressedData` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

whether property ColorNode::enableCompressedData is read-only

See also

setEnableCompressedData()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.6 bool DepthSense::ColorNode::exposureAutoIsReadOnly ( )

Checks whether property ColorNode::exposureAuto is read-only.

The ColorNode::exposureAuto property specifies the auto exposure mode (either manual or aperture priority).

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property ColorNode::exposureAuto is read-only

See also

setExposureAuto()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.7 bool DepthSense::ColorNode::exposureAutoPriorityIsReadOnly ( )

Checks whether property ColorNode::exposureAutoPriority is read-only.

The ColorNode::exposureAutoPriority property specifies whether to enable the auto exposure priority mode. If set to true, the frame rate can be dynamically modified by the device. Otherwise, the frame rate must remain constant.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property ColorNode::exposureAutoPriority is read-only

See also

setExposureAutoPriority()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.8 bool DepthSense::ColorNode::exposureIsReadOnly ( )

Checks whether property ColorNode::exposure is read-only.

The ColorNode::exposure property accepts a value ranging from 156 to 5000 with a step of 1 and is expressed in 100µs units

On Windows, the following mapping is applied:

156 - 233	1/64 seconds
234 - 467	1/32 seconds
468 - 936	1/16 seconds
937 - 1874	1/8 seconds
1875 - 3749	1/4 seconds
3750 - 5000	1/2 seconds

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property ColorNode::exposure is read-only

See also

setExposure()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.9 bool DepthSense::ColorNode::gammaIsReadOnly ( )

Checks whether property ColorNode::gamma is read-only.

The ColorNode::gamma property accepts a value ranging from 100 to 200 with a step of 1. The value is expressed in gamma multiplied by 100.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property ColorNode::gamma is read-only

See also

setGamma()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.10 int32_t DepthSense::ColorNode::getBrightness ( )

Gets the value of the ColorNode::brightness property.

The ColorNode::brightness property accepts a value ranging from -10 to 10 with a step of 1. This is a relative value where increasing values indicate increasing brightness.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the ColorNode::brightness property

See also

setBrightness()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.11 DepthSense::ColorNode::Configuration DepthSense::ColorNode::getConfiguration ( )

Gets the value of the ColorNode::configuration property.

The ColorNode::configuration property specifies the configuration of the color node.

Returns

the value of the ColorNode::configuration property

See also

setConfiguration()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.12 std::vector< DepthSense::ColorNode::Configuration > DepthSense::ColorNode::getConfigurations ( ) [inline]

Gets the value of the ColorNode::configurations property.

Returns

the value of the ColorNode::configurations property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.13 int32_t DepthSense::ColorNode::getContrast ( )

Gets the value of the ColorNode::contrast property.

The ColorNode::contrast property accepts a value ranging from 1 to 32 with a step of 1.

This is a relative value where increasing values indicate increasing contrast.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the ColorNode::contrast property

See also

setContrast()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.14 bool DepthSense::ColorNode::getEnableColorMap( )

Gets the value of the ColorNode::enableColorMap property.

The ColorNode::enableColorMap property specifies whether to capture the color stream and make it available through the `colorMap` argument of the `newSample-Received` event.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

the value of the ColorNode::enableColorMap property

See also

setEnableColorMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.15 bool DepthSense::ColorNode::getEnableCompressedData ( )

Gets the value of the ColorNode::enableCompressedData property.

The ColorNode::enableCompressedData property specifies whether to capture the compressed data and make it available through the `compressedData` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the node no longer exists

Returns

the value of the ColorNode::enableCompressedData property

See also

setEnableCompressedData()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.16 int32_t DepthSense::ColorNode::getExposure ( )

Gets the value of the ColorNode::exposure property.

The ColorNode::exposure property accepts a value ranging from 156 to 5000 with a step of 1 and is expressed in 100µs units

On Windows, the following mapping is applied:

156 - 233	1/64 seconds
234 - 467	1/32 seconds
468 - 936	1/16 seconds
937 - 1874	1/8 seconds
1875 - 3749	1/4 seconds
3750 - 5000	1/2 seconds

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the ColorNode::exposure property

See also

setExposure()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.17 DepthSense::ExposureAuto DepthSense::ColorNode::getExposureAuto ( )

Gets the value of the ColorNode::exposureAuto property.

The ColorNode::exposureAuto property specifies the auto exposure mode (either manual or aperture priority).

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the ColorNode::exposureAuto property

See also

setExposureAuto()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.18 bool DepthSense::ColorNode::getExposureAutoPriority ( )

Gets the value of the ColorNode::exposureAutoPriority property.

The ColorNode::exposureAutoPriority property specifies whether to enable the auto exposure priority mode. If set to true, the frame rate can be dynamically modified by the device. Otherwise, the frame rate must remain constant.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the ColorNode::exposureAutoPriority property

See also

setExposureAutoPriority()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.19 int32_t DepthSense::ColorNode::getGamma ( )

Gets the value of the ColorNode::gamma property.

The ColorNode::gamma property accepts a value ranging from 100 to 200 with a step of 1. The value is expressed in gamma multiplied by 100.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the ColorNode::gamma property

See also

setGamma()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.20 int32_t DepthSense::ColorNode::getHue ( )

Gets the value of the ColorNode::hue property.

The ColorNode::hue property accepts a value ranging from -5 to 5 with a step of 1. The value is expressed in degrees multiplied by 100.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the ColorNode::hue property

See also

setHue()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.21 int32_t DepthSense::ColorNode::getSaturation ( )

Gets the value of the ColorNode::saturation property.

The ColorNode::saturation property accepts a value ranging from 0 to 20 with a step of 1. This is a relative value where increasing values indicate increasing saturation.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the ColorNode::saturation property

See also

setSaturation()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.22 int32_t DepthSense::ColorNode::getSharpness ( )

Gets the value of the ColorNode::sharpness property.

The ColorNode::sharpness property accepts a value ranging from 0 to 10 with a step of 1.This is a relative value where increasing values indicate increasing sharpness.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the ColorNode::sharpness property

See also

setSharpness()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.23 int32_t DepthSense::ColorNode::getWhiteBalance ( )

Gets the value of the ColorNode::whiteBalance property.

The ColorNode::whiteBalance property accepts a value ranging from 2800 (incandescent) to 6500 (daylight) with a step of 1850. The value is expressed as a color temperature in Kelvin.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the ColorNode::whiteBalance property

See also

setWhiteBalance()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.24 bool DepthSense::ColorNode::getWhiteBalanceAuto ( )

Gets the value of the ColorNode::whiteBalanceAuto property.

The ColorNode::whiteBalanceAuto property specifies whether to enable automatic white balance.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the ColorNode::whiteBalanceAuto property

See also

setWhiteBalanceAuto()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.25 bool DepthSense::ColorNode::hueIsReadOnly ( )

Checks whether property ColorNode::hue is read-only.

The ColorNode::hue property accepts a value ranging from -5 to 5 with a step of 1. The value is expressed in degrees multiplied by 100.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property ColorNode::hue is read-only

See also

setHue()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.26 DepthSense::ColorNode::NewSampleReceivedEvent& DepthSense::ColorNode::newSampleReceivedEvent ( ) const

Returns a reference to the `newSampleReceived` event object, which can be used to connect handlers to that event.

Returns

the `newSampleReceived` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.6.2.27 bool DepthSense::ColorNode::saturationIsReadOnly ( )

Checks whether property ColorNode::saturation is read-only.

The ColorNode::saturation property accepts a value ranging from 0 to 20 with a step of 1. This is a relative value where increasing values indicate increasing saturation.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property ColorNode::saturation is read-only

See also

setSaturation()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.28 void DepthSense::ColorNode::setBrightness (int32_t *value*)

Sets the value of the ColorNode::brightness property.

The ColorNode::brightness property accepts a value ranging from -10 to 10 with a step of 1. This is a relative value where increasing values indicate increasing brightness.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getBrightness(), brightnessIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.29 void DepthSense::ColorNode::setConfiguration (DepthSense::ColorNode::Configuration *value*)

Sets the value of the ColorNode::configuration property.

The ColorNode::configuration property specifies the configuration of the color node.

Parameters

value	the value to set

See also

getConfiguration(), configurationIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided configuration is invalid
DepthSense::Configur	the provided configuration is valid but failed to apply
DepthSense::Streamir	streaming was enabled at the time of the call and could not be restarted because of a device or software error
DepthSense::InvalidO	when video synchronization is enabled, the configurations of the depth and color nodes are incompatible or the node no longer exists

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.30 void DepthSense::ColorNode::setContrast (int32_t *value*)

Sets the value of the ColorNode::contrast property.

The ColorNode::contrast property accepts a value ranging from 1 to 32 with a step of 1. This is a relative value where increasing values indicate increasing contrast.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getContrast(), contrastIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.31 void DepthSense::ColorNode::setEnableColorMap (bool *value*)

Sets the value of the ColorNode::enableColorMap property.

The ColorNode::enableColorMap property specifies whether to capture the color stream and make it available through the `colorMap` argument of the `newSample-Received` event.

Exceptions

DepthSense::InvalidOp	the node no longer exists

Parameters

value	the value to set

See also

getEnableColorMap(), enableColorMapIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.32 void DepthSense::ColorNode::setEnableCompressedData(bool *value*)

Sets the value of the ColorNode::enableCompressedData property.

The ColorNode::enableCompressedData property specifies whether to capture the compressed data and make it available through the `compressedData` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the node no longer exists

Parameters

value	the value to set

See also

getEnableCompressedData(), enableCompressedDataIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.33 void DepthSense::ColorNode::setExposure (int32_t *value*)

Sets the value of the ColorNode::exposure property.

The ColorNode::exposure property accepts a value ranging from 156 to 5000 with a step of 1 and is expressed in 100µs units

On Windows, the following mapping is applied:

156 - 233	1/64 seconds
234 - 467	1/32 seconds
468 - 936	1/16 seconds
937 - 1874	1/8 seconds
1875 - 3749	1/4 seconds
3750 - 5000	1/2 seconds

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getExposure(), exposureIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.34 void DepthSense::ColorNode::setExposureAuto (DepthSense::ExposureAuto *value*)

Sets the value of the ColorNode::exposureAuto property.

The ColorNode::exposureAuto property specifies the auto exposure mode (either manual or aperture priority).

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getExposureAuto(), exposureAutoIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumen	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.35 void DepthSense::ColorNode::setExposureAutoPriority (bool *value*)

Sets the value of the ColorNode::exposureAutoPriority property.

The ColorNode::exposureAutoPriority property specifies whether to enable the auto exposure priority mode. If set to true, the frame rate can be dynamically modified by the device. Otherwise, the frame rate must remain constant.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getExposureAutoPriority(), exposureAutoPriorityIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.36 void DepthSense::ColorNode::setGamma (int32_t *value*)

Sets the value of the ColorNode::gamma property.

The ColorNode::gamma property accepts a value ranging from 100 to 200 with a step of 1. The value is expressed in gamma multiplied by 100.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getGamma(), gammaIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.37 void DepthSense::ColorNode::setHue (int32_t *value*)

Sets the value of the ColorNode::hue property.

The ColorNode::hue property accepts a value ranging from -5 to 5 with a step of 1. The value is expressed in degrees multiplied by 100.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getHue(), hueIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumen	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.38 void DepthSense::ColorNode::setSaturation (int32_t *value*)

Sets the value of the ColorNode::saturation property.

The ColorNode::saturation property accepts a value ranging from 0 to 20 with a step of 1. This is a relative value where increasing values indicate increasing saturation.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getSaturation(), saturationIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumen	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.39 void DepthSense::ColorNode::setSharpness (int32_t *value*)

Sets the value of the ColorNode::sharpness property.

The ColorNode::sharpness property accepts a value ranging from 0 to 10 with a step of 1.This is a relative value where increasing values indicate increasing sharpness.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getSharpness(), sharpnessIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumen	the provided value is outside of the range of allowed values

DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.40 void DepthSense::ColorNode::setWhiteBalance (int32_t *value*)

Sets the value of the ColorNode::whiteBalance property.

The ColorNode::whiteBalance property accepts a value ranging from 2800 (incandescent) to 6500 (daylight) with a step of 1850. The value is expressed as a color temperature in Kelvin.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getWhiteBalance(), whiteBalanceIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided value is outside of the range of allowed values
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.41 void DepthSense::ColorNode::setWhiteBalanceAuto (bool *value*)

Sets the value of the ColorNode::whiteBalanceAuto property.

The ColorNode::whiteBalanceAuto property specifies whether to enable automatic white balance.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getWhiteBalanceAuto(), whiteBalanceAutoIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Configur	a valid configuration failed to apply
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.42 bool DepthSense::ColorNode::sharpnessIsReadOnly ( )

Checks whether property ColorNode::sharpness is read-only.

The ColorNode::sharpness property accepts a value ranging from 0 to 10 with a step of 1.This is a relative value where increasing values indicate increasing sharpness.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property ColorNode::sharpness is read-only

See also

setSharpness()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.43 static DepthSense::Type DepthSense::ColorNode::type( ) [static]

Returns the DepthSense::ColorNode type object

Returns

the DepthSense::ColorNode type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented from DepthSense::Node.

6.6.2.44 bool DepthSense::ColorNode::whiteBalanceAutoIsReadOnly( )

Checks whether property ColorNode::whiteBalanceAuto is read-only.

The ColorNode::whiteBalanceAuto property specifies whether to enable automatic white balance.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property ColorNode::whiteBalanceAuto is read-only

See also

setWhiteBalanceAuto()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.2.45 bool DepthSense::ColorNode::whiteBalanceIsReadOnly ( )

Checks whether property ColorNode::whiteBalance is read-only.

The ColorNode::whiteBalance property accepts a value ranging from 2800 (incandescent) to 6500 (daylight) with a step of 1850. The value is expressed as a color temperature in Kelvin.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property ColorNode::whiteBalance is read-only

See also

setWhiteBalance()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.6.3 Property Documentation

6.6.3.1 int32_t DepthSense::ColorNode::brightness `[read, write, assign]`

The ColorNode::brightness property accepts a value ranging from -10 to 10 with a step of 1. This is a relative value where increasing values indicate increasing brightness.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.6.3.2 DepthSense::ColorNode::Configuration DepthSense::ColorNode::configuration `[read, write, assign]`

The ColorNode::configuration property specifies the configuration of the color node.

6.6.3.3 std::vector< DepthSense::ColorNode::Configuration > DepthSense::ColorNode::configurations `[read, assign]`

The ColorNode::configurations property specifies the list of supported node configurations.

Exceptions

DepthSense::InvalidO	the node no longer exists

6.6.3.4 int32_t DepthSense::ColorNode::contrast `[read, write, assign]`

The ColorNode::contrast property accepts a value ranging from 1 to 32 with a step of 1. This is a relative value where increasing values indicate increasing contrast.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.6.3.5 bool DepthSense::ColorNode::enableColorMap `[read, write, assign]`

The ColorNode::enableColorMap property specifies whether to capture the color stream and make it available through the `colorMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the node no longer exists

6.6.3.6 bool DepthSense::ColorNode::enableCompressedData `[read, write, assign]`

The ColorNode::enableCompressedData property specifies whether to capture the compressed data and make it available through the `compressedData` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the node no longer exists

6.6.3.7 int32_t DepthSense::ColorNode::exposure `[read, write, assign]`

The ColorNode::exposure property accepts a value ranging from 156 to 5000 with a step of 1 and is expressed in 100µs units

On Windows, the following mapping is applied:

156 - 233	1/64 seconds
234 - 467	1/32 seconds
468 - 936	1/16 seconds
937 - 1874	1/8 seconds
1875 - 3749	1/4 seconds
3750 - 5000	1/2 seconds

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.6.3.8 DepthSense::ExposureAuto DepthSense::ColorNode::exposureAuto `[read, write, assign]`

The ColorNode::exposureAuto property specifies the auto exposure mode (either manual or aperture priority).

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.6.3.9 bool DepthSense::ColorNode::exposureAutoPriority `[read, write, assign]`

The ColorNode::exposureAutoPriority property specifies whether to enable the auto exposure priority mode. If set to true, the frame rate can be dynamically modified by the device. Otherwise, the frame rate must remain constant.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.6.3.10 int32_t DepthSense::ColorNode::gamma `[read, write, assign]`

The ColorNode::gamma property accepts a value ranging from 100 to 200 with a step of 1. The value is expressed in gamma multiplied by 100.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.6.3.11 int32_t DepthSense::ColorNode::hue `[read, write, assign]`

The ColorNode::hue property accepts a value ranging from -5 to 5 with a step of 1. The value is expressed in degrees multiplied by 100.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.6.3.12 int32_t DepthSense::ColorNode::saturation `[read, write, assign]`

The ColorNode::saturation property accepts a value ranging from 0 to 20 with a step of 1. This is a relative value where increasing values indicate increasing saturation.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.6.3.13 int32_t DepthSense::ColorNode::sharpness `[read, write, assign]`

The ColorNode::sharpness property accepts a value ranging from 0 to 10 with a step of 1.This is a relative value where increasing values indicate increasing sharpness.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.6.3.14 int32_t DepthSense::ColorNode::whiteBalance `[read, write, assign]`

The ColorNode::whiteBalance property accepts a value ranging from 2800 (incandescent) to 6500 (daylight) with a step of 1850. The value is expressed as a color temperature in Kelvin.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.6.3.15 bool DepthSense::ColorNode::whiteBalanceAuto `[read, write, assign]`

The ColorNode::whiteBalanceAuto property specifies whether to enable automatic white balance.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.7 DepthSense::ColorNode::Configuration Struct Reference

The configuration of a color node.

Public Member Functions

- Configuration (DepthSense::FrameFormat frameFormat, int32_t framerate, DepthSense::PowerLineFrequency powerLineFrequency, DepthSense::CompressionType compression)

  Constructs a Configuration instance.
- bool operator!= (const Configuration &other) const

  Compares two Configuration instances for inequality.
- bool operator== (const Configuration &other) const

  Compares two Configuration instances for equality.

Public Attributes

- DepthSense::CompressionType compression

  the compression type
- DepthSense::FrameFormat frameFormat

  the frame format and resolution
- int32_t framerate

  the frame rate in frames per second
- DepthSense::PowerLineFrequency powerLineFrequency

  the power line frequency in Hz

6.7.1 Detailed Description

The Configuration struct holds the configuration of a color node.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 DepthSense::ColorNode::Configuration::Configuration (DepthSense::FrameFormat *frameFormat,* int32_t *framerate,* DepthSense::PowerLineFrequency *powerLineFrequency,* DepthSense::CompressionType *compression*)

Constructs a Configuration instance, initializing the instance fields with the provided values.

Parameters

frameFormat	the value of the Configuration::frameFormat field
framerate	the value of the Configuration::framerate field
powerLine-Frequency	the value of the Configuration::powerLineFrequency field
compression	the value of the Configuration::compression field

6.7.3 Member Function Documentation

6.7.3.1 bool DepthSense::ColorNode::Configuration::operator!= (const Configuration & *other*) const

Checks whether the current Configuration instance is different from the Configuration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.7.3.2 bool DepthSense::ColorNode::Configuration::operator== (const Configuration & *other*) const

Checks whether the current Configuration instance is equal to the Configuration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.8 DepthSense::ColorNode::NewSampleReceivedData Struct - Reference

Holds the DepthSense::ColorNode::NewSampleReceivedEvent arguments.

Public Attributes

- DepthSense::ColorNode::Configuration captureConfiguration

  the camera configuration that was in effect at the time of capture

- ::DepthSense::Pointer< uint8_t > colorMap

  The color map. If `captureConfiguration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is BGR, otherwise the output format is YUY2.

- ::DepthSense::Pointer< uint8_t > compressedData

  The compressed data. If `captureConfiguration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.

- int32_t cumulativeDroppedSampleCount

  the number of dropped samples since the streaming was started

- int32_t droppedSampleCount

  the number of dropped samples since the last `newSampleReceived` event was raised

- uint64_t timeOfArrival

  the time of arrival of the sample in the library, expressed in µs

- uint64_t timeOfCapture

  the time of capture of the sample, expressed in µs

6.8.1 Detailed Description

The NewSampleReceivedData struct holds the DepthSense::ColorNode::NewSampleReceivedEvent parameters and is passed to callbacks connected to that event.

6.9 DepthSense::ColorNode::NewSampleReceivedEvent Class - Reference

Event raised when a color sample is captured.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data))

  Connects a function to the current event.

- void connect (void(∗handlerFunc)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Connects a function to the current event.

- template<class T >
  void connect (void(∗closure)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data, T closureData), T closureData)

  Connects a closure to the current event.

- template<class T >
  void connect (void(∗closure)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSample data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleR data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceiv data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::ColorNode obj,-::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

Disconnects a method from the current event.

6.9.1 Detailed Description

The `newSampleReceived` event is raised when a color sample is captured.

Parameters

colorMap	The color map. If `captureConfiguration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is B-GR, otherwise the output format is YUY2.
compressed-Data	The compressed data. If `captureConfiguration-::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.
capture-Configuration	the camera configuration that was in effect at the time of capture
timeOf-Capture	the time of capture of the sample, expressed in µs
timeOf-Arrival	the time of arrival of the sample in the library, expressed in µs
dropped-Sample-Count	the number of dropped samples since the last `newSample-Received` event was raised
cumulative-Dropped-Sample-Count	the number of dropped samples since the streaming was started

6.9.2 Member Function Documentation

6.9.2.1 void DepthSense::ColorNode::NewSampleReceivedEvent::connect (void(∗)(DepthSense::ColorNode obj, DepthSense::-ColorNode::NewSampleReceivedData data) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.2 void DepthSense::ColorNode::NewSampleReceivedEvent::connect (void(∗)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`colorMap`	The color map. If `capture-Configuration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is BGR, otherwise the output format is YUY2.
`compressedData`	The compressed data. If `capture-Configuration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.3 template<class T > void DepthSense::ColorNode::NewSampleReceivedEvent::connect (void(*)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.4 template<class T > void DepthSense::ColorNode::NewSampleReceivedEvent::connect (void(*)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`colorMap`	The color map. If `captureConfiguration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is BGR, otherwise the output format is YUY2.
`compressedData`	The compressed data. If `captureConfiguration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSampleCount`	the number of dropped samples since the streaming was started
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.5 template<class T > void DepthSense::ColorNode::NewSampleReceivedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.6 template<class T > void DepthSense::ColorNode::NewSampleReceivedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`colorMap`	The color map. If `captureConfiguration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is BGR, otherwise the output format is YUY2.
`compressedData`	The compressed data. If `captureConfiguration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSampleCount`	the number of dropped samples since the streaming was started

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.7 void DepthSense::ColorNode::NewSampleReceived-Event::disconnect (void(∗)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.8 void DepthSense::ColorNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

obj	the object for which the event was raised
colorMap	The color map. If capture-Configuration::compression is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is BGR, otherwise the output format is YUY2.
compressedData	The compressed data. If capture-Configuration::compression is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.
captureConfiguration	the camera configuration that was in effect at the time of capture
timeOfCapture	the time of capture of the sample, expressed in µs
timeOfArrival	the time of arrival of the sample in the library, expressed in µs
droppedSampleCount	the number of dropped samples since the last newSampleReceived event was raised
cumulativeDroppedSample-Count	the number of dropped samples since the streaming was started

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	handlerFunc is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.9 template<class T > void DepthSense::ColorNode::NewSample-ReceivedEvent::disconnect (void(∗)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

obj	the object for which the event was raised
data	the event parameters
closureData	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumen	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.10 template< class T > void DepthSense::ColorNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData) *closure,* T *closureData*) `[inline]`

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`colorMap`	The color map. If `capture-Configuration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is BGR, otherwise the output format is YUY2.
`compressedData`	The compressed data. If `capture-Configuration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.11 template<class T > void DepthSense::ColorNode::NewSampleReceivedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::ColorNode obj, DepthSense::ColorNode::NewSampleReceivedData data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.9.2.12 template<class T > void DepthSense::ColorNode::NewSampleReceivedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::ColorNode obj,::DepthSense::Pointer< uint8_t > colorMap,::DepthSense::Pointer< uint8_t > compressedData, DepthSense::ColorNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`colorMap`	The color map. If `capture-Configuration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, the output format is BGR, otherwise the output format is YUY2.
`compressedData`	The compressed data. If `capture-Configuration::compression` is DepthSense::COMPRESSION_TYPE_MJPEG, this array contains the compmressed data.
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

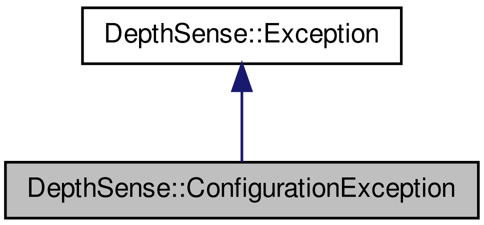
Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.10 DepthSense::ConfigurationException Class Reference

The type of the exception thrown when a valid configuration failed to apply.

Inheritance diagram for DepthSense::ConfigurationException:



Protected Member Functions

- **ConfigurationException** (void ∗data)

6.10.1 Detailed Description

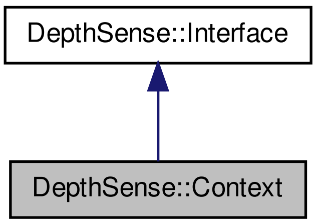
ConfigurationException is thrown when a valid configuration could not be applied because of a device or software error.

Contrast with ArgumentException, which is thrown by AudioNode::setConfiguration(), ColorNode::setConfiguration() and DepthNode::setConfiguration() when the user-provided configuration is invalid.

6.11 DepthSense::Context Class Reference

Represents an application session.

Inheritance diagram for DepthSense::Context:



Classes

- struct ClientConnectedData

  Holds the DepthSense::Context::ClientConnectedEvent arguments.
- class ClientConnectedEvent

  Event raised when a client connects.
- struct ClientDisconnectedData

  Holds the DepthSense::Context::ClientDisconnectedEvent arguments.
- class ClientDisconnectedEvent

  Event raised when a client disconnects.
- struct DeviceAddedData

  Holds the DepthSense::Context::DeviceAddedEvent arguments.
- class DeviceAddedEvent

  Event raised when a camera device is attached to the host.
- struct DeviceRemovedData

  Holds the DepthSense::Context::DeviceRemovedEvent arguments.
- class DeviceRemovedEvent

  Event raised when a camera device is detached from the host.

Public Member Functions

- DepthSense::Context::ClientConnectedEvent & clientConnectedEvent () const

  Returns the `clientConnected` *event object.*
- DepthSense::Context::ClientDisconnectedEvent & clientDisconnectedEvent () const

  Returns the `clientDisconnected` *event object.*
- DepthSense::Context::DeviceAddedEvent & deviceAddedEvent () const

Returns the `deviceAdded` event object.

- DepthSense::Context::DeviceRemovedEvent & deviceRemovedEvent () const

  Returns the `deviceRemoved` event object.
- DepthSense::Version getClientVersion ()

  Client-side version information.
- std::vector< DepthSense::Device > getDevices ()

  Gets the value of the Context::devices property.
- std::vector< DepthSense::Node > getRegisteredNodes ()

  Gets the value of the Context::registeredNodes property.
- DepthSense::Version getServerVersion ()

  Server-side version information.
- void quit ()

  Terminates the DepthSense event loop.
- void registerNode (DepthSense::Node node)

  Starts monitoring a node.
- void releaseControl (DepthSense::Device device)

  Releases control of a device.
- void releaseControl (DepthSense::Node node)

  Releases control of a node.
- void requestControl (DepthSense::Device device)

  Requests control of a device.
- void requestControl (DepthSense::Device device, int32_t timeout)

  Requests control of a device.
- void requestControl (DepthSense::Node node)

  Requests control of a node.
- void requestControl (DepthSense::Node node, int32_t timeout)

  Requests control of a node.
- void run ()

  Runs the DepthSense event loop.
- void startNodes ()

  Starts the capture on the registered nodes.
- void stopNodes ()

  Stops the capture on the registered nodes.
- void unregisterNode (DepthSense::Node node)

  Stops monitoring a node.

Static Public Member Functions

- static DepthSense::Context create ()

  Connects to a DepthSense server.
- static DepthSense::Context create (const char ∗hostname)

  Connects to a DepthSense server.
- static DepthSense::Context create (const char ∗hostname, int32_t port)

Connects to a DepthSense server.

- static DepthSense::Context createStandalone ()

  Creates a standalone DepthSense context.
- static DepthSense::Type type ()

  Returns the DepthSense::Context type object.

Properties

- std::vector< DepthSense::Device > devices

  The list of connected devices.
- std::vector< DepthSense::Node > registeredNodes

  The list of registered nodes.

6.11.1 Detailed Description

The Context class represents an application session (either a standalone session or a TCP/IP client connection). A context can group a number of nodes from different devices for simultaneous monitoring of several stream data sources.

Object instances obtained from a given Context instance (such as the list of devices connected to the host, or the list of nodes belonging to a specific device) are implicitly attached to that Context instance, which is then termed the object's *parent context*. Any operation performed on a device or node automatically makes use of its parent context.

6.11.2 Member Function Documentation

6.11.2.1 DepthSense::Context::ClientConnectedEvent& DepthSense::Context::clientConnectedEvent ( ) const

Returns a reference to the `clientConnected` event object, which can be used to connect handlers to that event.

Returns

the `clientConnected` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.11.2.2 DepthSense::Context::ClientDisconnectedEvent& DepthSense::Context::clientDisconnectedEvent ( ) const

Returns a reference to the `clientDisconnected` event object, which can be used to connect handlers to that event.

Returns

the `clientDisconnected` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.11.2.3 static DepthSense::Context DepthSense::Context::create ( ) `[static]`

Connects to host `localhost`, port `6809`.

Returns

the resulting context

Precondition

no standalone context must be active in the client application

See also

create(const char∗), create(const char∗, int32_t), createStandalone()

Exceptions

DepthSense::InvalidO	a standalone context is active
DepthSense::Initializat	an initialization error has occurred
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.4 static DepthSense::Context DepthSense::Context::create (const char ∗ *hostname*) `[static]`

Connects to host `hostname`, port `6809`.

Parameters

hostname	the host or IP address to connect to

Returns

the resulting context

Warning

the `hostname` parameter is currently ignored, `localhost` is always used

Precondition

no standalone context must be active in the client application

See also

create(), create(const char∗, int32_t), createStandalone()

Exceptions

DepthSense::InvalidOp	a standalone context is active
DepthSense::Initializat	an initialization error has occurred
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.5 static DepthSense::Context DepthSense::Context::create (const char ∗ *hostname*, int32_t *port*) `[static]`

Connects to host `hostname`, port `port`.

Parameters

hostname	the host or IP address to connect to
port	the port to connect to

Returns

the resulting context

Warning

the `hostname` parameter is currently ignored, `localhost` is always used

Precondition

no standalone context must be active in the client application

See also

create(), create(const char∗), createStandalone()

Exceptions

DepthSense::InvalidOp	a standalone context is active
DepthSense::Initializat	an initialization error has occurred
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.6 static DepthSense::Context DepthSense::Context::createStandalone ( ) [static]

Creates a standalone DepthSense context.

Returns

the resulting context

Precondition

no standalone context must be active in the client application

See also

create(), create(const char∗), create(const char∗, int32_t)

Exceptions

DepthSense::InvalidOp	a standalone context is active
DepthSense::Initializat	an initialization error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.7 DepthSense::Context::DeviceAddedEvent& DepthSense::Context::deviceAddedEvent( ) const

Returns a reference to the `deviceAdded` event object, which can be used to connect handlers to that event.

Returns

the `deviceAdded` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.11.2.8 DepthSense::Context::DeviceRemovedEvent& DepthSense::Context::deviceRemovedEvent( ) const

Returns a reference to the `deviceRemoved` event object, which can be used to connect handlers to that event.

Returns

the `deviceRemoved` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.11.2.9 DepthSense::Version DepthSense::Context::getClientVersion( )

The Context::getClientVersion method returns the client-side version information.

Returns

the client-side version information

See also

Context::getServerVersion
getLibraryVersion

Exceptions

DepthSense::InvalidO	The current context is unset, or not connected to the DepthSense server, or is a standalone context
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.10 std::vector< DepthSense::Device > DepthSense::Context::getDevices ( ) [inline]

Gets the value of the Context::devices property.

The Context::devices property contains the list of camera devices attached to the host.

Returns

the value of the Context::devices property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.11 std::vector< DepthSense::Node > DepthSense::Context::getRegistered-Nodes ( ) [inline]

Gets the value of the Context::registeredNodes property.

The Context::registeredNodes property contains the list of nodes registered with registerNode().

See also

registerNode(), unregisterNode()

Returns

the value of the Context::registeredNodes property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.12 DepthSense::Version DepthSense::Context::getServerVersion ( )

The Context::getServerVersion method returns the server-side version information.

Returns

the server-side version information

See also

Context::getClientVersion
getLibraryVersion

Exceptions

DepthSense::InvalidO	The current context is unset, or not connected to the DepthSense server, or is a standalone context
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.13 void DepthSense::Context::quit ( )

Terminates the DepthSense event loop.

See also

run()

Exceptions

DepthSense::InvalidO	the DepthSense event loop is not running in the current application
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.14 void DepthSense::Context::registerNode (DepthSense::Node *node*)

Registers a node with the current context. All registered nodes will be used as actual stream data sources. If the streaming is already started, the registered node will automatically start streaming.

Parameters

node	the node to be registered

See also

unregisterNode(), getRegisteredNodes()

Exceptions

DepthSense::Argumer	`node` is unset, already registered or the node is an DepthSense::UnsupportedNode node
DepthSense::Streamir	a streaming error has occured
DepthSense::Configur	a valid configuration failed to apply
DepthSense::InvalidO	when video synchronization is enabled, the configurations of the depth and color nodes are incompatible
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.15 void DepthSense::Context::releaseControl (DepthSense::Device *device*)

Releases full control access on `device`. If other clients/contexts are waiting for full control access on the same device or one of its nodes, control will be transferred to one of them, chosen randomly.

Parameters

device	the device to release control of

See also

requestControl(Device), requestControl(Device, int32_t), requestControl(Node), requestControl(Node, int32_t), releaseControl(Node)

Exceptions

DepthSense::Argumer	`device` is unset, has been disconnected from the host, or the current context does not control it
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.16 void DepthSense::Context::releaseControl (DepthSense::Node *node*)

Releases full control access on `node`. If other clients/contexts are waiting for full control access on the same node control will be transferred to one of them, chosen randomly.

Parameters

node	the node to release control of

See also

requestControl(Device), requestControl(Device, int32_t), requestControl(Node), requestControl(Node, int32_t), releaseControl(Device)

Exceptions

DepthSense::Argumer	`node` is unset, has been disconnected from the host, or the current context does not control it
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.17 void DepthSense::Context::requestControl (DepthSense::Device *device*)

Requests full control access on `device` and its nodes. This method provides a cooperation mechanism which allows multiple clients to share control over a specific camera device. Only one client (i.e. one context) at a time can modify the configuration of a device or any of its exposed properties.

This methods blocks indefinitely until control is granted to the caller or the device is detached from the host system.

Parameters

device	the device to request control of

See also

requestControl(Device, int32_t), requestControl(Node), requestControl(Node, int32_t), releaseControl(Device), releaseControl(Node)

Exceptions

DepthSense::Argumer	`device` is unset, has been disconnected from the host, or the current context already controls it
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.18 void DepthSense::Context::requestControl (DepthSense::Device *device*, int32_t *timeout*)

Requests full control access on `device` and its nodes. This method provides a cooperation mechanism which allows multiple clients to share control over a specific camera device. Only one client (i.e. one context) at a time can modify the configuration of a device or any of its exposed properties. This method will try to request the control of all the nodes of the DepthSense::Device. In case of failure, none of the nodes will be controlled. When a context has the control over a DepthSense::Device, the context will be granted control on any new node added to the DepthSense::Device.

Parameters

device	the device to request control of
timeout	the timeout in milliseconds

See also

requestControl(Device), requestControl(Node), requestControl(Node, int32_t), releaseControl(Device), releaseControl(Node)

Exceptions

DepthSense::Argumer	`device` is unset, has been disconnected from the host, or the current context already controls it
DepthSense::Timeoutl	`timeout` has expired before control could be obtained
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.19 void DepthSense::Context::requestControl (DepthSense::Node *node*)

Requests full control access on `node`. This method provides a cooperation mechanism which allows multiple clients to share control over a specific camera device node. Only one client (i.e. one context) at a time can modify the configuration of a node or any of its exposed properties.

This methods blocks indefinitely until control is granted to the caller or the node is detached from the host system.

Parameters

node	the node to request control of

See also

requestControl(Device), requestControl(Device, int32_t), requestControl(Node, int32_t), releaseControl(Device), releaseControl(Node)

Exceptions

DepthSense::Argumer	`node` is unset, has been disconnected from the host, the current context already controls it, or the node is an DepthSense::UnsupportedNode node
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.20 void DepthSense::Context::requestControl (DepthSense::Node *node*, int32_t *timeout*)

Requests full control access on `node`. This method provides a cooperation mechanism which allows multiple clients to share control over a specific camera device node. Only one client (i.e. one context) at a time can modify the configuration of a node or any of its exposed properties.

Parameters

node	the node to request control of
timeout	the timeout in milliseconds

See also

requestControl(Device), requestControl(Device, int32_t), requestControl(Node), releaseControl(Device), releaseControl(Node)

Exceptions

DepthSense::Argumer	`node` is unset, has been disconnected from the host, the current context already controls it, or the node is an DepthSense::UnsupportedNode node
DepthSense::TimeoutE	`timeout` has expired before control could be obtained
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.21 void DepthSense::Context::run ( )

Runs the DepthSense event loop. The connected event handlers are run in the thread that called run().

If the server throws an exception asynchronously (that is, not in reaction to a method call or property assignment/retrieval), it will be propagated to this method.

To exit the event loop, call quit().

See also

quit()

Exceptions

DepthSense::InvalidO	the DepthSense event loop is already running in the current application
DepthSense::Initializat	an initialization error has occurred
DepthSense::Streamir	a streaming error has occurred
DepthSense::Transpor	a network or protocol error has occurred
DepthSense::Configur	a valid configuration failed to apply
DepthSense::IOExcep	a device I/O operation has failed
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.22 void DepthSense::Context::startNodes ( )

Starts the capture (streaming) on the nodes registered with the current context.

See also

stopNodes()

Exceptions

DepthSense::Configur	a valid node configuration failed to apply
DepthSense::Streamir	streaming could not be started
DepthSense::InvalidO	when video synchronization is enabled, the configurations of the depth and color nodes are incompatible
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.23 void DepthSense::Context::stopNodes ( )

Stops the capture (streaming) on the nodes registered with the current context.

See also

startNodes()

Exceptions

DepthSense::Streamir	streaming could not be stopped
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.2.24 static DepthSense::Type DepthSense::Context::type ( ) [static]

Returns the DepthSense::Context type object

Returns

the DepthSense::Context type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented from DepthSense::Interface.

6.11.2.25 void DepthSense::Context::unregisterNode (DepthSense::Node *node*)

Removes the specified node from the list of monitored nodes of the current context.

Parameters

node	the node to be unregistered

Precondition

The provided node must have been subject to a prior call to registerNode().

See also

registerNode(), getRegisteredNodes()

Exceptions

DepthSense::Argumer	`node` is unset or not registered
DepthSense::Streamir	a streaming error has occured
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.11.3 Property Documentation

6.11.3.1 std::vector< DepthSense::Device > DepthSense::Context::devices `[read, assign]`

The Context::devices property contains the list of camera devices attached to the host.

6.11.3.2 std::vector< DepthSense::Node > DepthSense::Context::registeredNodes `[read, assign]`

The Context::registeredNodes property contains the list of nodes registered with registerNode().

See also

registerNode(), unregisterNode()

6.12 DepthSense::Context::ClientConnectedData Struct Reference

Holds the DepthSense::Context::ClientConnectedEvent arguments.

Public Attributes

- std::string appName

  the name of the client executable, or an empty string if it could not be determined
- int32_t pid

  the process ID of the client, or `-1` *if it could not be determined*
- std::string sourceIP

  the source IP address
- int32_t sourcePort

  the source IP port

6.12.1 Detailed Description

The ClientConnectedData struct holds the DepthSense::Context::ClientConnectedEvent parameters and is passed to callbacks connected to that event.

6.13 DepthSense::Context::ClientConnectedEvent Class Reference

Event raised when a client connects.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data, T closureData), T closureData)

  Connects a closure to the current event.

- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Disconnects a method from the current event.

6.13.1 Detailed Description

The `clientConnected` event is raised when a client connects to the DepthSense server.

Parameters

appName	the name of the client executable, or an empty string if it could not be determined
pid	the process ID of the client, or `-1` if it could not be determined
sourceIP	the source IP address
sourcePort	the source IP port

See also

ClientDisconnectedEvent

6.13.2 Member Function Documentation

6.13.2.1 void DepthSense::Context::ClientConnectedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.2 void DepthSense::Context::ClientConnectedEvent::connect (void(∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.3 template<class T > void DepthSense::Context::ClientConnectedEvent::connect (void(*)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data, T closureData) *closure*, T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.4 template<class T > void DepthSense::Context::ClientConnectedEvent::connect (void(∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData) *closure*, T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.5 template<class T > void DepthSense::Context::ClientConnectedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data) *method*) [inline]

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.6 template< class T > void DepthSense::Context::ClientConnectedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.7 void DepthSense::Context::ClientConnectedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.8 void DepthSense::Context::ClientConnectedEvent::disconnect (void(∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.9 template< class T > void DepthSense::Context::ClientConnectedEvent::disconnect (void(*)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data, T closureData) *closure*, T *closureData*) `[inline]`

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.10 template< class T > void DepthSense::Context::ClientConnectedEvent::disconnect (void(*)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData) *closure*, T *closureData*) `[inline]`

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.11 template<class T > void DepthSense::Context::ClientConnectedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::Context obj, DepthSense::Context::ClientConnectedData data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumen	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.13.2.12 template<class T > void DepthSense::Context::ClientConnectedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumen	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.14 DepthSense::Context::ClientDisconnectedData Struct - Reference

Holds the DepthSense::Context::ClientDisconnectedEvent arguments.

Public Attributes

- std::string appName

  the name of the client executable, or an empty string if it could not be determined
- int32_t pid

  the process ID of the client, or `-1` if it could not be determined
- std::string sourceIP

  the source IP address
- int32_t sourcePort

  the source IP port

6.14.1 Detailed Description

The ClientDisconnectedData struct holds the DepthSense::Context::ClientDisconnectedEvent parameters and is passed to callbacks connected to that event.

6.15 DepthSense::Context::ClientDisconnectedEvent Class - Reference

Event raised when a client disconnects.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedDat
  data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::ClientDisconnecte
  data))

  Connects a method to the current event.

- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort))

  Disconnects a method from the current event.

6.15.1 Detailed Description

The `clientDisconnected` event is raised when a client disconnects from the DepthSense server.

Parameters

appName	the name of the client executable, or an empty string if it could not be determined
pid	the process ID of the client, or `-1` if it could not be determined
sourceIP	the source IP address
sourcePort	the source IP port

See also

ClientConnectedEvent

6.15.2 Member Function Documentation

6.15.2.1 void DepthSense::Context::ClientDisconnectedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.2 void DepthSense::Context::ClientDisconnectedEvent::connect (void(∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.3 template<class T > void DepthSense::Context::ClientDisconnectedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.4 template<class T > void DepthSense::Context::ClientDisconnectedEvent::connect (void(∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.5 template<class T > void DepthSense::Context::ClientDisconnectedEvent::connect (T * *obj*, void(T::*)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.6 template<class T > void DepthSense::Context::ClientDisconnectedEvent::connect (T * *obj,* void(T::*)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.7 void DepthSense::Context::ClientDisconnectedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.8 void DepthSense::Context::ClientDisconnectedEvent::disconnect (void(∗)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.9 template<class T > void DepthSense::Context::ClientDisconnectedEvent::disconnect (void(*)(DepthSense::Context obj, DepthSense::Context::ClientDisconnectedData data, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.10 template<class T > void DepthSense::Context::ClientDisconnectedEvent::disconnect (void(*)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.11 template<class T > void **DepthSense::Context::ClientDisconnectedEvent::disconnect** (T * *obj*, void(T::*)(**DepthSense::Context** obj, **DepthSense::Context::ClientDisconnectedData** data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.15.2.12 template<class T > void DepthSense::Context::ClientDisconnectedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::Context obj, std::string appName, int32_t pid, std::string sourceIP, int32_t sourcePort) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`appName`	the name of the client executable, or an empty string if it could not be determined
`pid`	the process ID of the client, or `-1` if it could not be determined
`sourceIP`	the source IP address
`sourcePort`	the source IP port

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.16 DepthSense::Context::DeviceAddedData Struct Reference

Holds the DepthSense::Context::DeviceAddedEvent arguments.

Public Attributes

- DepthSense::Device device

  the camera device that was attached to the host

6.16.1 Detailed Description

The DeviceAddedData struct holds the DepthSense::Context::DeviceAddedEvent parameters and is passed to callbacks connected to that event.

6.17 DepthSense::Context::DeviceAddedEvent Class Reference

Event raised when a camera device is attached to the host.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Device device))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, DepthSense::Device device, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Device device))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Device device))

  Disconnects a function from the current event.

- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, DepthSense::Device device, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Device device))

  Disconnects a method from the current event.

6.17.1 Detailed Description

The `deviceAdded` event is raised when a camera device is attached to the host.

Parameters

device	the camera device that was attached to the host

See also

DeviceRemovedEvent

6.17.2 Member Function Documentation

6.17.2.1 void DepthSense::Context::DeviceAddedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.2 void DepthSense::Context::DeviceAddedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Device device) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`device`	the camera device that was attached to the host

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.3 template<class T > void DepthSense::Context::DeviceAddedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data, T closureData) *closure,* T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.4 template<class T > void DepthSense::Context::DeviceAddedEvent::connect (void(*)(DepthSense::Context obj, DepthSense::Device device, T closureData) *closure,* T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`device`	the camera device that was attached to the host
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.5 template<class T > void DepthSense::Context::DeviceAddedEvent::connect (T * *obj*, void(T::*)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data) *method*) [inline]

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.6 template<class T > void DepthSense::Context::DeviceAddedEvent::connect (T * *obj*, void(T::*)(DepthSense::Context obj, DepthSense::Device device) *method*) [inline]

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`device`	the camera device that was attached to the host

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.7 void DepthSense::Context::DeviceAddedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.8 void DepthSense::Context::DeviceAddedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Device device) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`device`	the camera device that was attached to the host

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.9 template<class T > void DepthSense::Context::DeviceAddedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.10 template<class T > void DepthSense::Context::DeviceAddedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Device device, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`device`	the camera device that was attached to the host
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.11 template<class T > void DepthSense::Context::DeviceAddedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::Context obj, DepthSense::Context::DeviceAddedData data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.17.2.12 template<class T > void DepthSense::Context::DeviceAddedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::Context obj, DepthSense::Device device) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`device`	the camera device that was attached to the host

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.18 DepthSense::Context::DeviceRemovedData Struct Reference

Holds the DepthSense::Context::DeviceRemovedEvent arguments.

Public Attributes

- DepthSense::Device device

  the camera device that was detached from the host

6.18.1 Detailed Description

The DeviceRemovedData struct holds the DepthSense::Context::DeviceRemovedEvent parameters and is passed to callbacks connected to that event.

6.19 DepthSense::Context::DeviceRemovedEvent Class Reference

Event raised when a camera device is detached from the host.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Device device))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Context obj, DepthSense::Device device, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Device device))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Context obj, DepthSense::Device device))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Context obj, DepthSense::Device device, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data))

Disconnects a method from the current event.

- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Context obj, DepthSense::Device device))

  Disconnects a method from the current event.

6.19.1 Detailed Description

The `deviceAdded` event is raised when a camera device is detached from the host.

Parameters

device	the camera device that was detached from the host

See also

DeviceAddedEvent

6.19.2 Member Function Documentation

6.19.2.1 void DepthSense::Context::DeviceRemovedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.2 void DepthSense::Context::DeviceRemovedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Device device) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`device`	the camera device that was detached from the host

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.3 template<class T > void DepthSense::Context::DeviceRemovedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event

std::bad_alloc	not enough memory to perform the requested operation

6.19.2.4 template<class T > void DepthSense::Context::DeviceRemovedEvent::connect (void(∗)(DepthSense::Context obj, DepthSense::Device device, T closureData) *closure,* T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`device`	the camera device that was detached from the host
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.5 template<class T > void DepthSense::Context::DeviceRemovedEvent::connect (T ∗ *obj,* void(T::∗)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.6 template<class T > void DepthSense::Context::DeviceRemovedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::Context obj, DepthSense::Device device) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`device`	the camera device that was detached from the host

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.7 void DepthSense::Context::DeviceRemovedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.8 void DepthSense::Context::DeviceRemovedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Device device) *handlerFunc*) [inline]

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`device`	the camera device that was detached from the host

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.9 template<class T > void DepthSense::Context::DeviceRemovedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumen	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.10 template<class T > void DepthSense::Context::DeviceRemovedEvent::disconnect (void(∗)(DepthSense::Context obj, DepthSense::Device device, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`device`	the camera device that was detached from the host
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.11 template<class T > void DepthSense::Context::DeviceRemovedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::Context obj, DepthSense::Context::DeviceRemovedData data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.19.2.12 template<class T > void DepthSense::Context::DeviceRemovedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::Context obj, DepthSense::Device device) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`device`	the camera device that was detached from the host

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

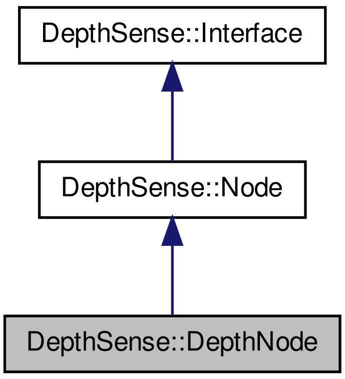
Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.20 DepthSense::DepthNode Class Reference

Represents a depth stream data source.

Inheritance diagram for DepthSense::DepthNode:



Classes

- struct Acceleration

  The acceleration returned by the camera.
- struct Configuration

  The configuration of a depth node.
- struct NewSampleReceivedData

  Holds the DepthSense::DepthNode::NewSampleReceivedEvent arguments.
- class NewSampleReceivedEvent

  Event raised when a depth sample is captured.

Public Types

- enum CameraMode { **CAMERA_MODE_CLOSE_MODE** = 0, **CAMERA_MODE-_LONG_RANGE** = 1 }

  The mode of the camera.

Public Member Functions

- bool _noFanIsReadOnly ()

  Checks whether property DepthNode::_noFan is read-only.
- bool confidenceThresholdIsReadOnly ()

  Checks whether property DepthNode::confidenceThreshold is read-only.
- bool configurationIsReadOnly ()

  Checks whether property DepthNode::configuration is read-only.
- bool depthMap3PlanesIsReadOnly ()

  Checks whether property DepthNode::depthMap3Planes is read-only.
- bool depthMapFloatingPoint3PlanesIsReadOnly ()

  Checks whether property DepthNode::depthMapFloatingPoint3Planes is read-only.
- bool enableAccelerometerIsReadOnly ()

  Checks whether property DepthNode::enableAccelerometer is read-only.
- bool enableConfidenceMapIsReadOnly ()

  Checks whether property DepthNode::enableConfidenceMap is read-only.
- bool enableDenoisingIsReadOnly ()

  Checks whether property DepthNode::enableDenoising is read-only.
- bool enableDepthMapFloatingPointIsReadOnly ()

  Checks whether property DepthNode::enableDepthMapFloatingPoint is read-only.
- bool enableDepthMapIsReadOnly ()

  Checks whether property DepthNode::enableDepthMap is read-only.
- bool enablePhaseMapIsReadOnly ()

  Checks whether property DepthNode::enablePhaseMap is read-only.
- bool enableUvMapIsReadOnly ()

  Checks whether property DepthNode::enableUvMap is read-only.

- bool enableVerticesFloatingPointIsReadOnly ()

  Checks whether property DepthNode::enableVerticesFloatingPoint is read-only.
- bool enableVerticesIsReadOnly ()

  Checks whether property DepthNode::enableVertices is read-only.
- bool get_noFan ()

  Gets the value of the DepthNode::_noFan property.
- int32_t getConfidenceThreshold ()

  Gets the value of the DepthNode::confidenceThreshold property.
- DepthSense::DepthNode::Configuration getConfiguration ()

  Gets the value of the DepthNode::configuration property.
- std::vector < DepthSense::DepthNode::Configuration > getConfigurations ()

  Gets the value of the DepthNode::configurations property.
- bool getDepthMap3Planes ()

  Gets the value of the DepthNode::depthMap3Planes property.
- bool getDepthMapFloatingPoint3Planes ()

  Gets the value of the DepthNode::depthMapFloatingPoint3Planes property.
- bool getEnableAccelerometer ()

  Gets the value of the DepthNode::enableAccelerometer property.
- bool getEnableConfidenceMap ()

  Gets the value of the DepthNode::enableConfidenceMap property.
- bool getEnableDenoising ()

  Gets the value of the DepthNode::enableDenoising property.
- bool getEnableDepthMap ()

  Gets the value of the DepthNode::enableDepthMap property.
- bool getEnableDepthMapFloatingPoint ()

  Gets the value of the DepthNode::enableDepthMapFloatingPoint property.
- bool getEnablePhaseMap ()

  Gets the value of the DepthNode::enablePhaseMap property.
- bool getEnableUvMap ()

  Gets the value of the DepthNode::enableUvMap property.
- bool getEnableVertices ()

  Gets the value of the DepthNode::enableVertices property.
- bool getEnableVerticesFloatingPoint ()

  Gets the value of the DepthNode::enableVerticesFloatingPoint property.
- int32_t getIlluminationLevel ()

  Gets the value of the DepthNode::illuminationLevel property.
- float getRange ()

  Gets the value of the DepthNode::range property.
- bool illuminationLevelIsReadOnly ()

  Checks whether property DepthNode::illuminationLevel is read-only.
- DepthSense::DepthNode::NewSampleReceivedEvent & newSampleReceivedEvent () const

  Returns the `newSampleReceived` event object.

- void set_noFan (bool value)

  Sets the value of the DepthNode::_noFan property.
- void setConfidenceThreshold (int32_t value)

  Sets the value of the DepthNode::confidenceThreshold property.
- void setConfiguration (DepthSense::DepthNode::Configuration value)

  Sets the value of the DepthNode::configuration property.
- void setDepthMap3Planes (bool value)

  Sets the value of the DepthNode::depthMap3Planes property.
- void setDepthMapFloatingPoint3Planes (bool value)

  Sets the value of the DepthNode::depthMapFloatingPoint3Planes property.
- void setEnableAccelerometer (bool value)

  Sets the value of the DepthNode::enableAccelerometer property.
- void setEnableConfidenceMap (bool value)

  Sets the value of the DepthNode::enableConfidenceMap property.
- void setEnableDenoising (bool value)

  Sets the value of the DepthNode::enableDenoising property.
- void setEnableDepthMap (bool value)

  Sets the value of the DepthNode::enableDepthMap property.
- void setEnableDepthMapFloatingPoint (bool value)

  Sets the value of the DepthNode::enableDepthMapFloatingPoint property.
- void setEnablePhaseMap (bool value)

  Sets the value of the DepthNode::enablePhaseMap property.
- void setEnableUvMap (bool value)

  Sets the value of the DepthNode::enableUvMap property.
- void setEnableVertices (bool value)

  Sets the value of the DepthNode::enableVertices property.
- void setEnableVerticesFloatingPoint (bool value)

  Sets the value of the DepthNode::enableVerticesFloatingPoint property.
- void setIlluminationLevel (int32_t value)

  Sets the value of the DepthNode::illuminationLevel property.

Static Public Member Functions

- static std::string CameraMode_toString (CameraMode value)

  Converts a DepthSense::DepthNode::CameraMode value to a string.
- static DepthSense::Type type ()

  Returns the DepthSense::DepthNode type object.

Properties

- bool **_noFan**
- int32_t confidenceThreshold

  Specify the confidence threshold.
- DepthSense::DepthNode::Configuration configuration

  The node configuration.
- std::vector < DepthSense::DepthNode::Configuration > configurations

  The list of supported node configurations.
- bool depthMap3Planes

  Whether the depth map is the XYZ coordinates as a planar representation or only the Z coordinate.
- bool depthMapFloatingPoint3Planes

  Whether the floating point depth map is the XYZ coordinates as a planar representation or only the Z coordinate.
- bool enableAccelerometer

  Whether to enable the accelerometer data.
- bool enableConfidenceMap

  Whether to enable confidence map computation.
- bool enableDenoising

  Whether to enable denoising.
- bool enableDepthMap

  Whether to enable fixed point depth map computation.
- bool enableDepthMapFloatingPoint

  Whether to enable floating point depth map computation.
- bool enablePhaseMap

  Whether to enable phase map computation.
- bool enableUvMap

  Whether to enable UV map computation.
- bool enableVertices

  Whether to enable fixed point vertices computation.
- bool enableVerticesFloatingPoint

  Whether to enable floating point vertices computation.
- int32_t illuminationLevel

  Specify the illumination level.
- float range

  The depth sensor range.

6.20.1 Detailed Description

The DepthNode class allows to capture depth data with the depth sensor of a given camera device.

6.20.2 Member Enumeration Documentation

6.20.2.1 enum DepthSense::DepthNode::CameraMode

A type enumerating the various operating modes supported by depth sensors.

6.20.3 Member Function Documentation

6.20.3.1 bool DepthSense::DepthNode::_noFanIsReadOnly ( )

Checks whether property DepthNode::_noFan is read-only.

Returns

whether property DepthNode::_noFan is read-only

See also

set_noFan()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.2 static std::string DepthSense::DepthNode::CameraMode_toString (CameraMode *value*) `[inline, static]`

Converts the provided enumeration value to a string.

Parameters

value	the enumeration value to convert

Returns

the name of the enumeration member whose value is `value`, or, if `value` is not a member of DepthSense::DepthNode::CameraMode, its numeric representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.20.3.3 bool DepthSense::DepthNode::confidenceThresholdIsReadOnly ( )

Checks whether property DepthNode::confidenceThreshold is read-only.

The DepthNode::confidenceThreshold property specifies the confidence threshold.

The DepthNode::confidenceThreshold property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property DepthNode::confidenceThreshold is read-only

See also

setConfidenceThreshold()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.4 bool DepthSense::DepthNode::configurationIsReadOnly ( )

Checks whether property DepthNode::configuration is read-only.

The DepthNode::configuration property specifies the configuration of the depth node.

Exceptions

DepthSense::InvalidO	when video synchronization is enabled, the configurations of the depth and color nodes are incompatible or the operation cannot be performed on this node

Returns

whether property DepthNode::configuration is read-only

See also

setConfiguration()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.5 bool DepthSense::DepthNode::depthMap3PlanesIsReadOnly ( )

Checks whether property DepthNode::depthMap3Planes is read-only.

The DepthNode::depthMap3Planes property specifies whether the `depthMap` buffer is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property DepthNode::depthMap3Planes is read-only

See also

setDepthMap3Planes()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.6 bool DepthSense::DepthNode::depthMapFloatingPoint3PlanesIsReadOnly ( )

Checks whether property DepthNode::depthMapFloatingPoint3Planes is read-only.

The DepthNode::depthMapFloatingPoint3Planes property specifies whether the `depthMapFloatingPoint` is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property DepthNode::depthMapFloatingPoint3Planes is read-only

See also

setDepthMapFloatingPoint3Planes()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.7 bool DepthSense::DepthNode::enableAccelerometerIsReadOnly ( )

Checks whether property DepthNode::enableAccelerometer is read-only.

The DepthNode::enableAccelerometer property specifies whether to capture the accelerometer data and make it available through the `acceleration` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property DepthNode::enableAccelerometer is read-only

See also

setEnableAccelerometer()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.8 bool DepthSense::DepthNode::enableConfidenceMapIsReadOnly ( )

Checks whether property DepthNode::enableConfidenceMap is read-only.

The DepthNode::enableConfidenceMap property specifies whether to enable confidence map computation and make it available through the `confidenceMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property DepthNode::enableConfidenceMap is read-only

See also

setEnableConfidenceMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.9 bool DepthSense::DepthNode::enableDenoisingIsReadOnly ( )

Checks whether property DepthNode::enableDenoising is read-only.

The DepthNode::enableDenoising property specifies whether to enable denoising.

Note: The denoising filter will be applied only on the phase map and not on the other maps.

The DepthNode::enableDenoising property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property DepthNode::enableDenoising is read-only

See also

setEnableDenoising()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.10 bool DepthSense::DepthNode::enableDepthMapFloatingPointIsReadOnly ( )

Checks whether property DepthNode::enableDepthMapFloatingPoint is read-only.

The DepthNode::enableDepthMapFloatingPoint property specifies whether to enable floating point depth map computation and make it available through the `depthMapFloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

whether property DepthNode::enableDepthMapFloatingPoint is read-only

See also

setEnableDepthMapFloatingPoint()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.11 bool DepthSense::DepthNode::enableDepthMapIsReadOnly ( )

Checks whether property DepthNode::enableDepthMap is read-only.

The DepthNode::enableDepthMap property specifies whether to enable fixed point depth map computation and make it available through the `depthMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property DepthNode::enableDepthMap is read-only

See also

setEnableDepthMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.12 bool DepthSense::DepthNode::enablePhaseMapIsReadOnly ( )

Checks whether property DepthNode::enablePhaseMap is read-only.

The DepthNode::enablePhaseMap property specifies whether to enable phase map computation and make it available through the `phaseMap` argument of the `new-SampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property DepthNode::enablePhaseMap is read-only

See also

setEnablePhaseMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.13 bool DepthSense::DepthNode::enableUvMapIsReadOnly ( )

Checks whether property DepthNode::enableUvMap is read-only.

The DepthNode::enableUvMap property specifies whether to enable UV map computation and make it available through the `uvMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property DepthNode::enableUvMap is read-only

See also

setEnableUvMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.14 bool DepthSense::DepthNode::enableVerticesFloatingPointIsReadOnly ( )

Checks whether property DepthNode::enableVerticesFloatingPoint is read-only.

The DepthNode::enableVertices property specifies whether to enable floating point vertices computation and make it available through the `verticesFloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property DepthNode::enableVerticesFloatingPoint is read-only

See also

setEnableVerticesFloatingPoint()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.15 bool DepthSense::DepthNode::enableVerticesIsReadOnly ( )

Checks whether property DepthNode::enableVertices is read-only.

The DepthNode::enableVertices property specifies whether to enable fixed point vertices computation and make it available through the `vertices` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property DepthNode::enableVertices is read-only

See also

setEnableVertices()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.16 bool DepthSense::DepthNode::get_noFan ( )

Gets the value of the DepthNode::_noFan property.

Returns

the value of the DepthNode::_noFan property

See also

set_noFan()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.17 int32_t DepthSense::DepthNode::getConfidenceThreshold ( )

Gets the value of the DepthNode::confidenceThreshold property.

The DepthNode::confidenceThreshold property specifies the confidence threshold.

The DepthNode::confidenceThreshold property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::confidenceThreshold property

See also

setConfidenceThreshold()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.18 DepthSense::DepthNode::Configuration DepthSense::DepthNode::getConfiguration ( )

Gets the value of the DepthNode::configuration property.

The DepthNode::configuration property specifies the configuration of the depth node.

Exceptions

DepthSense::InvalidO	when video synchronization is enabled, the configurations of the depth and color nodes are incompatible or the operation cannot be performed on this node

Returns

the value of the DepthNode::configuration property

See also

setConfiguration()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.19 std::vector< DepthSense::DepthNode::Configuration > DepthSense::DepthNode::getConfigurations( ) [inline]

Gets the value of the DepthNode::configurations property.

Returns

the value of the DepthNode::configurations property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.20 bool DepthSense::DepthNode::getDepthMap3Planes( )

Gets the value of the DepthNode::depthMap3Planes property.

The DepthNode::depthMap3Planes property specifies whether the `depthMap` buffer is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::depthMap3Planes property

See also

setDepthMap3Planes()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.21 bool DepthSense::DepthNode::getDepthMapFloatingPoint3Planes ( )

Gets the value of the DepthNode::depthMapFloatingPoint3Planes property.

The DepthNode::depthMapFloatingPoint3Planes property specifies whether the `depthMapFloatingPoint` is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::depthMapFloatingPoint3Planes property

See also

setDepthMapFloatingPoint3Planes()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.22 bool DepthSense::DepthNode::getEnableAccelerometer ( )

Gets the value of the DepthNode::enableAccelerometer property.

The DepthNode::enableAccelerometer property specifies whether to capture the accelerometer data and make it available through the `acceleration` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableAccelerometer property

See also

setEnableAccelerometer()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.23 bool DepthSense::DepthNode::getEnableConfidenceMap ( )

Gets the value of the DepthNode::enableConfidenceMap property.

The DepthNode::enableConfidenceMap property specifies whether to enable confidence map computation and make it available through the `confidenceMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableConfidenceMap property

See also

setEnableConfidenceMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.24 bool DepthSense::DepthNode::getEnableDenoising ( )

Gets the value of the DepthNode::enableDenoising property.

The DepthNode::enableDenoising property specifies whether to enable denoising.

Note: The denoising filter will be applied only on the phase map and not on the other maps.

The DepthNode::enableDenoising property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableDenoising property

See also

setEnableDenoising()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.25 bool DepthSense::DepthNode::getEnableDepthMap ( )

Gets the value of the DepthNode::enableDepthMap property.

The DepthNode::enableDepthMap property specifies whether to enable fixed point depth map computation and make it available through the `depthMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableDepthMap property

See also

setEnableDepthMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.26 bool DepthSense::DepthNode::getEnableDepthMapFloatingPoint ( )

Gets the value of the DepthNode::enableDepthMapFloatingPoint property.

The DepthNode::enableDepthMapFloatingPoint property specifies whether to enable floating point depth map computation and make it available through the `depthMap-FloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableDepthMapFloatingPoint property

See also

setEnableDepthMapFloatingPoint()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.27 bool DepthSense::DepthNode::getEnablePhaseMap ( )

Gets the value of the DepthNode::enablePhaseMap property.

The DepthNode::enablePhaseMap property specifies whether to enable phase map computation and make it available through the `phaseMap` argument of the `new-SampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the DepthNode::enablePhaseMap property

See also

setEnablePhaseMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.28 bool DepthSense::DepthNode::getEnableUvMap ( )

Gets the value of the DepthNode::enableUvMap property.

The DepthNode::enableUvMap property specifies whether to enable UV map computation and make it available through the `uvMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableUvMap property

See also

setEnableUvMap()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.29 bool DepthSense::DepthNode::getEnableVertices ( )

Gets the value of the DepthNode::enableVertices property.

The DepthNode::enableVertices property specifies whether to enable fixed point vertices computation and make it available through the `vertices` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableVertices property

See also

setEnableVertices()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.30 bool DepthSense::DepthNode::getEnableVerticesFloatingPoint ( )

Gets the value of the DepthNode::enableVerticesFloatingPoint property.

The DepthNode::enableVertices property specifies whether to enable floating point vertices computation and make it available through the `verticesFloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the DepthNode::enableVerticesFloatingPoint property

See also

setEnableVerticesFloatingPoint()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.31 int32_t DepthSense::DepthNode::getIlluminationLevel ( )

Gets the value of the DepthNode::illuminationLevel property.

The DepthNode::illuminationLevel property specifies the illumination level to be used by the camera. This property is currently not available for DS325 cameras.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Returns

the value of the DepthNode::illuminationLevel property

See also

setIlluminationLevel()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.32 float DepthSense::DepthNode::getRange ( )

Gets the value of the DepthNode::range property.

The DepthNode::range property specifies the range of the depth sensor, expressed in meters.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

the value of the DepthNode::range property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.33 bool DepthSense::DepthNode::illuminationLevelIsReadOnly ( )

Checks whether property DepthNode::illuminationLevel is read-only.

The DepthNode::illuminationLevel property specifies the illumination level to be used by the camera. This property is currently not available for DS325 cameras.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Returns

whether property DepthNode::illuminationLevel is read-only

See also

setIlluminationLevel()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.34 DepthSense::DepthNode::NewSampleReceivedEvent& DepthSense::DepthNode::newSampleReceivedEvent ( ) const

Returns a reference to the `newSampleReceived` event object, which can be used to connect handlers to that event.

Returns

the `newSampleReceived` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.20.3.35 void DepthSense::DepthNode::set_noFan (bool *value*)

Sets the value of the DepthNode::_noFan property.

Parameters

value	the value to set

See also

get_noFan(), _noFanIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.36 void DepthSense::DepthNode::setConfidenceThreshold (int32_t *value*)

Sets the value of the DepthNode::confidenceThreshold property.

The DepthNode::confidenceThreshold property specifies the confidence threshold.

The DepthNode::confidenceThreshold property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getConfidenceThreshold(), confidenceThresholdIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the value passed is out of range
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.37 void DepthSense::DepthNode::setConfiguration (DepthSense::DepthNode::Configuration *value*)

Sets the value of the DepthNode::configuration property.

The DepthNode::configuration property specifies the configuration of the depth node.

Exceptions

DepthSense::InvalidO	when video synchronization is enabled, the configurations of the depth and color nodes are incompatible or the operation cannot be performed on this node

Parameters

value	the value to set

See also

getConfiguration(), configurationIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the provided configuration is invalid
DepthSense::Configur	the provided configuration is valid but failed to apply

DepthSense::Streamin	streaming was enabled at the time of the call and could not be restarted because of a device or software error
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.38 void DepthSense::DepthNode::setDepthMap3Planes (bool *value*)

Sets the value of the DepthNode::depthMap3Planes property.

The DepthNode::depthMap3Planes property specifies whether the `depthMap` buffer is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getDepthMap3Planes(), depthMap3PlanesIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.39 void DepthSense::DepthNode::setDepthMapFloatingPoint3Planes (bool *value*)

Sets the value of the DepthNode::depthMapFloatingPoint3Planes property.

The DepthNode::depthMapFloatingPoint3Planes property specifies whether the `depthMapFloatingPoint` is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getDepthMapFloatingPoint3Planes(), depthMapFloatingPoint3PlanesIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.40 void DepthSense::DepthNode::setEnableAccelerometer (bool *value*)

Sets the value of the DepthNode::enableAccelerometer property.

The DepthNode::enableAccelerometer property specifies whether to capture the accelerometer data and make it available through the `acceleration` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableAccelerometer(), enableAccelerometerIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.41 void DepthSense::DepthNode::setEnableConfidenceMap (bool *value*)

Sets the value of the DepthNode::enableConfidenceMap property.

The DepthNode::enableConfidenceMap property specifies whether to enable confidence map computation and make it available through the `confidenceMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableConfidenceMap(), enableConfidenceMapIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.42 void DepthSense::DepthNode::setEnableDenoising (bool *value*)

Sets the value of the DepthNode::enableDenoising property.

The DepthNode::enableDenoising property specifies whether to enable denoising.

Note: The denoising filter will be applied only on the phase map and not on the other maps.

The DepthNode::enableDenoising property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableDenoising(), enableDenoisingIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.43 void DepthSense::DepthNode::setEnableDepthMap (bool *value*)

Sets the value of the DepthNode::enableDepthMap property.

The DepthNode::enableDepthMap property specifies whether to enable fixed point depth map computation and make it available through the `depthMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableDepthMap(), enableDepthMapIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.44 void DepthSense::DepthNode::setEnableDepthMapFloatingPoint (bool *value*)

Sets the value of the DepthNode::enableDepthMapFloatingPoint property.

The DepthNode::enableDepthMapFloatingPoint property specifies whether to enable floating point depth map computation and make it available through the `depthMap-FloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableDepthMapFloatingPoint(), enableDepthMapFloatingPointIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.45 void DepthSense::DepthNode::setEnablePhaseMap (bool *value*)

Sets the value of the DepthNode::enablePhaseMap property.

The DepthNode::enablePhaseMap property specifies whether to enable phase map computation and make it available through the `phaseMap` argument of the `new-SampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnablePhaseMap(), enablePhaseMapIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.46 void DepthSense::DepthNode::setEnableUvMap (bool *value*)

Sets the value of the DepthNode::enableUvMap property.

The DepthNode::enableUvMap property specifies whether to enable UV map computation and make it available through the `uvMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableUvMap(), enableUvMapIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.47 void DepthSense::DepthNode::setEnableVertices (bool *value*)

Sets the value of the DepthNode::enableVertices property.

The DepthNode::enableVertices property specifies whether to enable fixed point vertices computation and make it available through the `vertices` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableVertices(), enableVerticesIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.48 void DepthSense::DepthNode::setEnableVerticesFloatingPoint (bool *value*)

Sets the value of the DepthNode::enableVerticesFloatingPoint property.

The DepthNode::enableVertices property specifies whether to enable floating point vertices computation and make it available through the `verticesFloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getEnableVerticesFloatingPoint(), enableVerticesFloatingPointIsReadOnly()

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.49 void DepthSense::DepthNode::setIlluminationLevel (int32_t *value*)

Sets the value of the DepthNode::illuminationLevel property.

The DepthNode::illuminationLevel property specifies the illumination level to be used by the camera. This property is currently not available for DS325 cameras.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

Parameters

value	the value to set

See also

getIlluminationLevel(), illuminationLevelIsReadOnly()

Exceptions

DepthSense::Unautho	the parent context does not have control of the current node
DepthSense::Argumer	the value passed is out of range
DepthSense::NotSupp	the property is not supported by the node
DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.20.3.50 static DepthSense::Type DepthSense::DepthNode::type ( ) [static]

Returns the DepthSense::DepthNode type object

Returns

the DepthSense::DepthNode type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented from DepthSense::Node.

6.20.4 Property Documentation

6.20.4.1 int32_t DepthSense::DepthNode::confidenceThreshold `[read, write, assign]`

The DepthNode::confidenceThreshold property specifies the confidence threshold.

The DepthNode::confidenceThreshold property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.20.4.2 DepthSense::DepthNode::Configuration DepthSense::DepthNode::configuration `[read, write, assign]`

The DepthNode::configuration property specifies the configuration of the depth node.

Exceptions

DepthSense::InvalidOp	when video synchronization is enabled, the configurations of the depth and color nodes are incompatible or the operation cannot be performed on this node

6.20.4.3 std::vector< DepthSense::DepthNode::Configuration > DepthSense::DepthNode::configurations `[read, assign]`

The DepthNode::configurations property specifies the list of supported node configurations.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.20.4.4 bool DepthSense::DepthNode::depthMap3Planes `[read, write, assign]`

The DepthNode::depthMap3Planes property specifies whether the `depthMap` buffer is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.5 bool DepthSense::DepthNode::depthMapFloatingPoint3Planes `[read, write, assign]`

The DepthNode::depthMapFloatingPoint3Planes property specifies whether the `depthMapFloatingPoint` is the XYZ coordinates as a planar representation (true) or only the Z coordinate (false).

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.6 bool DepthSense::DepthNode::enableAccelerometer `[read, write, assign]`

The DepthNode::enableAccelerometer property specifies whether to capture the accelerometer data and make it available through the `acceleration` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.7 bool DepthSense::DepthNode::enableConfidenceMap `[read, write, assign]`

The DepthNode::enableConfidenceMap property specifies whether to enable confidence map computation and make it available through the `confidenceMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.8 bool DepthSense::DepthNode::enableDenoising `[read, write, assign]`

The DepthNode::enableDenoising property specifies whether to enable denoising.

Note: The denoising filter will be applied only on the phase map and not on the other maps.

The DepthNode::enableDenoising property is deprecated and will be removed in future builds.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.9 bool DepthSense::DepthNode::enableDepthMap `[read, write, assign]`

The DepthNode::enableDepthMap property specifies whether to enable fixed point depth map computation and make it available through the `depthMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.10 bool DepthSense::DepthNode::enableDepthMapFloatingPoint `[read, write, assign]`

The DepthNode::enableDepthMapFloatingPoint property specifies whether to enable floating point depth map computation and make it available through the `depthMapFloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.11 bool DepthSense::DepthNode::enablePhaseMap `[read, write, assign]`

The DepthNode::enablePhaseMap property specifies whether to enable phase map computation and make it available through the `phaseMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.12 bool DepthSense::DepthNode::enableUvMap `[read, write, assign]`

The DepthNode::enableUvMap property specifies whether to enable UV map computation and make it available through the `uvMap` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.13 bool DepthSense::DepthNode::enableVertices `[read, write, assign]`

The DepthNode::enableVertices property specifies whether to enable fixed point vertices computation and make it available through the `vertices` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.20.4.14 bool DepthSense::DepthNode::enableVerticesFloatingPoint `[read, write, assign]`

The DepthNode::enableVertices property specifies whether to enable floating point vertices computation and make it available through the `verticesFloatingPoint` argument of the `newSampleReceived` event.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.20.4.15 int32_t DepthSense::DepthNode::illuminationLevel `[read, write, assign]`

The DepthNode::illuminationLevel property specifies the illumination level to be used by the camera. This property is currently not available for DS325 cameras.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.20.4.16 float DepthSense::DepthNode::range `[read, assign]`

The DepthNode::range property specifies the range of the depth sensor, expressed in meters.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.21 DepthSense::DepthNode::Acceleration Struct Reference

The acceleration returned by the camera.

Public Member Functions

- Acceleration (float x, float y, float z)

Constructs a Acceleration instance.

- bool operator!= (const Acceleration &other) const

  Compares two Acceleration instances for inequality.

- bool operator== (const Acceleration &other) const

  Compares two Acceleration instances for equality.

Public Attributes

- float x

  the x acceleration

- float y

  the y acceleration

- float z

  the z acceleration

6.21.1 Detailed Description

The Acceleration struct holds the 3 axis acceleration expressed in g (9.81 m/s²) units. Coordinates are expressed in the same coordinates as the vertices.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 DepthSense::DepthNode::Acceleration::Acceleration (float *x*, float *y*, float *z*)

Constructs a Acceleration instance, initializing the instance fields with the provided values.

Parameters

x	the value of the Acceleration::x field
y	the value of the Acceleration::y field
z	the value of the Acceleration::z field

6.21.3 Member Function Documentation

6.21.3.1 bool DepthSense::DepthNode::Acceleration::operator!= (const Acceleration & *other*) const

Checks whether the current Acceleration instance is different from the Acceleration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.21.3.2 bool DepthSense::DepthNode::Acceleration::operator== (const Acceleration & *other*) const

Checks whether the current Acceleration instance is equal to the Acceleration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.22 DepthSense::DepthNode::Configuration Struct Reference

The configuration of a depth node.

Public Member Functions

- Configuration (DepthSense::FrameFormat frameFormat, int32_t framerate, DepthSense::DepthNode::CameraMode mode, bool saturation)

  Constructs a Configuration instance.
- bool operator!= (const Configuration &other) const

  Compares two Configuration instances for inequality.
- bool operator== (const Configuration &other) const

  Compares two Configuration instances for equality.

Public Attributes

- DepthSense::FrameFormat frameFormat

  the frame format and resolution
- int32_t framerate

  the frame rate in frames per second
- DepthSense::DepthNode::CameraMode mode

  the mode
- bool saturation

  whether the saturation is enabled or not

6.22.1 Detailed Description

The Configuration struct holds the configuration of a depth node.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 DepthSense::DepthNode::Configuration::Configuration (DepthSense::FrameFormat *frameFormat,* int32_t *framerate,* DepthSense::DepthNode::CameraMode *mode,* bool *saturation*)

Constructs a Configuration instance, initializing the instance fields with the provided values.

Parameters

frameFormat	the value of the Configuration::frameFormat field
framerate	the value of the Configuration::framerate field
mode	the value of the Configuration::mode field
saturation	the value of the Configuration::saturation field

6.22.3 Member Function Documentation

6.22.3.1 bool DepthSense::DepthNode::Configuration::operator!= (const Configuration & *other*) const

Checks whether the current Configuration instance is different from the Configuration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.22.3.2 bool DepthSense::DepthNode::Configuration::operator== (const Configuration & *other*) const

Checks whether the current Configuration instance is equal to the Configuration instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.23 DepthSense::DepthNode::NewSampleReceivedData Struct Reference

Holds the DepthSense::DepthNode::NewSampleReceivedEvent arguments.

Public Attributes

- DepthSense::DepthNode::Acceleration acceleration

  The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
- DepthSense::DepthNode::Configuration captureConfiguration

  the camera configuration that was in effect at the time of capture
- ::DepthSense::Pointer< int16_t > confidenceMap

  the confidence map
- int32_t cumulativeDroppedSampleCount

  the number of dropped samples since the streaming was started
- ::DepthSense::Pointer< int16_t > depthMap

  The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value `32002`.

- ::DepthSense::Pointer< float > depthMapFloatingPoint

  The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
- int32_t droppedSampleCount

  the number of dropped samples since the last `newSampleReceived` event was raised
- ::DepthSense::Pointer< int16_t > phaseMap

  The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value `-32767`.
- DepthSense::StereoCameraParameters stereoCameraParameters

  the system model parameters that were in effect at the time of capture
- uint64_t timeOfArrival

  the time of arrival of the sample in the library, expressed in µs
- uint64_t timeOfCapture

  the time of capture of the sample, expressed in µs
- ::DepthSense::Pointer < DepthSense::UV > uvMap

  The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value `-FLT_MAX`.
- ::DepthSense::Pointer < DepthSense::Vertex > vertices

  The vertices in fixed point format. This map represents the cartesian 3D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value `32002`.
- ::DepthSense::Pointer < DepthSense::FPVertex > verticesFloatingPoint

  The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.

6.23.1 Detailed Description

The NewSampleReceivedData struct holds the DepthSense::DepthNode::NewSampleReceivedEvent parameters and is passed to callbacks connected to that event.

6.24 DepthSense::DepthNode::NewSampleReceivedEvent Class - Reference

Event raised when a depth sample is captured.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data))

  Connects a function to the current event.

- void connect (void(∗handlerFunc)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Connects a function to the current event.

- template<class T >
  void connect (void(∗closure)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceive
  data, T closureData), T closureData)

  Connects a closure to the current event.

- template<class T >
  void connect (void(∗closure)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData), T closureData)

  Connects a closure to the current event.

- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampl
  data))

  Connects a method to the current event.

- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Connects a method to the current event.

- void disconnect (void(∗handlerFunc)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSample
  data))

  Disconnects a function from the current event.

- void disconnect (void(∗handlerFunc)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Disconnects a function from the current event.
- template< class T >
  void disconnect (void(∗closure)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template< class T >
  void disconnect (void(∗closure)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData), T closureData)

  Disconnects a closure from the current event.
- template< class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data))

  Disconnects a method from the current event.
- template< class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount))

  Disconnects a method from the current event.

6.24.1 Detailed Description

The `newSampleReceived` event is raised when a depth sample is captured.

Parameters

confidence-Map	the confidence map
phaseMap	The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value `-32767`.
depthMap	The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value `32002`.
depthMap-Floating-Point	The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
vertices	The vertices in fixed point format. This map represents the cartesian 3-D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value `32002`.
vertices-Floating-Point	The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
uvMap	The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value `-FLT_MAX`.
acceleration	The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
stereo-Camera-Parameters	the system model parameters that were in effect at the time of capture
capture-Configuration	the camera configuration that was in effect at the time of capture
timeOf-Capture	the time of capture of the sample, expressed in µs
timeOf-Arrival	the time of arrival of the sample in the library, expressed in µs
dropped-Sample-Count	the number of dropped samples since the last `newSample-Received` event was raised
cumulative-Dropped-Sample-Count	the number of dropped samples since the streaming was started

6.24.2 Member Function Documentation

6.24.2.1 void DepthSense::DepthNode::NewSampleReceivedEvent::connect (void(*)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.2 void DepthSense::DepthNode::NewSampleReceivedEvent::connect (void(*)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`confidenceMap`	the confidence map
`phaseMap`	The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value `-32767`.
`depthMap`	The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value `32002`.
`depthMapFloatingPoint`	The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`vertices`	The vertices in fixed point format. This map represents the cartesian 3D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value `32002`.
`verticesFloatingPoint`	The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`uvMap`	The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value `-FLT_MAX`.
`acceleration`	The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
`stereoCameraParameters`	the system model parameters that were in effect at the time of capture
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.3 template<class T > void DepthSense::DepthNode::NewSampleReceivedEvent::connect (void(∗)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data, T closureData) *closure,* T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.4 template< class T > void DepthSense::DepthNode::NewSampleReceivedEvent::connect (void(∗)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData) *closure,* T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`confidenceMap`	the confidence map
`phaseMap`	The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value `-32767`.
`depthMap`	The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value `32002`.
`depthMapFloatingPoint`	The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`vertices`	The vertices in fixed point format. This map represents the cartesian 3D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value `32002`.
`verticesFloatingPoint`	The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`uvMap`	The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value `-FLT_MAX`.
`acceleration`	The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
`stereoCameraParameters`	the system model parameters that were in effect at the time of capture
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.5 template<class T > void DepthSense::DepthNode::NewSampleReceivedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.6 template< class T > void DepthSense::DepthNode::NewSampleReceivedEvent::connect (T * *obj*, void(T::*)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`confidenceMap`	the confidence map
`phaseMap`	The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value `-32767`.
`depthMap`	The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value `32002`.
`depthMapFloatingPoint`	The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`vertices`	The vertices in fixed point format. This map represents the cartesian 3D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value `32002`.
`verticesFloatingPoint`	The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`uvMap`	The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value `-FLT_MAX`.
`acceleration`	The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
`stereoCameraParameters`	the system model parameters that were in effect at the time of capture
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.7 void DepthSense::DepthNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.8 void DepthSense::DepthNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

obj	the object for which the event was raised
confidenceMap	the confidence map
phaseMap	The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value -32767.
depthMap	The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value 32002.
depthMapFloatingPoint	The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value -2.0.
vertices	The vertices in fixed point format. This map represents the cartesian 3D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value 32002.
verticesFloatingPoint	The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value -2.0.
uvMap	The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value -FLT_MAX.
acceleration	The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
stereoCameraParameters	the system model parameters that were in effect at the time of capture
captureConfiguration	the camera configuration that was in effect at the time of capture
timeOfCapture	the time of capture of the sample, expressed in µs
timeOfArrival	the time of arrival of the sample in the library, expressed in µs
droppedSampleCount	the number of dropped samples since the last newSampleReceived event was raised
cumulativeDroppedSample-Count	the number of dropped samples since the streaming was started

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumen	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.9 template<class T > void DepthSense::DepthNode::NewSampleReceivedEvent::disconnect (void(∗)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data, T closureData) *closure*, T *closureData*) `[inline]`

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumen	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.10 template< class T > void DepthSense::DepthNode::NewSampleReceivedEvent::disconnect (void(*)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`confidenceMap`	the confidence map
`phaseMap`	The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value `-32767`.
`depthMap`	The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value `32002`.
`depthMapFloatingPoint`	The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`vertices`	The vertices in fixed point format. This map represents the cartesian 3D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value `32002`.
`verticesFloatingPoint`	The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`uvMap`	The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value `-FLT_MAX`.
`acceleration`	The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
`stereoCameraParameters`	the system model parameters that were in effect at the time of capture
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.11 template<class T > void DepthSense::DepthNode::NewSampleReceivedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::DepthNode obj, DepthSense::DepthNode::NewSampleReceivedData data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.24.2.12 template<class T > void DepthSense::DepthNode::NewSampleReceivedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::DepthNode obj,::DepthSense::Pointer< int16_t > confidenceMap,::DepthSense::Pointer< int16_t > phaseMap,::DepthSense::Pointer< int16_t > depthMap,::DepthSense::Pointer< float > depthMapFloatingPoint,::DepthSense::Pointer< DepthSense::Vertex > vertices,::DepthSense::Pointer< DepthSense::FPVertex > verticesFloatingPoint,::DepthSense::Pointer< DepthSense::UV > uvMap, DepthSense::DepthNode::Acceleration acceleration, DepthSense::StereoCameraParameters stereoCameraParameters, DepthSense::DepthNode::Configuration captureConfiguration, uint64_t timeOfCapture, uint64_t timeOfArrival, int32_t droppedSampleCount, int32_t cumulativeDroppedSampleCount) *method*) [inline]

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`confidenceMap`	the confidence map
`phaseMap`	The phase map. This map represents the radial phase ([0 - 2PI[) with respect to the center of the depth camera. Valid values lie in the range [0 - 32767]. Saturated pixels are given the special value `-32767`.
`depthMap`	The depth map in fixed point format. This map represents the cartesian depth of each pixel, expressed in millimeters. Valid values lies in the range [0 - 31999]. Saturated pixels are given the special value `32002`.
`depthMapFloatingPoint`	The depth map in floating point format. This map represents the cartesian depth of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`vertices`	The vertices in fixed point format. This map represents the cartesian 3D coordinates of each pixel, expressed in millimeters. Saturated pixels are given the special value `32002`.
`verticesFloatingPoint`	The vertices in floating point format. This map represents the cartesian 3D coordinates of each pixel, expressed in meters. Saturated pixels are given the special value `-2.0`.
`uvMap`	The UV mapping. This map represents the normalized coordinates of each pixel in the color map. Invalid pixels are given the special value `-FLT_MAX`.
`acceleration`	The acceleration of the camera when the frame was captured. The sampling frequency of this value is 1 Hz.
`stereoCameraParameters`	the system model parameters that were in effect at the time of capture
`captureConfiguration`	the camera configuration that was in effect at the time of capture
`timeOfCapture`	the time of capture of the sample, expressed in µs
`timeOfArrival`	the time of arrival of the sample in the library, expressed in µs
`droppedSampleCount`	the number of dropped samples since the last `newSampleReceived` event was raised
`cumulativeDroppedSample-Count`	the number of dropped samples since the streaming was started

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

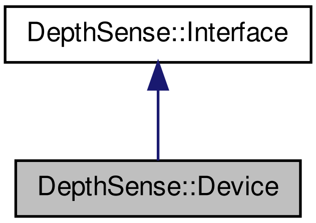
Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.25 DepthSense::Device Class Reference

Represents a camera device.

Inheritance diagram for DepthSense::Device:



Classes

- struct NodeAddedData

  Holds the DepthSense::Device::NodeAddedEvent arguments.
- class NodeAddedEvent

  Event raised when a node is attached to the current device.
- struct NodeRemovedData

  Holds the DepthSense::Device::NodeRemovedEvent arguments.

- class NodeRemovedEvent

    Event raised when a node is detached from the current device.

Public Types

- enum Capabilities { CAPABILITIES_COLOR = 1, CAPABILITIES_DEPTH = 2, CAPABILITIES_AUDIO = 4, CAPABILITIES_ACCELEROMETER = 8 }
- enum Model { MODEL_UNKNOWN = 0, MODEL_DS320 = 1, MODEL_DS325 = 2, MODEL_DS311 = 3, MODEL_GENERIC = 4, MODEL_VF0780 = 5 }

    The camera model.

Public Member Functions

- DepthSense::Device::Capabilities getCapabilities ()

    Gets the value of the Device::capabilities property.
- DepthSense::Device::Model getModel ()

    Gets the value of the Device::model property.
- std::vector< DepthSense::Node > getNodes ()

    Gets the value of the Device::nodes property.
- std::string getSerialNumber ()

    Gets the value of the Device::serialNumber property.
- DepthSense::StereoCameraParameters getStereoCameraParameters ()

    Gets the value of the Device::stereoCameraParameters property.
- int32_t getTofControllerVersion ()

    Gets the value of the Device::tofControllerVersion property.
- int32_t getUsbBackendVersion ()

    Gets the value of the Device::usbBackendVersion property.
- DepthSense::Device::NodeAddedEvent & nodeAddedEvent () const

    Returns the `nodeAdded` event object.
- DepthSense::Device::NodeRemovedEvent & nodeRemovedEvent () const

    Returns the `nodeRemoved` event object.

Static Public Member Functions

- static std::string Capabilities_toString (Capabilities value)

    Converts a DepthSense::Device::Capabilities value to a string.
- static std::string Model_toString (Model value)

    Converts a DepthSense::Device::Model value to a string.
- static DepthSense::Type type ()

    Returns the DepthSense::Device type object.

Properties

- DepthSense::Device::Capabilities capabilities

  The camera capabilities.
- DepthSense::Device::Model model

  The camera model.
- std::vector< DepthSense::Node > nodes

  The stream data sources.
- std::string serialNumber

  The camera serial number.
- DepthSense::StereoCameraParameters stereoCameraParameters

  The system model parameters.
- int32_t tofControllerVersion

  The TOF controller firmware version.
- int32_t usbBackendVersion

  The USB backend firmware version.

6.25.1 Detailed Description

The Device class represents a physical camera device connected to the host. It exposes device information (serial number, model and so on) and contains a number of stream data sources designated as *nodes*.

6.25.2 Member Enumeration Documentation

6.25.2.1 enum DepthSense::Device::Capabilities

A bitmask of capabilities supported by the camera.

Enumerator:

CAPABILITIES_COLOR	the camera supports color streaming
CAPABILITIES_DEPTH	the camera supports depth streaming
CAPABILITIES_AUDIO	the camera supports audio streaming
CAPABILITIES_ACCELEROMETER	the camera has an accelerometer

6.25.2.2 enum DepthSense::Device::Model

An enumeration comprising all the camera models supported by DepthSense SDK.

Enumerator:

MODEL_UNKNOWN	unknown model
MODEL_DS320	DepthSense 320
MODEL_DS325	DepthSense 325
MODEL_DS311	DepthSense 311
MODEL_GENERIC	Generic
MODEL_VF0780	DepthSense 325

6.25.3 Member Function Documentation

6.25.3.1 static std::string DepthSense::Device::Capabilities_toString (Capabilities *value*) [inline, static]

Converts the provided bitmask to a string.

Parameters

value	the bitmask to convert

Returns

a string of the form `"Flag1 | Flag2 | Flag3"`; unknown bits are omitted from the representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.25.3.2 DepthSense::Device::Capabilities DepthSense::Device::getCapabilities ( )

Gets the value of the Device::capabilities property.

Returns

the value of the Device::capabilities property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.25.3.3 DepthSense::Device::Model DepthSense::Device::getModel ( )

Gets the value of the Device::model property.

Returns

the value of the Device::model property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.25.3.4 std::vector< DepthSense::Node > DepthSense::Device::getNodes ( ) [inline]

Gets the value of the Device::nodes property.

Returns

the value of the Device::nodes property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.25.3.5 std::string DepthSense::Device::getSerialNumber ( ) [inline]

Gets the value of the Device::serialNumber property.

Returns

the value of the Device::serialNumber property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.25.3.6 DepthSense::StereoCameraParameters DepthSense::Device::getStereoCameraParameters ( )

Gets the value of the Device::stereoCameraParameters property.

The DepthSense::StereoCameraParameters property specifies the system model parameters.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this device

Returns

the value of the Device::stereoCameraParameters property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.25.3.7 int32_t DepthSense::Device::getTofControllerVersion ( )

Gets the value of the Device::tofControllerVersion property.

The Device::tofControllerVersion property specifies the TOF controller firmware version of the current device. This property is initialized after the node has streamed for the first time.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this device

Returns

the value of the Device::tofControllerVersion property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.25.3.8 int32_t DepthSense::Device::getUsbBackendVersion ( )

Gets the value of the Device::usbBackendVersion property.

The Device::usbBackendVersion property specifies the USB backend firmware version of the current device. This property is initialized after the node has streamed for the first time.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this device

Returns

the value of the Device::usbBackendVersion property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.25.3.9 static std::string DepthSense::Device::Model_toString (Model *value*) `[inline, static]`

Converts the provided enumeration value to a string.

Parameters

value	the enumeration value to convert

Returns

the name of the enumeration member whose value is `value`, or, if `value` is not a member of DepthSense::Device::Model, its numeric representation

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.25.3.10 DepthSense::Device::NodeAddedEvent& DepthSense::Device::nodeAddedEvent ( ) const

Returns a reference to the `nodeAdded` event object, which can be used to connect handlers to that event.

Returns

the `nodeAdded` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.25.3.11 DepthSense::Device::NodeRemovedEvent& DepthSense::Device::nodeRemovedEvent ( ) const

Returns a reference to the `nodeRemoved` event object, which can be used to connect handlers to that event.

Returns

the `nodeRemoved` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.25.3.12 static DepthSense::Type DepthSense::Device::type ( ) `[static]`

Returns the DepthSense::Device type object

Returns

the DepthSense::Device type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented from DepthSense::Interface.

6.25.4 Property Documentation

6.25.4.1 DepthSense::Device::Capabilities DepthSense::Device::capabilities `[read, assign]`

The Device::capabilities property specifies the capabilities of the current device.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this device

6.25.4.2 DepthSense::Device::Model DepthSense::Device::model `[read, assign]`

The Device::model property specifies the model of the current device.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this device

6.25.4.3 std::vector< DepthSense::Node > DepthSense::Device::nodes `[read, assign]`

The Device::nodes property specifies the stream data sources exposed by the current device.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this device

6.25.4.4 std::string DepthSense::Device::serialNumber `[read, assign]`

The Device::serialNumber property specifies the serial number of the current device.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this device

6.25.4.5 DepthSense::StereoCameraParameters DepthSense::Device::stereoCameraParameters `[read, assign]`

The DepthSense::StereoCameraParameters property specifies the system model parameters.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this device

6.25.4.6 int32_t DepthSense::Device::tofControllerVersion `[read, assign]`

The Device::tofControllerVersion property specifies the TOF controller firmware version of the current device. This property is initialized after the node has streamed for the first time.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this device

6.25.4.7 int32_t DepthSense::Device::usbBackendVersion `[read, assign]`

The Device::usbBackendVersion property specifies the USB backend firmware version of the current device. This property is initialized after the node has streamed for the first time.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this device

6.26 DepthSense::Device::NodeAddedData Struct Reference

Holds the DepthSense::Device::NodeAddedEvent arguments.

Public Attributes

- DepthSense::Node node

  the node that was attached to the current device

6.26.1 Detailed Description

The NodeAddedData struct holds the DepthSense::Device::NodeAddedEvent parameters and is passed to callbacks connected to that event.

6.27 DepthSense::Device::NodeAddedEvent Class Reference

Event raised when a node is attached to the current device.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Node node))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data, T closureData), T closureData)

  Connects a closure to the current event.

- template<class T >
  void connect (void(∗closure)(DepthSense::Device obj, DepthSense::Node node, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Node node))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Node node))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Device obj, DepthSense::Node node, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Node node))

  Disconnects a method from the current event.

6.27.1 Detailed Description

The `nodeAdded` event is raised when a stream data source is attached to the current device.

Parameters

node	the node that was attached to the current device

See also

NodeRemovedEvent

6.27.2 Member Function Documentation

6.27.2.1 void DepthSense::Device::NodeAddedEvent::connect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.2 void DepthSense::Device::NodeAddedEvent::connect (void(∗)(DepthSense::Device obj, DepthSense::Node node) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`node`	the node that was attached to the current device

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.3 template<class T > void DepthSense::Device::NodeAddedEvent::connect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data, T closureData) *closure*, T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.4 template<class T > void DepthSense::Device::NodeAddedEvent::connect (void(∗)(DepthSense::Device obj, DepthSense::Node node, T closureData) *closure*, T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`node`	the node that was attached to the current device
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.5 template<class T > void DepthSense::Device::NodeAddedEvent::connect (T * *obj*, void(T::*)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.6 template< class T > void DepthSense::Device::NodeAddedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::Device obj, DepthSense::Node node) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`node`	the node that was attached to the current device

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.7 void DepthSense::Device::NodeAddedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.8 void DepthSense::Device::NodeAddedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Node node) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`node`	the node that was attached to the current device

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.9 template<class T > void DepthSense::Device::NodeAddedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data, T closureData) *closure*, T *closureData*) `[inline]`

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.10 template<class T > void DepthSense::Device::NodeAddedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Node node, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`node`	the node that was attached to the current device
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.11 template<class T > void DepthSense::Device::NodeAddedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::Device obj, DepthSense::Device::NodeAddedData data) *method*) [inline]

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumen	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.27.2.12 template<class T > void DepthSense::Device::NodeAddedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::Device obj, DepthSense::Node node) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`node`	the node that was attached to the current device

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumen	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.28 DepthSense::Device::NodeRemovedData Struct Reference

Holds the DepthSense::Device::NodeRemovedEvent arguments.

Public Attributes

- DepthSense::Node node

  the node that was detached from the current device

6.28.1 Detailed Description

The NodeRemovedData struct holds the DepthSense::Device::NodeRemovedEvent parameters and is passed to callbacks connected to that event.

6.29 DepthSense::Device::NodeRemovedEvent Class Reference

Event raised when a node is detached from the current device.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Node node))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Device obj, DepthSense::Node node, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Node node))

  Connects a method to the current event.

- void disconnect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Device obj, DepthSense::Node node))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Device obj, DepthSense::Node node, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Device::NodeRemovedDa
  data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Device obj, DepthSense::Node node))

  Disconnects a method from the current event.

6.29.1 Detailed Description

The `nodeRemoved` event is raised when a stream data source is detached from the current device.

Parameters

node	the node that was detached from the current device

See also

NodeAddedEvent

6.29.2 Member Function Documentation

6.29.2.1 void DepthSense::Device::NodeRemovedEvent::connect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.2 void DepthSense::Device::NodeRemovedEvent::connect (void(∗)(DepthSense::Device obj, DepthSense::Node node) *handlerFunc*) `[inline]`

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`node`	the node that was detached from the current device

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.3 template<class T > void DepthSense::Device::NodeRemovedEvent::connect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.4 template<class T > void DepthSense::Device::NodeRemovedEvent::connect (void(*)(DepthSense::Device obj, DepthSense::Node node, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`node`	the node that was detached from the current device
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.5 template<class T > void DepthSense::Device::NodeRemovedEvent::connect (T * *obj*, void(T::*)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data) *method*) [inline]

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.6 template<class T > void DepthSense::Device::NodeRemovedEvent::connect (T * *obj*, void(T::*)(DepthSense::Device obj, DepthSense::Node node) *method*) [inline]

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`node`	the node that was detached from the current device

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.7 void DepthSense::Device::NodeRemovedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.8 void DepthSense::Device::NodeRemovedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Node node) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`node`	the node that was detached from the current device

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.9 template<class T > void DepthSense::Device::NodeRemovedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.10 template<class T > void DepthSense::Device::NodeRemovedEvent::disconnect (void(∗)(DepthSense::Device obj, DepthSense::Node node, T closureData) *closure*, T *closureData*) [inline]

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`node`	the node that was detached from the current device
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.11 template<class T > void DepthSense::Device::NodeRemovedEvent::disconnect (T ∗ *obj*, void(T::∗)(DepthSense::Device obj, DepthSense::Device::NodeRemovedData data) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.29.2.12 template<class T > void DepthSense::Device::NodeRemovedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::Device obj, DepthSense::Node node) *method*) `[inline]`

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`node`	the node that was detached from the current device

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

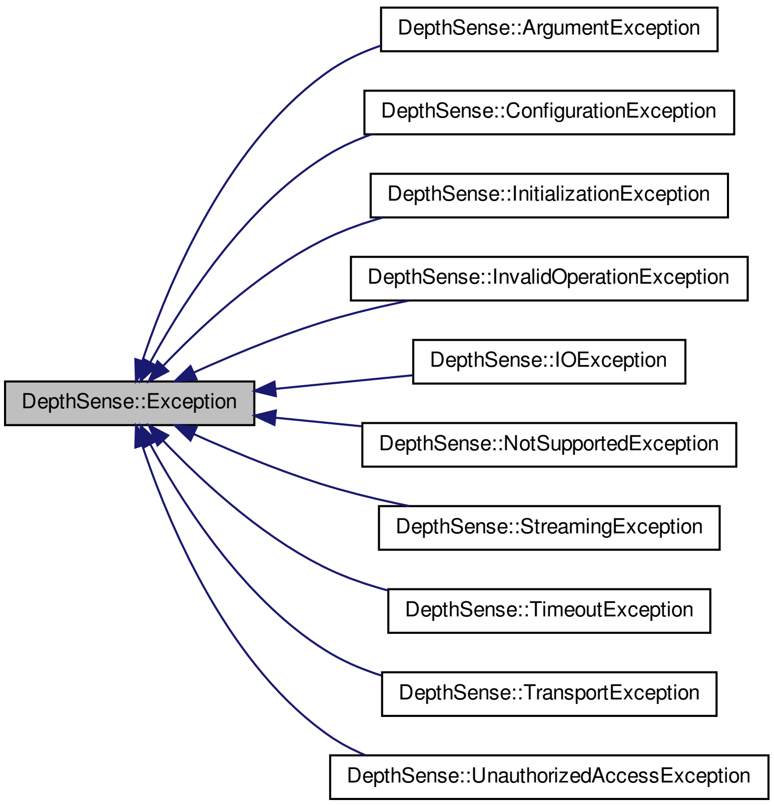
Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.30 DepthSense::Exception Class Reference

The base exception class.

Inheritance diagram for DepthSense::Exception:


DepthSense::ArgumentException
DepthSense::ConfigurationException
DepthSense::InitializationException
DepthSense::InvalidOperationException
DepthSense::IOException
DepthSense::Exception
DepthSense::NotSupportedException
DepthSense::StreamingException
DepthSense::TimeoutException
DepthSense::TransportException
DepthSense::UnauthorizedAccessException

Public Member Functions

- std::string getMessage () const

  Returns the error message.

Protected Member Functions

- **Exception** (void ∗data)

6.30.1 Detailed Description

The Exception class is the common base class for all DepthSense exception types. It is never thrown directly, one of its subclasses is.

DepthSense exceptions feature automatic memory management, implemented with reference counted smart pointer mechanisms.

The Exception class derives from `std::exception`.

6.30.2 Member Function Documentation

6.30.2.1 std::string DepthSense::Exception::getMessage ( ) const

Returns the error message contained in the current exception object.

Returns

the error message

6.31 DepthSense::Extended2DPoint Struct Reference

A point in the cartesian space as defined by its floating point pixel coordinates, and its integral cartesian depth.

Public Member Functions

- Extended2DPoint (DepthSense::Point2D point, int16_t depth)

  Constructs a Extended2DPoint instance.
- bool operator!= (const Extended2DPoint &other) const

  Compares two Extended2DPoint instances for inequality.
- bool operator== (const Extended2DPoint &other) const

  Compares two Extended2DPoint instances for equality.

Public Attributes

- int16_t depth

  the depth at this location, expressed in millimeters
- DepthSense::Point2D point

  the coordinates of the 2D point

6.31.1 Detailed Description

The Extended2DPoint struct holds the position of a point in the cartesian space as defined by its floating point pixel coordinates, and its integral cartesian depth.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 DepthSense::Extended2DPoint::Extended2DPoint (DepthSense::Point2D *point*, int16_t *depth*)

Constructs a Extended2DPoint instance, initializing the instance fields with the provided values.

Parameters

point	the value of the Extended2DPoint::point field
depth	the value of the Extended2DPoint::depth field

6.31.3 Member Function Documentation

6.31.3.1 bool DepthSense::Extended2DPoint::operator!= (const Extended2DPoint & *other*) const

Checks whether the current Extended2DPoint instance is different from the Extended2DPoint instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.31.3.2 bool DepthSense::Extended2DPoint::operator== (const Extended2DPoint & *other*) const

Checks whether the current Extended2DPoint instance is equal to the Extended2DPoint instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.32 DepthSense::ExtrinsicParameters Struct Reference

The extrinsic parameters of the camera system.

Public Member Functions

- ExtrinsicParameters (float r11, float r12, float r13, float r21, float r22, float r23, float r31, float r32, float r33, float t1, float t2, float t3)

  Constructs a ExtrinsicParameters instance.
- bool operator!= (const ExtrinsicParameters &other) const

  Compares two ExtrinsicParameters instances for inequality.
- bool operator== (const ExtrinsicParameters &other) const

  Compares two ExtrinsicParameters instances for equality.

Public Attributes

- float r11

  the r11 parameter
- float r12

  the r12 parameter
- float r13

  the r13 parameter
- float r21

  the r21 parameter
- float r22

  the r22 parameter
- float r23

  the r32 parameter
- float r31

  the r31 parameter
- float r32

  the r32 parameter
- float r33

  the r33 parameter
- float t1

  the t1 parameter
- float t2

the t2 parameter
- float t3

  the t3 parameter

6.32.1 Detailed Description

The ExtrinsicsParameters struct holds the extrinsic parameters of the camera system. Elements are given row by row.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 DepthSense::ExtrinsicParameters::ExtrinsicParameters (float *r11*, float *r12*, float *r13*, float *r21*, float *r22*, float *r23*, float *r31*, float *r32*, float *r33*, float *t1*, float *t2*, float *t3*)

Constructs a ExtrinsicParameters instance, initializing the instance fields with the provided values.

Parameters

r11	the value of the ExtrinsicParameters::r11 field
r12	the value of the ExtrinsicParameters::r12 field
r13	the value of the ExtrinsicParameters::r13 field
r21	the value of the ExtrinsicParameters::r21 field
r22	the value of the ExtrinsicParameters::r22 field
r23	the value of the ExtrinsicParameters::r23 field
r31	the value of the ExtrinsicParameters::r31 field
r32	the value of the ExtrinsicParameters::r32 field
r33	the value of the ExtrinsicParameters::r33 field
t1	the value of the ExtrinsicParameters::t1 field
t2	the value of the ExtrinsicParameters::t2 field
t3	the value of the ExtrinsicParameters::t3 field

6.32.3 Member Function Documentation

6.32.3.1 bool DepthSense::ExtrinsicParameters::operator!= (const ExtrinsicParameters & *other*) const

Checks whether the current ExtrinsicParameters instance is different from the ExtrinsicParameters instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.32.3.2 bool DepthSense::ExtrinsicParameters::operator== (const ExtrinsicParameters & *other*) const

Checks whether the current ExtrinsicParameters instance is equal to the ExtrinsicParameters instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.33 DepthSense::FPExtended2DPoint Struct Reference

A point in the cartesian space as defined by its floating point pixel coordinates, and its floating point cartesian depth.

Public Member Functions

- FPExtended2DPoint (DepthSense::Point2D point, float depth)

  Constructs a FPExtended2DPoint instance.
- bool operator!= (const FPExtended2DPoint &other) const

  Compares two FPExtended2DPoint instances for inequality.
- bool operator== (const FPExtended2DPoint &other) const

  Compares two FPExtended2DPoint instances for equality.

Public Attributes

- float depth

  the depth at this location.
- DepthSense::Point2D point

  the coordinates of the 2D point

6.33.1 Detailed Description

The FPExtended2DPoint struct holds the position of a point in the cartesian space as defined by its floating point pixel coordinates, and its floating point cartesian depth.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 DepthSense::FPExtended2DPoint::FPExtended2DPoint (DepthSense::Point2D *point,* float *depth*)

Constructs a FPExtended2DPoint instance, initializing the instance fields with the provided values.

Parameters

point	the value of the FPExtended2DPoint::point field
depth	the value of the FPExtended2DPoint::depth field

6.33.3 Member Function Documentation

6.33.3.1 bool DepthSense::FPExtended2DPoint::operator!= (const FPExtended2DPoint & *other*) const

Checks whether the current FPExtended2DPoint instance is different from the FPExtended2DPoint instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.33.3.2 bool DepthSense::FPExtended2DPoint::operator== (const FPExtended2DPoint & *other*) const

Checks whether the current FPExtended2DPoint instance is equal to the FPExtended2DPoint instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.34 DepthSense::FPVertex Struct Reference

A point in space as defined by its floating point coordinates.

Public Member Functions

- FPVertex (float x, float y, float z)

  Constructs a FPVertex instance.
- bool operator!= (const FPVertex &other) const

  Compares two FPVertex instances for inequality.
- bool operator== (const FPVertex &other) const

  Compares two FPVertex instances for equality.

Public Attributes

- float x

  the x value
- float y

  the y value
- float z

  the z value

6.34.1 Detailed Description

The FPVertex struct holds the position of a point in space as defined by its 3D floating point coordinates.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 DepthSense::FPVertex::FPVertex (float *x*, float *y*, float *z*)

Constructs a FPVertex instance, initializing the instance fields with the provided values.

Parameters

x	the value of the FPVertex::x field
y	the value of the FPVertex::y field
z	the value of the FPVertex::z field

6.34.3 Member Function Documentation

6.34.3.1 bool DepthSense::FPVertex::operator!= (const FPVertex & *other*) const

Checks whether the current FPVertex instance is different from the FPVertex instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.34.3.2 bool DepthSense::FPVertex::operator== (const FPVertex & *other*) const

Checks whether the current FPVertex instance is equal to the FPVertex instance `other`.

Parameters

other	the instance to compare the current instance with

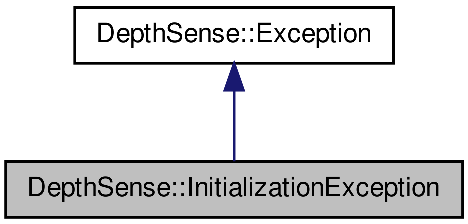
Returns

whether the current instance is equal to instance `other`

6.35 DepthSense::InitializationException Class Reference

The type of the exception thrown when an initialization error has occurred.

Inheritance diagram for DepthSense::InitializationException:



Protected Member Functions

- **InitializationException** (void ∗data)

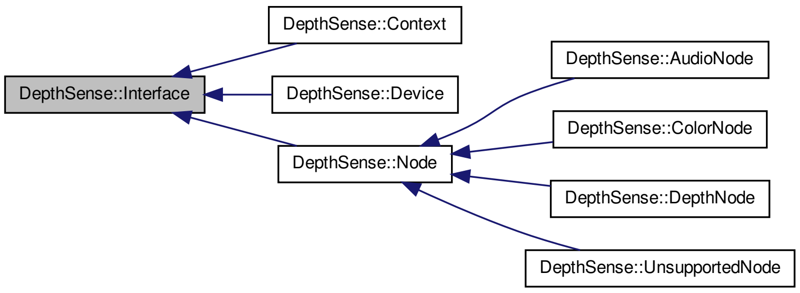
6.35.1 Detailed Description

InitializationException is thrown when a component has failed to initialize properly.

6.36 DepthSense::Interface Class Reference

The base interface class.

Inheritance diagram for DepthSense::Interface:



Classes

- struct PropertyChangedData

  Holds the DepthSense::Interface::PropertyChangedEvent arguments.
- class PropertyChangedEvent

  Event raised when a property has changed.

Public Member Functions

- DepthSense::Context getContext () const

  Returns the parent context.
- DepthSense::Type getType () const

  Returns the runtime type of the current instance.
- bool isSet () const

  Checks if the current instance is set.
- DepthSense::Interface::PropertyChangedEvent & propertyChangedEvent () const

  Returns the `propertyChanged` event object.
- void unset ()

  Unsets the current instance.

Static Public Member Functions

- static DepthSense::Type type ()

  Returns the DepthSense::Interface type object.

6.36.1 Detailed Description

The Interface class is the common base class for all DepthSense interfaces. It provides facilities for introspecting the runtime type of an object and for setting or unsetting the current instance.

The Interface class and its subclasses feature automatic memory management, implemented with reference counted smart pointer mechanisms.

6.36.2 Member Function Documentation

6.36.2.1 DepthSense::Context DepthSense::Interface::getContext ( ) const

Returns the context associated with the current interface.

Returns

the parent context

6.36.2.2 DepthSense::Type DepthSense::Interface::getType ( ) const

Returns the runtime type of the current instance.

Returns

the runtime type of the current instance

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.36.2.3 bool DepthSense::Interface::isSet ( ) const

Checks if the current instance is set.

Given a variable `i` (of type Interface) and a variable `p` (of type `void*`), the `i.isSet()` expression is semantically equivalent to `p != NULL`.

Example:

```
DepthSense::AudioNode audioNode;
bool b = audioNode.isSet(); // b is false
```

Returns

whether the current instance is set

6.36.2.4 DepthSense::Interface::PropertyChangedEvent& DepthSense::Interface::propertyChangedEvent ( ) const

Returns a reference to the `propertyChanged` event object, which can be used to connect handlers to that event.

Returns

the `propertyChanged` event object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.36.2.5 static DepthSense::Type DepthSense::Interface::type( ) [static]

Returns the DepthSense::Interface type object

Returns

the DepthSense::Interface type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented in DepthSense::Context, DepthSense::Device, DepthSense::UnsupportedNode, DepthSense::DepthNode, DepthSense::ColorNode, DepthSense::AudioNode, and DepthSense::Node.

6.36.2.6 void DepthSense::Interface::unset( )

Unsets the current instance.

Given a variable `i` (of type Interface) and a variable `p` (of type `void*`), the `i.unset();` statement is semantically equivalent to `p = NULL;`.

6.37 DepthSense::Interface::PropertyChangedData Struct Reference

Holds the DepthSense::Interface::PropertyChangedEvent arguments.

Public Attributes

- DepthSense::PropertyBase property

  the property whose value has changed

6.37.1 Detailed Description

The PropertyChangedData struct holds the DepthSense::Interface::PropertyChangedEvent parameters and is passed to callbacks connected to that event.

6.38 DepthSense::Interface::PropertyChangedEvent Class - Reference

Event raised when a property has changed.

Public Member Functions

- void connect (void(∗handlerFunc)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data))

  Connects a function to the current event.
- void connect (void(∗handlerFunc)(DepthSense::Interface obj, DepthSense::PropertyBase property))

  Connects a function to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (void(∗closure)(DepthSense::Interface obj, DepthSense::PropertyBase property, T closureData), T closureData)

  Connects a closure to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data))

  Connects a method to the current event.
- template<class T >
  void connect (T ∗obj, void(T::∗method)(DepthSense::Interface obj, DepthSense::PropertyBase property))

  Connects a method to the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data))

  Disconnects a function from the current event.
- void disconnect (void(∗handlerFunc)(DepthSense::Interface obj, DepthSense::PropertyBase property))

  Disconnects a function from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data, T closureData), T closureData)

  Disconnects a closure from the current event.
- template<class T >
  void disconnect (void(∗closure)(DepthSense::Interface obj, DepthSense::PropertyBase property, T closureData), T closureData)

  Disconnects a closure from the current event.

- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Interface obj, DepthSense::Interface::PropertyChan
  data))

  Disconnects a method from the current event.
- template<class T >
  void disconnect (T ∗obj, void(T::∗method)(DepthSense::Interface obj, DepthSense::PropertyBase
  property))

  Disconnects a method from the current event.

6.38.1 Detailed Description

The `propertyChanged` event is raised when the value of a property has changed.

Parameters

property	the property whose value has changed

6.38.2 Member Function Documentation

6.38.2.1 void DepthSense::Interface::PropertyChangedEvent::connect (void(∗)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.2 void DepthSense::Interface::PropertyChangedEvent::connect (void(∗)(DepthSense::Interface obj, DepthSense::PropertyBase property) *handlerFunc*) [inline]

Connects a function to the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`property`	the property whose value has changed

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.3 template<class T > void DepthSense::Interface::PropertyChangedEvent::connect (void(∗)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data, T closureData) *closure*, T *closureData*) [inline]

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.4 template<class T > void DepthSense::Interface::PropertyChangedEvent::connect (void(∗)(DepthSense::Interface obj, DepthSense::PropertyBase property, T closureData) *closure*, T *closureData*) `[inline]`

Connects a closure to the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`property`	the property whose value has changed
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.5 template<class T > void DepthSense::Interface::PropertyChangedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.6 template<class T > void DepthSense::Interface::PropertyChangedEvent::connect (T ∗ *obj*, void(T::∗)(DepthSense::Interface obj, DepthSense::PropertyBase property) *method*) `[inline]`

Connects a method to the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`property`	the property whose value has changed

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is already connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.7 void DepthSense::Interface::PropertyChangedEvent::disconnect (void(∗)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.8 void DepthSense::Interface::PropertyChangedEvent::disconnect (void(∗)(DepthSense::Interface obj, DepthSense::PropertyBase property) *handlerFunc*) `[inline]`

Disconnects a function from the current event. The parameters of the supplied function must be:

`obj`	the object for which the event was raised
`property`	the property whose value has changed

Parameters

handlerFunc	the handler function

Exceptions

DepthSense::Argumer	`handlerFunc` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.9 template<class T > void DepthSense::Interface::PropertyChangedEvent::disconnect (void(∗)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data, T closureData) *closure*, T *closureData*) `[inline]`

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`data`	the event parameters
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.10 template<class T > void DepthSense::Interface::PropertyChangedEvent::disconnect (void(∗)(DepthSense::Interface obj, DepthSense::PropertyBase property, T closureData) *closure*, T *closureData*) `[inline]`

Disconnects a closure from the current event. The parameters of the supplied closure must be:

`obj`	the object for which the event was raised
`property`	the property whose value has changed
`closureData`	the user-supplied lexical environment

Template Parameters

T	the type of the user-supplied lexical environment

Parameters

closure	the closure
closureData	the user-supplied lexical environment

Exceptions

DepthSense::Argumer	the closure identified by `closure` and `closureData` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.11 template<class T > void DepthSense::Interface::PropertyChangedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::Interface obj, DepthSense::Interface::PropertyChangedData data) *method*) [inline]

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`data`	the event parameters

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.38.2.12 template<class T > void DepthSense::Interface::PropertyChangedEvent::disconnect (T * *obj*, void(T::*)(DepthSense::Interface obj, DepthSense::PropertyBase property) *method*) [inline]

Disconnects a method from the current event. The parameters of the supplied method must be:

`obj`	the object for which the event was raised
`property`	the property whose value has changed

Template Parameters

T	the method's parent type

Parameters

obj	the object on which to invoke `method`
method	the method

Exceptions

DepthSense::Argumer	the method handler identified by `obj` and `method` is not connected to the current event
std::bad_alloc	not enough memory to perform the requested operation

6.39 DepthSense::IntrinsicParameters Struct Reference

The intrinsic parameters of the camera system.

Public Member Functions

- IntrinsicParameters (int32_t width, int32_t height, float fx, float fy, float cx, float cy, float k1, float k2, float k3, float p1, float p2)

  Constructs a IntrinsicParameters instance.
- bool operator!= (const IntrinsicParameters &other) const

  Compares two IntrinsicParameters instances for inequality.
- bool operator== (const IntrinsicParameters &other) const

  Compares two IntrinsicParameters instances for equality.

Public Attributes

- float cx

  the central point along the x axis, expressed in pixel units
- float cy

  the central point along the y axis, expressed in pixel units
- float fx

  the focal length along the x axis, expressed in pixel units
- float fy

  the focal length along the y axis, expressed in pixel units
- int32_t height

  the height of the map when the frame was captured
- float k1

  the first radial distortion coefficient
- float k2

  the second radial distortion coefficient

- float k3

  the third radial distortion coefficient
- float p1

  the first tangential distortion coefficient
- float p2

  the second tangential distortion coefficient
- int32_t width

  the width of the map when the frame was captured

6.39.1 Detailed Description

The IntrinsicParameters struct holds the intrinsic parameters of the camera system.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 DepthSense::IntrinsicParameters::IntrinsicParameters (int32_t *width,* int32_t *height,* float *fx,* float *fy,* float *cx,* float *cy,* float *k1,* float *k2,* float *k3,* float *p1,* float *p2*)

Constructs a IntrinsicParameters instance, initializing the instance fields with the provided values.

Parameters

width	the value of the IntrinsicParameters::width field
height	the value of the IntrinsicParameters::height field
fx	the value of the IntrinsicParameters::fx field
fy	the value of the IntrinsicParameters::fy field
cx	the value of the IntrinsicParameters::cx field
cy	the value of the IntrinsicParameters::cy field
k1	the value of the IntrinsicParameters::k1 field
k2	the value of the IntrinsicParameters::k2 field
k3	the value of the IntrinsicParameters::k3 field
p1	the value of the IntrinsicParameters::p1 field
p2	the value of the IntrinsicParameters::p2 field

6.39.3 Member Function Documentation

6.39.3.1 bool DepthSense::IntrinsicParameters::operator!= (const IntrinsicParameters & *other*) const

Checks whether the current IntrinsicParameters instance is different from the IntrinsicParameters instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.39.3.2 bool DepthSense::IntrinsicParameters::operator== (const IntrinsicParameters & *other*) const

Checks whether the current IntrinsicParameters instance is equal to the IntrinsicParameters instance `other`.

Parameters

other	the instance to compare the current instance with

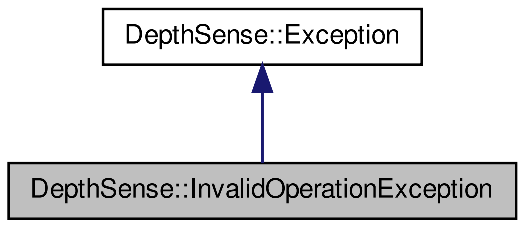
Returns

whether the current instance is equal to instance `other`

6.40 DepthSense::InvalidOperationException Class Reference

The type of the exception thrown when the current state of an object does not support the requested operation.

Inheritance diagram for DepthSense::InvalidOperationException:



Protected Member Functions

- **InvalidOperationException** (void $*$data)

6.40.1 Detailed Description

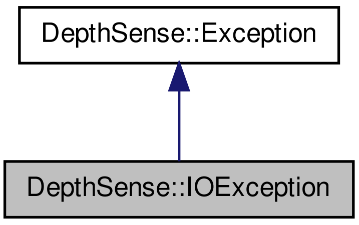
InvalidOperationException is thrown when an operation is not valid because of the current state of an object.

Contrast with NotSupportedException, which is thrown when an operation is not implemented, regardless of any program or library state.

6.41 DepthSense::IOException Class Reference

The type of the exception throw when a device or file I/O operation has failed.

Inheritance diagram for DepthSense::IOException:



Protected Member Functions

- **IOException** (void $*$data)

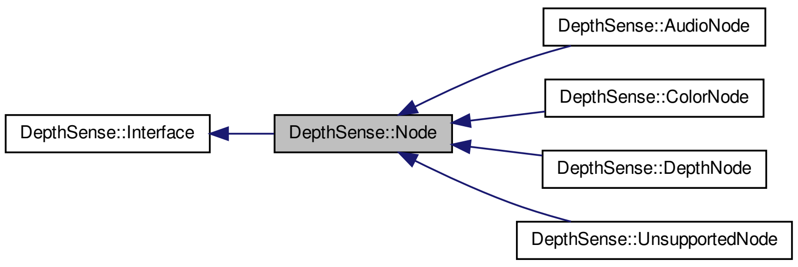
6.41.1 Detailed Description

IOException is thrown when an operation on a device or a file has failed because of an I/O error.

6.42 DepthSense::Node Class Reference

Represents a stream data source.

Inheritance diagram for DepthSense::Node:



Public Member Functions

- int32_t getMediaInterface ()

  Gets the value of the Node::mediaInterface property.
- int32_t getPID ()

  Gets the value of the Node::PID property.
- int32_t getRevision ()

  Gets the value of the Node::revision property.
- std::string getSerialNumber ()

  Gets the value of the Node::serialNumber property.
- int32_t getVID ()

  Gets the value of the Node::VID property.

Static Public Member Functions

- static DepthSense::Type type ()

  Returns the DepthSense::Node type object.

Properties

- int32_t mediaInterface

  The node media interface.
- int32_t PID

  The node product ID.
- int32_t revision

  The node revision.
- std::string serialNumber

  The node serial number.
- int32_t VID

  The node vendor ID.

6.42.1 Detailed Description

The Node class represents a stream data source belonging to a given device. A device may contain several nodes (depth and color sensors, and a microphone array).

6.42.2 Member Function Documentation

6.42.2.1 int32_t DepthSense::Node::getMediaInterface ( )

Gets the value of the Node::mediaInterface property.

Returns

the value of the Node::mediaInterface property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.42.2.2 int32_t DepthSense::Node::getPID ( )

Gets the value of the Node::PID property.

Returns

the value of the Node::PID property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.42.2.3 int32_t DepthSense::Node::getRevision ( )

Gets the value of the Node::revision property.

Returns

the value of the Node::revision property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.42.2.4 std::string DepthSense::Node::getSerialNumber ( ) [inline]

Gets the value of the Node::serialNumber property.

Returns

the value of the Node::serialNumber property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.42.2.5 int32_t DepthSense::Node::getVID ( )

Gets the value of the Node::VID property.

Returns

the value of the Node::VID property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.42.2.6 static DepthSense::Type DepthSense::Node::type ( ) [static]

Returns the DepthSense::Node type object

Returns

the DepthSense::Node type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented from DepthSense::Interface.

Reimplemented in DepthSense::UnsupportedNode, DepthSense::DepthNode, DepthSense::ColorNode, and DepthSense::AudioNode.

6.42.3 Property Documentation

6.42.3.1 int32_t DepthSense::Node::mediaInterface `[read, assign]`

The Node::mediaInterface property specifies the media interface of the node.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.42.3.2 int32_t DepthSense::Node::PID `[read, assign]`

The Node::PID property specifies the product ID of the node.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.42.3.3 int32_t DepthSense::Node::revision `[read, assign]`

The Node::revision property specifies the revision of the node.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.42.3.4 std::string DepthSense::Node::serialNumber `[read, assign]`

The Node::serialNumber property specifies the serial number of the node.

Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.42.3.5 int32_t DepthSense::Node::VID `[read, assign]`

The Node::VID property specifies the vendor ID of the node.

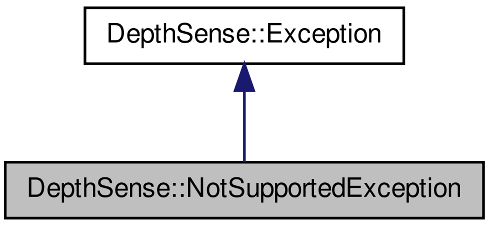
Exceptions

DepthSense::InvalidOp	the operation cannot be performed on this node

6.43 DepthSense::NotSupportedException Class Reference

The type of the exception thrown when a unsupported operation is requested.

Inheritance diagram for DepthSense::NotSupportedException:



Protected Member Functions

- **NotSupportedException** (void ∗data)

6.43.1 Detailed Description

NotSupportedException is thrown when an unimplemented method or property is used.

6.44 DepthSense::Point2D Struct Reference

A point in the cartesian space as defined by its floating point pixel coordinates.

Public Member Functions

- bool operator!= (const Point2D &other) const

  Compares two Point2D instances for inequality.
- bool operator== (const Point2D &other) const

  Compares two Point2D instances for equality.
- Point2D (float x, float y)

  Constructs a Point2D instance.

Public Attributes

- float x

  the x coordinate
- float y

  the y coordinate

6.44.1 Detailed Description

The Point2D struct holds the position of a point in the cartesian space as defined by its floating point pixel coordinates. The origin of the coordinate system is the topleft corner of the image.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 DepthSense::Point2D::Point2D (float *x*, float *y*)

Constructs a Point2D instance, initializing the instance fields with the provided values.

Parameters

x	the value of the Point2D::x field
y	the value of the Point2D::y field

6.44.3 Member Function Documentation

6.44.3.1 bool DepthSense::Point2D::operator!= (const Point2D & *other*) const

Checks whether the current Point2D instance is different from the Point2D instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.44.3.2 bool DepthSense::Point2D::operator== (const Point2D & *other*) const

Checks whether the current Point2D instance is equal to the Point2D instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.45 DepthSense::Pointer$<$ T $>$ Class Template Reference

Exposes a memory buffer.

Public Member Functions

- operator const T $*$ () const

  Returns the internal memory buffer.
- int32_t size () const

  Returns the number of elements.

6.45.1 Detailed Description

template<class T>class DepthSense::Pointer< T >

The Pointer class exposes a read-only memory buffer and its number of elements to the client application.

The Pointer class feature automatic memory management, implemented with reference counted smart pointer mechanisms.

Template Parameters

T	the element type

6.45.2 Member Function Documentation

6.45.2.1 template<class T> DepthSense::Pointer< T >::operator const T * ( ) const

Returns the internal memory buffer. The returned memory is owned by the Pointer instance and should not be modified, deleted or freed.

Returns

the internal memory buffer, or `NULL` if the pointer is unset

Exceptions

DepthSense::IOExcep	could not open the shared memory buffer
std::bad_alloc	not enough memory to perform the requested operation

6.45.2.2 template<class T> int32_t DepthSense::Pointer< T >::size ( ) const

Returns the number of `T` elements contained in the current pointer.

Returns

the number of elements, or `0` if the pointer is unset

6.46 DepthSense::ProcessingHelper Class Reference

Public Member Functions

- void **applyConfidenceThreshold** (const short ∗confidenceMap, const short ∗input, short ∗output)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, const float ∗input, float ∗output)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, const Vertex ∗input, Vertex ∗output)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, const FPVertex ∗input, FPVertex ∗output)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, const UV ∗input, UV ∗output)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, short ∗buffer)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, float ∗buffer)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, Vertex ∗buffer)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, FPVertex ∗buffer)
- void **applyConfidenceThreshold** (const short ∗confidenceMap, UV ∗buffer)
- **ProcessingHelper** (StereoCameraParameters parameters, short confidenceThreshold)
- void setParameters (StereoCameraParameters parameters, short confidenceThreshold)

  Sets the system model parameters.

6.46.1 Member Function Documentation

6.46.1.1 void DepthSense::ProcessingHelper::setParameters (StereoCameraParameters *parameters*, short *confidenceThreshold*)

Sets the system model parameters to use in the computations.

Parameters

parameters	the system model parameters to use (see DepthSense::DepthNode::NewSampleReceivedData::stereoCameraParameters)
confidenceThreshold	the confidence threshold

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47 DepthSense::ProjectionHelper Class Reference

Computes the UV mapping of image points.

Public Member Functions

- void get2DCoordinates (const Vertex ∗input, Point2D ∗output, int32_t npoints, CameraPlane plane)

  Computes the 2D coordinates of a Vertex set.
- void get2DCoordinates (const FPVertex ∗input, Point2D ∗output, int32_t npoints, CameraPlane plane)

  Computes the 2D coordinates of a FPVertex set.
- void get3DCoordinates (const Extended2DPoint ∗input, Vertex ∗output, int32_t npoints)

  Computes the 3D coordinates of an Extended2DPoint set.
- void get3DCoordinates (const FPExtended2DPoint ∗input, FPVertex ∗output, int32_t npoints)

  Computes the 3D coordinates of a FPExtended2DPoint set.
- StereoCameraParameters getStereoCameraParameters () const

  Gets the system model parameters.
- void getUVMapping (const Extended2DPoint ∗input, UV ∗output, int32_t npoints)

  Computes the UV mapping of an Extended2DPoint set.
- void getUVMapping (const FPExtended2DPoint ∗input, UV ∗output, int32_-t npoints)

  Computes the UV mapping of a FPExtended2DPoint set.
- ProjectionHelper (StereoCameraParameters parameters)

  Instantiates a ProjectionHelper.
- void setStereoCameraParameters (StereoCameraParameters parameters)

  Sets the system model parameters.

6.47.1 Detailed Description

The ProjectionHelper class computes the UV mapping of a set of image points.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 DepthSense::ProjectionHelper::ProjectionHelper (StereoCameraParameters *parameters*)

Creates a new instance of the ProjectionHelper class.

Parameters

parameters	the system model parameters to use (see DepthSense::DepthNode::NewSampleReceivedData::stereoCameraParameters)

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3 Member Function Documentation

6.47.3.1 void DepthSense::ProjectionHelper::get2DCoordinates (const Vertex ∗ *input*, Point2D ∗ *output*, int32_t *npoints*, CameraPlane *plane*)

Computes the 2D coordinates of a Vertex set on one camera plane. If a point lies outside of the field of view of the camera, the distortion coefficient is assumed to be the one of the last point within the field of view.

Parameters

input	a Vertex array
output	an user-allocated array of 2D points
npoints	the number of points to be transformed
plane	the plane to project on

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3.2 void DepthSense::ProjectionHelper::get2DCoordinates (const FPVertex ∗ *input*, Point2D ∗ *output*, int32_t *npoints*, CameraPlane *plane*)

Computes the 2D coordinates of a FPVertex set on one camera plane. If a point lies outside of the field of view of the camera, the distortion coefficient is assumed to be the one of the last point within the field of view.

Parameters

input	a FPVertex array
output	an user-allocated array of 2D points
npoints	the number of points to be transformed
plane	the plane to project on

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3.3 void DepthSense::ProjectionHelper::get3DCoordinates (const Extended2DPoint ∗ *input*, Vertex ∗ *output*, int32_t *npoints*)

Computes the 3D coordinates of an Extended2DPoint set. If a point lies outside of the field of view of the camera, the distortion coefficient is assumed to be the one of the last point within the field of view.

The depth of each Extended2DPoint should lie in the range [0 - 31999], otherwise the corresponding 3D output point will be mapped to {0, 0, input[i].z}.

Parameters

input	an Extended2DPoint array
output	an user-allocated array of 3D points
npoints	the number of points to be transformed

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3.4 void DepthSense::ProjectionHelper::get3DCoordinates (const FPExtended2DPoint ∗ *input*, FPVertex ∗ *output*, int32_t *npoints*)

Computes the 3D coordinates of a FPExtended2DPoint set. If a point lies outside of the field of view of the camera, the distortion coefficient is assumed to be the one of the last point within the field of view.

The depth of each FPExtended2DPoint should be greater than 0.0, otherwise the corresponding 3D output point will be mapped to {0.0, 0.0, input[i].z}.

Parameters

input	a FPExtended2DPoint array
output	an user-allocated array of 3D points
npoints	the number of points to be transformed

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3.5 StereoCameraParameters DepthSense::ProjectionHelper::getStereoCameraParameters ( ) const

Gets the system model parameters used in the computations.

Returns

the system model parameters currently in use

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3.6 void DepthSense::ProjectionHelper::getUVMapping (const Extended2DPoint ∗ *input*, UV ∗ *output*, int32_t *npoints*)

Computes the UV mapping of an Extended2DPoint set.

Parameters

input	an Extended2DPoint array
output	a user-allocated array of UV coordinates
npoints	the number of points to be transformed

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3.7 void DepthSense::ProjectionHelper::getUVMapping (const FPExtended2DPoint ∗ *input*, UV ∗ *output*, int32_t *npoints*)

Computes the UV mapping of a FPExtended2DPoint set.

Parameters

input	a FPExtended2DPoint array
output	a user-allocated array of UV coordinates
npoints	the number of points to be transformed

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.47.3.8 void DepthSense::ProjectionHelper::setStereoCameraParameters (StereoCameraParameters *parameters*)

Sets the system model parameters to use in the computations.

Parameters

parameters	the system model parameters to use (see DepthSense::DepthNode::NewSampleReceivedData::stereoCameraParameters)

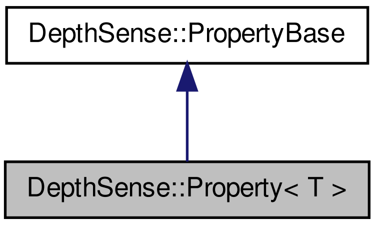
Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.48 DepthSense::Property< T > Class Template Reference

The strongly-typed property leaf class.

Inheritance diagram for DepthSense::Property< T >:



Public Types

- typedef T Type

    The property type.

Public Member Functions

- T getValue (Interface iface) const

  Returns the value of the current property in an interface.
- void setValue (Interface iface, T value) const

  Sets the value of the current property in an interface.

Static Public Member Functions

- static T getValue (Interface iface, const char ∗propertyName)

  Returns the value of a property in an interface.
- static void setValue (Interface iface, const char ∗propertyName, T value)

  Sets the value of a property in an interface.

6.48.1 Detailed Description

template<class T>class DepthSense::Property< T >

The Property class provides strongly-typed runtime property introspection facilities for Interface and its subclasses.

Template Parameters

T	the property type

6.48.2 Member Typedef Documentation

6.48.2.1 template<class T> typedef T DepthSense::Property< T >::Type

Exposes the property type.

6.48.3 Member Function Documentation

6.48.3.1 template<class T> T DepthSense::Property< T >::getValue (Interface *iface*) const

Returns the value of the current property in `iface`.

Parameters

iface	the interface to examine

Returns

the property value

Exceptions

DepthSense::Argumer	`iface` is unset, or the current property does not exist in the type of `iface`
std::bad_alloc	not enough memory to perform the requested operation

Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the getter of the property.

6.48.3.2 template<class T> static T DepthSense::Property< T >::getValue (Interface *iface,* const char ∗ *propertyName*) `[static]`

Returns the value of property named `propertyName` in `iface`.

Parameters

iface	the interface to examine
property-Name	the name of the property to query

Returns

the property value

Exceptions

DepthSense::Argumer	`iface` is unset, or no property named `propertyName` exists in the type of `iface`
DepthSense::InvalidO	the type of the looked up property does not match `T`
std::bad_alloc	not enough memory to perform the requested operation

Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the getter of the property.

6.48.3.3 template<class T> void DepthSense::Property< T >::setValue (Interface *iface*, T *value*) const

Sets the value of the current property in `iface`.

Parameters

iface	the interface to modify
value	the value to set

Exceptions

DepthSense::Argumer	`iface` is unset, or the current property does not exist in the type of `iface`
std::bad_alloc	not enough memory to perform the requested operation

Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the setter of the property.

6.48.3.4 template<class T> static void DepthSense::Property< T >::setValue (Interface *iface*, const char ∗ *propertyName*, T *value*) `[static]`

Sets the value of property named `propertyName` in `iface`.

Parameters

iface	the interface to modify
property-Name	the name of the property whose value to set
value	the value to set

Exceptions

DepthSense::Argumer	`iface` is unset, or no property named `propertyName` exists in the type of `iface`
DepthSense::InvalidOp	the type of the looked up property does not match T
std::bad_alloc	not enough memory to perform the requested operation

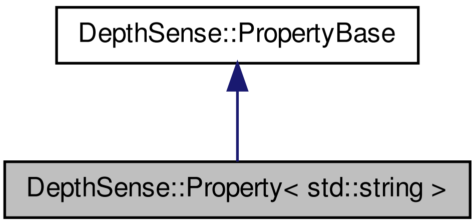
Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the setter of the property.

6.49 DepthSense::Property< std::string > Class Template Reference

The string property class.

Inheritance diagram for DepthSense::Property< std::string >:



Public Types

- typedef std::string Type

  The property type.

Public Member Functions

- std::string getValue (Interface iface) const

  Returns the value of the current property in an interface.
- void setValue (Interface iface, const char ∗value) const

  Sets the value of the current property in an interface.

Static Public Member Functions

- static std::string getValue (Interface iface, const char ∗propertyName)

  Returns the value of a property in an interface.

- static void setValue (Interface iface, const char ∗propertyName, const char ∗value)

  Sets the value of a property in an interface.

6.49.1 Detailed Description

template<>class DepthSense::Property< std::string >

In DepthSense SDK, properties of type `std::string` are particular in so that they are asymmetric: while the getter returns a `std::string` value, the setter accepts a `const char∗` argument.

To account for this peculiarity, the Property<std::string> template specialization is defined. The only difference with the non-specialized Property template is that the setValue() methods accept a `value` argument of type `const char∗` rather than of type T.

6.49.2 Member Typedef Documentation

6.49.2.1 typedef std::string DepthSense::Property< std::string >::Type

Exposes the property type.

6.49.3 Member Function Documentation

6.49.3.1 std::string DepthSense::Property< std::string >::getValue (Interface *iface*) const

Returns the value of the current property in `iface`.

Parameters

iface	the interface to examine

Returns

the property value

Exceptions

DepthSense::Argumen	`iface` is unset, or the current property does not exist in the type of `iface`
std::bad_alloc	not enough memory to perform the requested operation

Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the getter of the property.

6.49.3.2 static std::string **DepthSense::Property**< std::string >**::getValue** (**Interface** *iface*, const char ∗ *propertyName*) `[static]`

Returns the value of property named `propertyName` in `iface`.

Parameters

iface	the interface to examine
property-Name	the name of the property to query

Returns

the property value

Exceptions

DepthSense::Argumer	`iface` is unset, or no property named `propertyName` exists in the type of `iface`
DepthSense::InvalidO	the type of the looked up property does not match T
std::bad_alloc	not enough memory to perform the requested operation

Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the getter of the property.

6.49.3.3 void **DepthSense::Property**< std::string >**::setValue** (**Interface** *iface*, const char ∗ *value*) const

Sets the value of the current property in `iface`.

Parameters

iface	the interface to modify
value	the value to set

Exceptions

DepthSense::Argumer	`iface` is unset, or the current property does not exist in the type of `iface`
std::bad_alloc	not enough memory to perform the requested operation

Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the setter of the property.

6.49.3.4 static void DepthSense::Property< std::string >::setValue (Interface *iface*, const char ∗ *propertyName*, const char ∗ *value*) [static]

Sets the value of property named `propertyName` in `iface`.

Parameters

iface	the interface to modify
property-Name	the name of the property whose value to set
value	the value to set

Exceptions

DepthSense::Argumer	`iface` is unset, or no property named `propertyName` exists in the type of `iface`
DepthSense::InvalidO	the type of the looked up property does not match T
std::bad_alloc	not enough memory to perform the requested operation

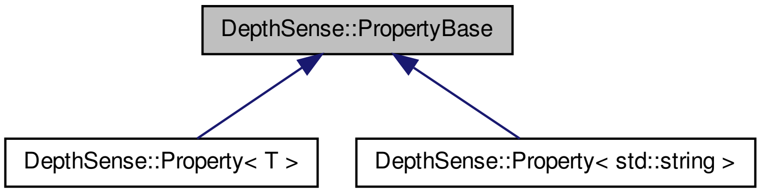
Note

In addition to the exceptions specified above, this method can throw any of the exceptions specified in the setter of the property.

6.50 DepthSense::PropertyBase Class Reference

The base property class.

Inheritance diagram for DepthSense::PropertyBase:


DepthSense::PropertyBase
DepthSense::Property< T >
DepthSense::Property< std::string >

Public Member Functions

- Type declaringType () const

    Returns the declaring type of the current property.
- bool isReadOnly () const

    Returns whether the current property is read-only.
- bool isReadOnly (Interface iface) const

    Returns whether the current property is read-only in the provided interface.
- bool isSet () const

    Checks if the current property instance is set.
- std::string name () const

    Returns the name of the current property.
- void unset ()

    Unsets the current property instance.

6.50.1 Detailed Description

The PropertyBase class is the common base class for the strongly-typed Property template instantiations. It provides runtime property introspection facilities for Interface and its subclasses.

A PropertyBase object can be obtained by using Type::getProperty(), Type::getProperties() or by connecting to the `propertyChanged` event of the Interface class.

The PropertyBase class and its subclasses feature automatic memory management, implemented with reference counted smart pointer mechanisms.

6.50.2 Member Function Documentation

6.50.2.1 Type DepthSense::PropertyBase::declaringType ( ) const

Returns the type in which the current property was declared. This type might or might not correspond to the derived runtime type of the interface being queried.

Returns

the declaring type

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.50.2.2 bool DepthSense::PropertyBase::isReadOnly ( ) const

Returns whether the current property is read-only. Contrast with isReadOnly(Interface) const, which determines whether the current property is read-only in the provided interface.

Returns

whether the current property is read-only

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.50.2.3 bool DepthSense::PropertyBase::isReadOnly (Interface *iface*) const

Returns whether the current property is read-only in `iface`. Properties which are read-write by design (that is, for which isReadOnly() returns true) can be read-only in a given interface if, for instance, the interface is a Node and the client application did not request full control access of the node, which can be done with Context::requestControl(Node).

Contrast with isReadOnly(), which returns whether the current property is read-only as per its static, compile-time specifications.

Parameters

iface	the interface to examine

Returns

whether the current property is read-only in `iface`

Exceptions

DepthSense::Argumer	`iface` is unset
std::bad_alloc	not enough memory to perform the requested operation

6.50.2.4 bool DepthSense::PropertyBase::isSet ( ) const

Checks if the current property instance is set.

Given a variable `prop` (of type PropertyBase) and a variable `ptr` (of type `void`∗), the `prop.isSet()` expression is semantically equivalent to `ptr != NULL`.

Example:

```
DepthSense::PropertyBase prop;
bool b = prop.isSet(); // b is false
```

Returns

whether the current property instance is set

6.50.2.5 std::string DepthSense::PropertyBase::name ( ) const [inline]

Returns the name of the current property.

Returns

the property name

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.50.2.6 void DepthSense::PropertyBase::unset ( )

Unsets the current property instance.

Given a variable `prop` (of type PropertyBase) and a variable `ptr` (of type `void`∗), the `prop.unset();` statement is semantically equivalent to `p = NULL;`.

6.51 DepthSense::StereoCameraParameters Struct Reference

The intrinsic and extrinsic parameters of the camera system.

Public Member Functions

- bool operator!= (const StereoCameraParameters &other) const

  Compares two StereoCameraParameters instances for inequality.
- bool operator== (const StereoCameraParameters &other) const

  Compares two StereoCameraParameters instances for equality.
- StereoCameraParameters (DepthSense::IntrinsicParameters depthIntrinsics, DepthSense::IntrinsicParameters colorIntrinsics, DepthSense::ExtrinsicParameters extrinsics)

  Constructs a StereoCameraParameters instance.

Public Attributes

- DepthSense::IntrinsicParameters colorIntrinsics

  The intrinsic parameters of the color camera.
- DepthSense::IntrinsicParameters depthIntrinsics

  The intrinsic parameters of the depth camera.
- DepthSense::ExtrinsicParameters extrinsics

  The extrinsic parameters of the system.

6.51.1 Detailed Description

The StereoCameraParameters holds the intrinsic and extrinsic parameters of the camera system.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 DepthSense::StereoCameraParameters::StereoCameraParameters (DepthSense::IntrinsicParameters *depthIntrinsics*, DepthSense::IntrinsicParameters *colorIntrinsics*, DepthSense::ExtrinsicParameters *extrinsics*)

Constructs a StereoCameraParameters instance, initializing the instance fields with the provided values.

Parameters

depth-Intrinsics	the value of the StereoCameraParameters::depthIntrinsics field
color-Intrinsics	the value of the StereoCameraParameters::colorIntrinsics field
extrinsics	the value of the StereoCameraParameters::extrinsics field

6.51.3 Member Function Documentation

6.51.3.1 bool DepthSense::StereoCameraParameters::operator!= (const StereoCameraParameters & *other*) const

Checks whether the current StereoCameraParameters instance is different from the StereoCameraParameters instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.51.3.2 bool DepthSense::StereoCameraParameters::operator== (const StereoCameraParameters & *other*) const

Checks whether the current StereoCameraParameters instance is equal to the StereoCameraParameters instance `other`.

Parameters

other	the instance to compare the current instance with

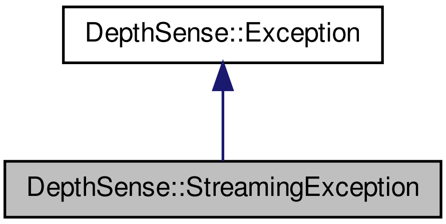
Returns

whether the current instance is equal to instance `other`

6.52 DepthSense::StreamingException Class Reference

The type of the exception thrown when a streaming error has occurred.

Inheritance diagram for DepthSense::StreamingException:



Protected Member Functions

- **StreamingException** (void ∗data)

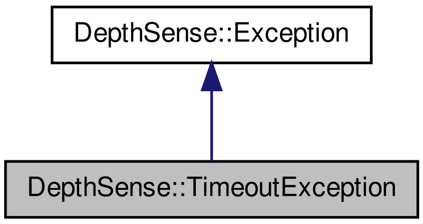
6.52.1 Detailed Description

StreamingException is thrown when a device or software error prevents streaming from starting or from resuming after a configuration change.

6.53 DepthSense::TimeoutException Class Reference

The type of the exception thrown when a timeout condition occurs.

Inheritance diagram for DepthSense::TimeoutException:



Protected Member Functions

- **TimeoutException** (void ∗data)

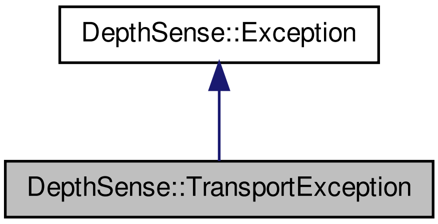
6.53.1 Detailed Description

TimeoutException is thrown when the user-provided timeout of a method accepting a timeout parameter expires.

6.54 DepthSense::TransportException Class Reference

The type of the exception thrown when a network or protocol error has occurred.

Inheritance diagram for DepthSense::TransportException:



Protected Member Functions

- **TransportException** (void ∗data)

6.54.1 Detailed Description

TransportException is thrown when a network I/O operation has failed, or when an invalid protocol packet has been received.

6.55 DepthSense::Type Class Reference

Represents a DepthSense instance type.

Public Member Functions

- std::vector< PropertyBase > getProperties () const

  Gets the properties of the current type and of its type ancestry.
- PropertyBase getProperty (const char ∗name) const

  Gets a property by name.
- std::string name () const

  Returns the qualified name of the current type.

6.55.1 Detailed Description

The Type class can be used to obtain information about the runtime type of a DepthSense::Interface instance.

Example:

```
void displayTypeName (DepthSense::Interface iface)
{
  DepthSense::Type type = iface.getType();
  cout << "Interface is of type " << type.name() << endl;
}
```

The Type class features automatic memory management, implemented with reference counted smart pointer mechanisms.

6.55.2 Member Function Documentation

6.55.2.1 std::vector<PropertyBase> DepthSense::Type::getProperties ( ) const [inline]

Returns the properties of the current type and of its type ancestry, sorted in no particular order.

Returns

the property list

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.55.2.2 PropertyBase DepthSense::Type::getProperty (const char * *name*) const

Returns the property whose name is `name`, or an unset PropertyBase instance if no such property exists.

Name lookup is performed in the current type and in its type ancestry.

Returns

the property named `name`, or an unset instance if not found

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.55.2.3 std::string DepthSense::Type::name ( ) const [inline]

Returns the qualified name of the current type.

Returns

a string of the form `"DepthSense.ColorNode"`

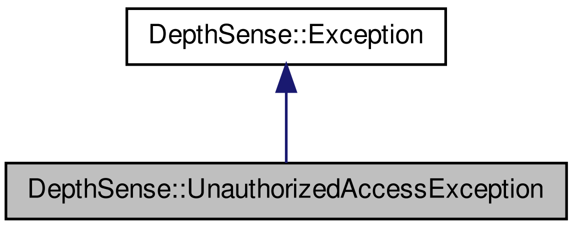
Exceptions

std::bad_alloc	not enough memory to perform the requested operation

6.56 DepthSense::UnauthorizedAccessException Class Reference

The type of the exception thrown when access to a privileged operation is denied.

Inheritance diagram for DepthSense::UnauthorizedAccessException:



Protected Member Functions

- **UnauthorizedAccessException** (void ∗data)

6.56.1 Detailed Description

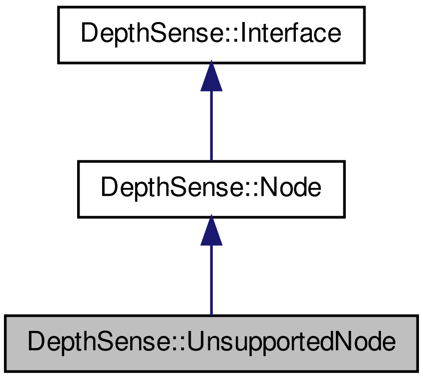
UnauthorizedAccessException is thrown when a privileged method is called or when a privileged property is set.

A privileged method or property is one which requires exclusive control of a device or node. Such exclusive control can be obtained with a call to Context::requestControl().

6.57 DepthSense::UnsupportedNode Class Reference

Represents an unsupported stream data source.

Inheritance diagram for DepthSense::UnsupportedNode:



Public Member Functions

- std::string getReason ()

  Gets the value of the UnsupportedNode::reason property.

Static Public Member Functions

- static DepthSense::Type type ()

  Returns the DepthSense::UnsupportedNode type object.

Properties

- std::string reason

  The reason the node is unsupported.

6.57.1 Detailed Description

The UnsupportedNode class allows to have some information about an unsupported device. This node can not be registered neither controlled.

6.57.2 Member Function Documentation

6.57.2.1 std::string DepthSense::UnsupportedNode::getReason( ) [inline]

Gets the value of the UnsupportedNode::reason property.

Returns

the value of the UnsupportedNode::reason property

Exceptions

DepthSense::Transpor	a network or protocol error has occurred
std::bad_alloc	not enough memory to perform the requested operation

6.57.2.2 static DepthSense::Type DepthSense::UnsupportedNode::type( ) [static]

Returns the DepthSense::UnsupportedNode type object

Returns

the DepthSense::UnsupportedNode type object

Exceptions

std::bad_alloc	not enough memory to perform the requested operation

Reimplemented from DepthSense::Node.

6.57.3 Property Documentation

6.57.3.1 std::string DepthSense::UnsupportedNode::reason [read, assign]

The UnsupportedNode::reason property specifies the reason why the node is not supported.

Exceptions

DepthSense::InvalidO	the operation cannot be performed on this node

6.58 DepthSense::UV Struct Reference

UV coordinates.

Public Member Functions

- bool operator!= (const UV &other) const

  Compares two UV instances for inequality.
- bool operator== (const UV &other) const

  Compares two UV instances for equality.
- UV (float u, float v)

  Constructs a UV instance.

Public Attributes

- float u

  the u value
- float v

  the v value

6.58.1 Detailed Description

The UV struct holds the UV coordinates of a point of a UV map.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 DepthSense::UV::UV (float *u*, float *v*)

Constructs a UV instance, initializing the instance fields with the provided values.

Parameters

u	the value of the UV::u field
v	the value of the UV::v field

6.58.3 Member Function Documentation

6.58.3.1 bool DepthSense::UV::operator!= (const UV & *other*) const

Checks whether the current UV instance is different from the UV instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.58.3.2 bool DepthSense::UV::operator== (const UV & *other*) const

Checks whether the current UV instance is equal to the UV instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.59 DepthSense::Version Struct Reference

DepthSense version information.

Public Member Functions

- bool operator!= (const Version &other) const

  Compares two Version instances for inequality.
- bool operator== (const Version &other) const

  Compares two Version instances for equality.
- Version (int32_t major, int32_t minor, int32_t patch, int32_t build)

  Constructs a Version instance.

Public Attributes

- int32_t build

the package build number

- int32_t major

  the major version number
- int32_t minor

  the minor version number
- int32_t patch

  the patch level

6.59.1 Detailed Description

The Version struct contains version information for the DepthSenseSDK software.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 DepthSense::Version::Version (int32_t *major,* int32_t *minor,* int32_t *patch,* int32_t *build*)

Constructs a Version instance, initializing the instance fields with the provided values.

Parameters

major	the value of the Version::major field
minor	the value of the Version::minor field
patch	the value of the Version::patch field
build	the value of the Version::build field

6.59.3 Member Function Documentation

6.59.3.1 bool DepthSense::Version::operator!= (const Version & *other*) const

Checks whether the current Version instance is different from the Version instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.59.3.2 bool DepthSense::Version::operator== (const Version & *other*) const

Checks whether the current Version instance is equal to the Version instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`

6.60 DepthSense::Vertex Struct Reference

A point in space as defined by its integer coordinates.

Public Member Functions

- bool operator!= (const Vertex &other) const

  Compares two Vertex instances for inequality.
- bool operator== (const Vertex &other) const

  Compares two Vertex instances for equality.
- Vertex (int16_t x, int16_t y, int16_t z)

  Constructs a Vertex instance.

Public Attributes

- int16_t x

  the x value
- int16_t y

  the y value
- int16_t z

  the z value

6.60.1 Detailed Description

The Vertex struct holds the position of a point in space as defined by its 3D integer coordinates.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 DepthSense::Vertex::Vertex (int16_t *x*, int16_t *y*, int16_t *z*)

Constructs a Vertex instance, initializing the instance fields with the provided values.

Parameters

x	the value of the Vertex::x field
y	the value of the Vertex::y field
z	the value of the Vertex::z field

6.60.3 Member Function Documentation

6.60.3.1 bool DepthSense::Vertex::operator!= (const Vertex & *other*) const

Checks whether the current Vertex instance is different from the Vertex instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is different from instance `other`

6.60.3.2 bool DepthSense::Vertex::operator== (const Vertex & *other*) const

Checks whether the current Vertex instance is equal to the Vertex instance `other`.

Parameters

other	the instance to compare the current instance with

Returns

whether the current instance is equal to instance `other`