

# MYNT EYE S SDK

2.2.3

Generated by Doxygen 1.8.14



# Contents

- 1 MYNT EYE S SDK 1**
  
- 2 Device Data Specification 3**
  - 2.1 Device Information ..... 3
  - 2.2 Image Params ..... 3
  - 2.3 IMU Params ..... 4
  - 2.4 Image Data ..... 4
  - 2.5 IMU Data ..... 5
  
- 3 Device Control Specification 7**
  - 3.1 Control Protocols ..... 7
  - 3.2 Control Channels ..... 8
  
- 4 Module Index 11**
  - 4.1 Modules ..... 11
  
- 5 Hierarchical Index 13**
  - 5.1 Class Hierarchy ..... 13
  
- 6 Class Index 15**
  - 6.1 Class List ..... 15

---

<b>7</b>	<b>Module Documentation</b>	<b>17</b>
7.1	Enumerations	17
7.1.1	Detailed Description	18
7.1.2	Enumeration Type Documentation	18
7.1.2.1	AddOns	18
7.1.2.2	Capabilities	18
7.1.2.3	Format	19
7.1.2.4	Info	19
7.1.2.5	Model	19
7.1.2.6	Option	20
7.1.2.7	Source	20
7.1.2.8	Stream	20
7.2	Intrinsics & Extrinsics	22
7.2.1	Detailed Description	22
7.3	Datatypes	23
7.3.1	Detailed Description	23
7.4	Utilities	24
7.4.1	Detailed Description	24
7.4.2	Function Documentation	24
7.4.2.1	get_real_exposure_time()	24
7.4.2.2	select()	24

---

<b>8 Class Documentation</b>	<b>25</b>
8.1 mynteye::API Class Reference	25
8.1.1 Detailed Description	27
8.1.2 Member Typedef Documentation	27
8.1.2.1 motion_callback_t	27
8.1.2.2 stream_callback_t	27
8.1.3 Member Function Documentation	27
8.1.3.1 Create() [1/4]	27
8.1.3.2 Create() [2/4]	27
8.1.3.3 Create() [3/4]	28
8.1.3.4 Create() [4/4]	28
8.1.3.5 EnableStreamData()	29
8.1.3.6 GetStreamDatas()	29
8.2 mynteye::AsyncCallback< Data > Class Template Reference	29
8.3 mynteye::Context Class Reference	29
8.3.1 Detailed Description	30
8.3.2 Member Function Documentation	30
8.3.2.1 devices()	30
8.4 mynteye::Device Class Reference	30
8.4.1 Detailed Description	32
8.4.2 Member Typedef Documentation	32
8.4.2.1 motion_callback_t	32
8.4.2.2 stream_callback_t	32
8.4.3 Member Function Documentation	32
8.4.3.1 Create()	32
8.4.3.2 GetStreamDatas()	33
8.5 mynteye::Extrinsics Struct Reference	33
8.5.1 Detailed Description	33
8.5.2 Member Function Documentation	33
8.5.2.1 Inverse()	34

8.6	mynteye::device::Frame Class Reference	34
8.6.1	Detailed Description	34
8.6.2	Member Function Documentation	34
8.6.2.1	clone()	35
8.6.2.2	data() [1/2]	35
8.6.2.3	data() [2/2]	35
8.6.2.4	format()	35
8.6.2.5	height()	35
8.6.2.6	size()	35
8.6.2.7	width()	36
8.7	mynteye::ImgData Struct Reference	36
8.7.1	Detailed Description	36
8.8	mynteye::ImuData Struct Reference	36
8.8.1	Detailed Description	36
8.8.2	Member Data Documentation	37
8.8.2.1	accel	37
8.8.2.2	gyro	37
8.9	mynteye::ImuIntrinsics Struct Reference	37
8.9.1	Detailed Description	37
8.9.2	Member Data Documentation	37
8.9.2.1	scale	37
8.10	mynteye::Intrinsics Struct Reference	38
8.10.1	Detailed Description	38
8.11	mynteye::device::MotionData Struct Reference	38
8.11.1	Detailed Description	38
8.11.2	Member Data Documentation	38
8.11.2.1	imu	39
8.12	mynteye::api::MotionData Struct Reference	39
8.12.1	Detailed Description	39
8.12.2	Member Data Documentation	39

8.12.2.1 imu . . . . .	39
8.13 mynteye::MotionIntrinsics Struct Reference . . . . .	39
8.13.1 Detailed Description . . . . .	40
8.14 mynteye::Object Struct Reference . . . . .	40
8.14.1 Detailed Description . . . . .	40
8.15 mynteye::ObjMat Struct Reference . . . . .	40
8.15.1 Detailed Description . . . . .	41
8.16 mynteye::ObjMat2 Struct Reference . . . . .	41
8.16.1 Detailed Description . . . . .	41
8.17 mynteye::OptionInfo Struct Reference . . . . .	41
8.17.1 Detailed Description . . . . .	42
8.18 mynteye::Plugin Class Reference . . . . .	42
8.18.1 Detailed Description . . . . .	42
8.18.2 Member Function Documentation . . . . .	42
8.18.2.1 OnCreate() . . . . .	42
8.18.2.2 OnDepthProcess() . . . . .	43
8.18.2.3 OnDisparityNormalizedProcess() . . . . .	43
8.18.2.4 OnDisparityProcess() . . . . .	43
8.18.2.5 OnPointsProcess() . . . . .	44
8.18.2.6 OnRectifyProcess() . . . . .	44
8.19 mynteye::device::StreamData Struct Reference . . . . .	44
8.19.1 Detailed Description . . . . .	45
8.19.2 Member Data Documentation . . . . .	45
8.19.2.1 frame . . . . .	45
8.19.2.2 img . . . . .	45
8.20 mynteye::api::StreamData Struct Reference . . . . .	45
8.20.1 Detailed Description . . . . .	45
8.20.2 Member Data Documentation . . . . .	46
8.20.2.1 frame . . . . .	46
8.20.2.2 frame_raw . . . . .	46
8.20.2.3 img . . . . .	46
8.21 mynteye::StreamRequest Struct Reference . . . . .	46
8.21.1 Detailed Description . . . . .	46
8.22 mynteye::strings_error Class Reference . . . . .	47
8.22.1 Detailed Description . . . . .	47





# Chapter 1

## MYNT EYE S SDK

- [API Classes](#)
- [API Modules](#)
  - [Enumerations](#)
  - [Datatypes](#)
  - [Utilities](#)
  - [Intrinsics & Extrinsics](#)
- [Device Specifications](#)
  - [Device Data Specification](#)
  - [Device Control Specification](#)



## Chapter 2

# Device Data Specification

- [Device Information](#)
- [Image Params](#)
- [IMU Params](#)
- [Image Data](#)
- [IMU Data](#)

### 2.1 Device Information

Name	Field	Fixed Value	USB De- scriptor	UVC Exten- sion Unit	Bytes	Note
VID	vid	0x04B4	✓	×	2	
PID	pid	0x00F9	✓	×	2	
Device name	name	MYNT-EYE-?	✓	✓ Get	16	MYNT-EYE-S1000
Serial number	serial_↔ number	-	✓	✓ Get	16	
Firmware ver- sion	firmware_↔ version	-	✓	✓ Get	2	major,minor
Hardware ver- sion	hardware_↔ version	-	×	✓ Get	3	major,minor,flag
Spec version	spec_version	-	×	✓ Get	2	major,minor
Lens type	lens_type	-	×	✓ Get/Set	4	vendor(2),product(2); default: 0
IMU type	imu_type	-	×	✓ Get/Set	4	vendor(2),product(2); default: 0
Nominal baseline	nominal_↔ baseline	-	×	✓ Get/Set	2	unit: mm; default: 0

### 2.2 Image Params

## Image Intrinsic

Name	Field	Unit	Bytes	Note
Image width	width	px	2	uint16_t; [0,65535]
Image height	height	px	2	uint16_t; [0,65535]
Focal length	fx	-	8	double
	fy	-	8	double
Principal point	cx	-	8	double
	cy	-	8	double
Distortion model	model	-	1	uint8_t; pinhole,...
Distortion coefficients	coeffs[5]	-	40	double; k1,k2,p1,p2,k3

## Image Extrinsic

Transformation matrix from left image to right image.

Name	Field	Unit	Bytes	Note
Rotation matrix	rotation[3][3]	-	72	double
Translation vector	translation[3]	-	24	double

## 2.3 IMU Params

### IMU Intrinsic

Name	Field	Unit	Bytes	Note
Scale matrix	acc_scale[3][3]	-	72	double
	gyro_scale[3][3]	-	72	double
Zero-drift	acc_drift[3]	-	24	double
	gyro_drift[3]	-	24	double
Noise density	acc_noise[3]	-	24	double
	gyro_noise[3]	-	24	double
Random walk	acc_bias[3]	-	24	double
	gyro_bias[3]	-	24	double

### IMU Extrinsic

Transformation matrix from left image to IMU.

Name	Field	Unit	Bytes	Note
Rotation matrix	rotation[3][3]	-	72	double
Translation vector	translation[3]	-	24	double

## 2.4 Image Data

Name	Field	Unit	Bytes	Note
Frame ID	frame_id	-	2	uint16_t; [0,65535]
Timestamp	timestamp	10 us	4	uint32_t
Exposure Time	exposure_time	10 us	2	uint16_t

### Image Packet

Name	Header	Size	Frame ID	Timestamp	Exposure Time	Checksum
Bytes	1	1	2	4	2	1
Type	uint8↔ _t	uint8_t	uint16_t	uint32_t	uint16_t	uint8_t
Description	0x3B	0x08, content size	Frame ID	Timestamp	Exposure time	Checksum, X↔OR of all content bytes

- The image packet will be dropped, if checksum is incorrect.
- The accuracy of the time unit: 0.01 ms / 10 us.
  - The timestamp could indicate 11.9 hours, it will accumulate again after overflow.
- The timestamp accumulation starts from the time of power-on, instead of opening.

## 2.5 IMU Data

### IMU Request Packet

Name	Header	Serial Number
Bytes	1	4
Type	uint8↔ _t	uint32_t
Description	0x5A	First request should be 0, otherwise the last one

### IMU Response Packet

The IMU response packet contains multiple IMU packets, and each IMU packet contains multiple IMU segments.

Name	Header	State	Size	IMU Packets	Checksum
Bytes	1	1	2	...	1
Type	uint8↔ _t	uint8_t	uint16_t	-	uint8_t
Description	0x5B	0 is success, others are failed	Content size	IMU packets	Checksum, XOR of all content bytes

### IMU Packet

The IMU packet is an array of IMU datas.

Name	Serial Number	Timestamp	Count	IMU Datas
Bytes	4	4	1	...
Type	uint32_t	uint32_t	uint8_t	-
Description	Serial number	IMU basic timestamp	The number of IMU datas	IMU datas

### IMU Segment

Name	Offset	Frame ID	Accelerometer	Temperature	Gyroscope
Bytes	2	2	6	2	6
Type	int16_t	uint16_t	int16_t * 3	int16_t	int16_t * 3
Description	The timestamp offset	Image frame ID	Accel x,y,z values	IMU temperature	Gyro x,y,z values

- Formula for converting the accel & gyro values to real ones: **real = data \* range / 0x10000** .
  - accel default range is **8 g**, gyro default range is **1000 deg/s**.
- Formula for converting the temperature to real value: **real = data / ratio + offset** .
  - default ratio is **326.8**, default offset is **25°C**.

# Chapter 3

## Device Control Specification

- [Control Protocols](#)
- [Control Channels](#)

### 3.1 Control Protocols

There are two control modes, one is through UVC standard protocol, the other is through UVC custom protocol with extension unit.

#### Standard Protocol

Name	Field	Bytes	Default	Min	Max	Stored	Flash Address	Note
Gain	gain	2	24	0	48	✓	0x12	valid if manual-exposure
Brightness	brightness/exposure_time↔	2	120	0	240	✓	0x14	valid if manual-exposure
Contrast	contrast/black_level↔ calibration	2	127	0	255	✓	0x10	valid if manual-exposure

#### Custom Protocol

Name	Field	Bytes	Default	Min	Max	Stored	Flash Address	Channel	Note
Frame rate	frame_rate↔	2	25	10	60	✓	0x21	XU_CAMERA_CTRL	values: {10,15,20,25,30,35,40,45,50,55,60}
IMU frequency	imu_frequency↔	2	200	100	500	✓	0x23	XU_CAMERA_CTRL	values: {100,200,250,333,500}

Name	Field	Bytes	Default	Min	Max	Stored	Flash Address	Channel	Note
Exposure mode	exposure↔ _mode	1	0	0	1	✓	0x0F	XU_CA↔ M_CTRL	0: enable auto-exposure; 1↔ : manual-exposure
Max gain	max_gain	2	48	0	48	✓	0x1D	XU_CA↔ M_CTRL	valid if auto-exposure
Max exposure time	max_↔ exposure↔ _time	2	240	0	240	✓	0x1B	XU_CA↔ M_CTRL	valid if auto-exposure
Desired brightness	desired↔ _↔ brightness	2	192	0	255	✓	0x19	XU_CA↔ M_CTRL	valid if auto-exposure
IR control	ir_control	1	0	0	160	×	-	XU_CA↔ M_CTRL	
HDR mode	hdr_↔ mode	1	0	0	1	✓	0x1F	XU_CA↔ M_CTRL	0: 10-bit; 1: 12-bit
Zero drift calibration	zero_↔ drift_↔ calibration		-	-	-	×	-	XU_HA↔ LF_DU↔ PLEX	
Erase chip	erase_↔ chip		-	-	-	×	-	XU_HA↔ LF_DU↔ PLEX	

## 3.2 Control Channels

Name	Field	Address	Bandwidth	Node
Camera control channel	XU_CAM_CTRL_CHANNEL	1	3	
Half-Duplex channel	XU_HALF_DUPLEX_CHANNEL	2	20	
IMU write channel	XU_IMUDATA_WRITE_CHANNEL	3	5	
IMU read channel	XU_IMUDATA_READ_CHANNEL	4	2000	
File channel	XU_FILE_CHANNEL	5	2000	

### Camera Control Channel

The channel provides get, set and query (min, max, default).

### Half-Duplex Channel

The channel only provides set, such as zero drift correction.

### IMU Channel

The channel is used to request and response IMU data, see [IMU Data](#).



## File Channel

The channel is used to read and write device information, image params, and IMU params.

Name	Header	Size	File	Checks
Bytes	1	2	-	1
Type	uint8↔ _t	uint16_t	-	uint8_t
Description	Flags	Content size	Content data	Checksum, XOR of all content bytes

Header Bit Subscript	Description
0	Device information
1	Image params
2	IMU params
3~6	Undefined
7	0: Get; 1: Set

## File Content Packet

Name	ID	Size	Content
Bytes	1	2	-
Type	uint8_t	uint16_t	-
Description	Content ID	Content size	Content data

File	ID	Max Size
Device information	1	250
Image params	2	250
IMU params	4	500



# Chapter 4

## Module Index

### 4.1 Modules

Here is a list of all modules:

Enumerations . . . . .	17
Intrinsics & Extrinsics . . . . .	22
Datatypes . . . . .	23
Utilities . . . . .	24



# Chapter 5

## Hierarchical Index

### 5.1 Class Hierarchy

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

mynteye::API	25
mynteye::AsyncCallback< Data >	29
mynteye::Context	29
mynteye::Device	30
mynteye::Extrinsics	33
mynteye::device::Frame	34
mynteye::ImgData	36
mynteye::ImuData	36
mynteye::ImuIntrinsics	37
mynteye::Intrinsics	38
mynteye::device::MotionData	38
mynteye::api::MotionData	39
mynteye::MotionIntrinsics	39
mynteye::Object	40
mynteye::ObjMat	40
mynteye::ObjMat2	41
mynteye::OptionInfo	41
mynteye::Plugin	42
runtime_error	
mynteye::strings_error	47
mynteye::device::StreamData	44
mynteye::api::StreamData	45
mynteye::StreamRequest	46



# Chapter 6

## Class Index

### 6.1 Class List

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

<a href="#">mynteye::API</a>	To communicate with MYNT® EYE device	25
<a href="#">mynteye::AsyncCallback&lt; Data &gt;</a>		29
<a href="#">mynteye::Context</a>	The context about devices	29
<a href="#">mynteye::Device</a>	To communicate with MYNT® EYE device	30
<a href="#">mynteye::Extrinsics</a>	Extrinsics, represent how the different datas are connected	33
<a href="#">mynteye::device::Frame</a>	Frame with raw data	34
<a href="#">mynteye::ImgData</a>	Image data	36
<a href="#">mynteye::ImuData</a>	IMU data	36
<a href="#">mynteye::ImuIntrinsics</a>	IMU intrinsics: scale, drift and variances	37
<a href="#">mynteye::Intrinsics</a>	Stream intrinsics,	38
<a href="#">mynteye::device::MotionData</a>	Device motion data	38
<a href="#">mynteye::api::MotionData</a>	API motion data	39
<a href="#">mynteye::MotionIntrinsics</a>	Motion intrinsics, including accelerometer and gyroscope	39
<a href="#">mynteye::Object</a>	Input & output object	40
<a href="#">mynteye::ObjMat</a>	Input & output object of one cv::Mat	40
<a href="#">mynteye::ObjMat2</a>	Input & output object of two cv::Mat	41
<a href="#">mynteye::OptionInfo</a>	Option info	41
<a href="#">mynteye::Plugin</a>	The plugin which could implement processing by yourself	42

---

<a href="#">mynteye::device::StreamData</a>	
Device stream data . . . . .	44
<a href="#">mynteye::api::StreamData</a>	
API stream data . . . . .	45
<a href="#">mynteye::StreamRequest</a>	
Stream request . . . . .	46
<a href="#">mynteye::strings_error</a>	
The strings error . . . . .	47



# Chapter 7

## Module Documentation

### 7.1 Enumerations

Public enumeration types.

#### Enumerations

- enum `mynteye::Model` : `std::uint8_t` { `mynteye::Model::STANDARD`, `mynteye::Model::LAST` }

*Device model.*

- enum `mynteye::Stream` : `std::uint8_t` {  
`mynteye::Stream::LEFT`, `mynteye::Stream::RIGHT`, `mynteye::Stream::LEFT_RECTIFIED`, `mynteye::Stream::RIGHT_RECTIFIED`,  
`mynteye::Stream::DISPARITY`, `mynteye::Stream::DISPARITY_NORMALIZED`, `mynteye::Stream::DEPTH`,  
`mynteye::Stream::POINTS`,  
`mynteye::Stream::LAST` }

*Streams define different type of data.*

- enum `mynteye::Capabilities` : `std::uint8_t` {  
`mynteye::Capabilities::STEREO`, `mynteye::Capabilities::COLOR`, `mynteye::Capabilities::DEPTH`, `mynteye::Capabilities::POINT`,  
`mynteye::Capabilities::FISHEYE`, `mynteye::Capabilities::INFRARED`, `mynteye::Capabilities::INFRARED2`,  
`mynteye::Capabilities::IMU`,  
`mynteye::Capabilities::LAST` }

*Capabilities define the full set of functionality that the device might provide.*

- enum `mynteye::Info` : `std::uint8_t` {  
`mynteye::Info::DEVICE_NAME`, `mynteye::Info::SERIAL_NUMBER`, `mynteye::Info::FIRMWARE_VERSION`,  
`mynteye::Info::HARDWARE_VERSION`,  
`mynteye::Info::SPEC_VERSION`, `mynteye::Info::LENS_TYPE`, `mynteye::Info::IMU_TYPE`, `mynteye::Info::NOMINAL_BASELINE`,  
`mynteye::Info::LAST` }

*Camera info fields are read-only strings that can be queried from the device.*

- enum `mynteye::Option` : `std::uint8_t` {  
`mynteye::Option::GAIN`, `mynteye::Option::BRIGHTNESS`, `mynteye::Option::CONTRAST`, `mynteye::Option::FRAME_RATE`,  
`mynteye::Option::IMU_FREQUENCY`, `mynteye::Option::EXPOSURE_MODE`, `mynteye::Option::MAX_GAIN`,  
`mynteye::Option::MAX_EXPOSURE_TIME`,  
`mynteye::Option::DESIRED_BRIGHTNESS`, `mynteye::Option::IR_CONTROL`, `mynteye::Option::HDR_MODE`,  
`mynteye::Option::ZERO_DRIFT_CALIBRATION`,  
`mynteye::Option::ERASE_CHIP`, `mynteye::Option::LAST` }

*Camera control options define general configuration controls.*

- enum `mynteye::Source` : `std::uint8_t` { `mynteye::Source::VIDEO_STREAMING`, `mynteye::Source::MOTION_TRACKING`,  
`mynteye::Source::ALL`, `mynteye::Source::LAST` }

Source allows the user to choose which data to be captured.

- enum `mynteye::AddOns` : `std::uint8_t` { `mynteye::AddOns::INFRARED`, `mynteye::AddOns::INFRARED2`, `mynteye::AddOns::LAST` }

Add-Ons are peripheral modules of our hardware.

- enum `mynteye::Format` : `std::uint32_t` { `mynteye::Format::GREY` =  $((\text{std::uint32\_t}('G') \mid (\text{std::uint32\_t}('R') \ll 8) \mid (\text{std::uint32\_t}('E') \ll 16) \mid (\text{std::uint32\_t}('Y') \ll 24))$ , `mynteye::Format::YUYV` =  $((\text{std::uint32\_t}('Y') \mid (\text{std::uint32\_t}('U') \ll 8) \mid (\text{std::uint32\_t}('Y') \ll 16) \mid (\text{std::uint32\_t}('V') \ll 24))$ , `mynteye::Format::LAST` }

Formats define how each stream can be encoded.

## 7.1.1 Detailed Description

Public enumeration types.

## 7.1.2 Enumeration Type Documentation

### 7.1.2.1 AddOns

```
enum mynteye::AddOns : std::uint8_t [strong]
```

Add-Ons are peripheral modules of our hardware.

#### Enumerator

INFRARED	Infrared.
INFRARED2	Second infrared.
LAST	Last guard.

### 7.1.2.2 Capabilities

```
enum mynteye::Capabilities : std::uint8_t [strong]
```

Capabilities define the full set of functionality that the device might provide.

#### Enumerator

STEREO	Provides stereo stream.
COLOR	Provides color stream.
DEPTH	Provides depth stream.
POINTS	Provides point cloud stream.
FISHEYE	Provides fisheye stream.
INFRARED	Provides infrared stream.
INFRARED2	Provides second infrared stream.
IMU	Provides IMU (accelerometer, gyroscope) data.
LAST	Last guard.

## 7.1.2.3 Format

```
enum mynteye::Format : std::uint32_t [strong]
```

Formats define how each stream can be encoded.

## Enumerator

GREY	Greyscale, 8 bits per pixel.
YUYV	YUV 4:2:2, 16 bits per pixel.
LAST	Last guard.

## 7.1.2.4 Info

```
enum mynteye::Info : std::uint8_t [strong]
```

Camera info fields are read-only strings that can be queried from the device.

## Enumerator

DEVICE_NAME	<a href="#">Device</a> name.
SERIAL_NUMBER	Serial number.
FIRMWARE_VERSION	Firmware version.
HARDWARE_VERSION	Hardware version.
SPEC_VERSION	Spec version.
LENS_TYPE	Lens type.
IMU_TYPE	IMU type.
NOMINAL_BASELINE	Nominal baseline.
LAST	Last guard.

## 7.1.2.5 Model

```
enum mynteye::Model : std::uint8_t [strong]
```

[Device](#) model.

## Enumerator

STANDARD	Standard.
LAST	Last guard.

### 7.1.2.6 Option

```
enum mynteye::Option : std::uint8_t [strong]
```

Camera control options define general configuration controls.

#### Enumerator

GAIN	Image gain, valid if manual-exposure. range: [0,48], default: 24
BRIGHTNESS	Image brightness, valid if manual-exposure. range: [0,240], default: 120
CONTRAST	Image contrast, valid if manual-exposure. range: [0,255], default: 127
FRAME_RATE	Image frame rate, must set IMU_FREQUENCY together. values: {10,15,20,25,30,35,40,45,50,55}, default: 25
IMU_FREQUENCY	IMU frequency, must set FRAME_RATE together. values: {100,200,250,333,500}, default: 200
EXPOSURE_MODE	Exposure mode. 0: enable auto-exposure 1: disable auto-exposure (manual-exposure)
MAX_GAIN	Max gain, valid if auto-exposure. range: [0,48], default: 48
MAX_EXPOSURE_TIME	Max exposure time, valid if auto-exposure. range: [0,240], default: 240
DESIRED_BRIGHTNESS	Desired brightness, valid if auto-exposure. range: [0,255], default: 192
IR_CONTROL	IR control. range: [0,160], default: 0
HDR_MODE	HDR mode. 0: 10-bit 1: 12-bit
ZERO_DRIFT_CALIBRATION	Zero drift calibration.
ERASE_CHIP	Erase chip.
LAST	Last guard.

### 7.1.2.7 Source

```
enum mynteye::Source : std::uint8_t [strong]
```

Source allows the user to choose which data to be captured.

#### Enumerator

VIDEO_STREAMING	Video streaming of stereo, color, depth, etc.
MOTION_TRACKING	Motion tracking of IMU (accelerometer, gyroscope)
ALL	Enable everything together.
LAST	Last guard.

### 7.1.2.8 Stream

```
enum mynteye::Stream : std::uint8_t [strong]
```

Streams define different type of data.

## Enumerator

LEFT	Left stream.
RIGHT	Right stream.
LEFT_RECTIFIED	Left stream, rectified.
RIGHT_RECTIFIED	Right stream, rectified.
DISPARITY	Disparity stream.
DISPARITY_NORMALIZED	Disparity stream, normalized.
DEPTH	Depth stream.
POINTS	Point cloud stream.
LAST	Last guard.

## 7.2 Intrinsic & Extrinsic

Intrinsic and extrinsic properties.

### Classes

- struct [mynteye::Intrinsic](#)  
*Stream intrinsic.*
- struct [mynteye::ImuIntrinsic](#)  
*IMU intrinsic: scale, drift and variances.*
- struct [mynteye::MotionIntrinsic](#)  
*Motion intrinsic, including accelerometer and gyroscope.*
- struct [mynteye::Extrinsic](#)  
*Extrinsic, represent how the different datas are connected.*

### 7.2.1 Detailed Description

Intrinsic and extrinsic properties.

## 7.3 Datatypes

Public data types.

### Classes

- struct `mynteye::api::StreamData`  
*API stream data.*
- struct `mynteye::api::MotionData`  
*API motion data.*
- class `mynteye::device::Frame`  
*Frame with raw data.*
- struct `mynteye::device::StreamData`  
*Device stream data.*
- struct `mynteye::device::MotionData`  
*Device motion data.*
- struct `mynteye::ImgData`  
*Image data.*
- struct `mynteye::ImuData`  
*IMU data.*
- struct `mynteye::OptionInfo`  
*Option info.*

### 7.3.1 Detailed Description

Public data types.

## 7.4 Utilities

### Functions

- MYNTEYE\_API `std::shared_ptr< Device > mynteye::device::select ()`  
*Detecting MYNT EYE devices and prompt user to select one.*
- MYNTEYE\_API `float mynteye::utils::get_real_exposure_time (std::int32_t frame_rate, std::uint16_t exposure_time)`  
*Get real exposure time in ms from virtual value, according to its frame rate.*

#### 7.4.1 Detailed Description

#### 7.4.2 Function Documentation

##### 7.4.2.1 `get_real_exposure_time()`

```
MYNTEYE_API float mynteye::utils::get_real_exposure_time (
    std::int32_t frame_rate,
    std::uint16_t exposure_time )
```

Get real exposure time in ms from virtual value, according to its frame rate.

##### Parameters

<i>frame_rate</i>	the frame rate of the device.
<i>exposure_time</i>	the virtual exposure time.

##### Returns

the real exposure time in ms, or the virtual value if frame rate is invalid.

##### 7.4.2.2 `select()`

```
MYNTEYE_API std::shared_ptr<Device> mynteye::device::select ( )
```

Detecting MYNT EYE devices and prompt user to select one.

##### Returns

the selected device, or `nullptr` if none.



# Chapter 8

## Class Documentation

### 8.1 mynteye::API Class Reference

The [API](#) class to communicate with MYNT® EYE device.

#### Public Types

- using [stream\\_callback\\_t](#) = std::function< void(const [api::StreamData](#) &data)>  
*The [api::StreamData](#) callback.*
- using [motion\\_callback\\_t](#) = std::function< void(const [api::MotionData](#) &data)>  
*The [api::MotionData](#) callback.*

#### Public Member Functions

- [Model GetModel](#) () const  
*Get the model.*
- bool [Supports](#) (const [Stream](#) &stream) const  
*Supports the stream or not.*
- bool [Supports](#) (const [Capabilities](#) &capability) const  
*Supports the capability or not.*
- bool [Supports](#) (const [Option](#) &option) const  
*Supports the option or not.*
- bool [Supports](#) (const [AddOns](#) &addon) const  
*Supports the addon or not.*
- const std::vector< [StreamRequest](#) > & [GetStreamRequests](#) (const [Capabilities](#) &capability) const  
*Get all stream requests of the capability.*
- void [ConfigStreamRequest](#) (const [Capabilities](#) &capability, const [StreamRequest](#) &request)  
*Config the stream request to the capability.*
- std::string [GetInfo](#) (const [Info](#) &info) const  
*Get the device info.*
- [Intrinsics GetIntrinsics](#) (const [Stream](#) &stream) const  
*Get the intrinsics of stream.*
- [Extrinsics GetExtrinsics](#) (const [Stream](#) &from, const [Stream](#) &to) const  
*Get the extrinsics from one stream to another.*

- [MotionIntrinsics GetMotionIntrinsics](#) () const  
*Get the intrinsics of motion.*
- [Extrinsics GetMotionExtrinsics](#) (const [Stream](#) &from) const  
*Get the extrinsics from one stream to motion.*
- void [LogOptionInfos](#) () const  
*Log all option infos.*
- [OptionInfo GetOptionInfo](#) (const [Option](#) &option) const  
*Get the option info.*
- std::int32\_t [GetOptionValue](#) (const [Option](#) &option) const  
*Get the option value.*
- void [SetOptionValue](#) (const [Option](#) &option, std::int32\_t value)  
*Set the option value.*
- bool [RunOptionAction](#) (const [Option](#) &option) const  
*Run the option action.*
- void [SetStreamCallback](#) (const [Stream](#) &stream, [stream\\_callback\\_t](#) callback)  
*Set the callback of stream.*
- void [SetMotionCallback](#) ([motion\\_callback\\_t](#) callback)  
*Set the callback of motion.*
- bool [HasStreamCallback](#) (const [Stream](#) &stream) const  
*Has the callback of stream.*
- bool [HasMotionCallback](#) () const  
*Has the callback of motion.*
- void [Start](#) (const [Source](#) &source)  
*Start capturing the source.*
- void [Stop](#) (const [Source](#) &source)  
*Stop capturing the source.*
- void [WaitForStreams](#) ()  
*Wait the streams are ready.*
- void [EnableStreamData](#) (const [Stream](#) &stream)  
*Enable the data of stream.*
- void [DisableStreamData](#) (const [Stream](#) &stream)  
*Disable the data of stream.*
- [api::StreamData GetStreamData](#) (const [Stream](#) &stream)  
*Get the latest data of stream.*
- std::vector< [api::StreamData](#) > [GetStreamDatas](#) (const [Stream](#) &stream)  
*Get the datas of stream.*
- void [EnableMotionDatas](#) (std::size\_t max\_size=std::numeric\_limits< std::size\_t >::max())  
*Enable cache motion datas.*
- std::vector< [api::MotionData](#) > [GetMotionDatas](#) ()  
*Get the motion datas.*
- void [EnablePlugin](#) (const std::string &path)  
*Enable the plugin.*

### Static Public Member Functions

- static std::shared\_ptr< [API](#) > [Create](#) ()  
*Create the [API](#) instance.*
- static std::shared\_ptr< [API](#) > [Create](#) (std::shared\_ptr< [Device](#) > device)  
*Create the [API](#) instance.*
- static std::shared\_ptr< [API](#) > [Create](#) (int argc, char \*argv[])  
*Create the [API](#) instance.*
- static std::shared\_ptr< [API](#) > [Create](#) (int argc, char \*argv[], std::shared\_ptr< [Device](#) > device)  
*Create the [API](#) instance.*

### 8.1.1 Detailed Description

The [API](#) class to communicate with MYNT® EYE device.

### 8.1.2 Member Typedef Documentation

#### 8.1.2.1 motion\_callback\_t

```
using mynteye::API::motion_callback_t = std::function<void(const api::MotionData &data)>
```

The [api::MotionData](#) callback.

#### 8.1.2.2 stream\_callback\_t

```
using mynteye::API::stream_callback_t = std::function<void(const api::StreamData &data)>
```

The [api::StreamData](#) callback.

### 8.1.3 Member Function Documentation

#### 8.1.3.1 Create() [1/4]

```
static std::shared_ptr<API> mynteye::API::Create ( ) [static]
```

Create the [API](#) instance.

##### Returns

the [API](#) instance.

##### Note

This will call [device::select\(\)](#) to select a device.

#### 8.1.3.2 Create() [2/4]

```
static std::shared_ptr<API> mynteye::API::Create (
    std::shared_ptr< Device > device ) [static]
```

Create the [API](#) instance.

**Parameters**

<i>device</i>	the selected device.
---------------	----------------------

**Returns**

the [API](#) instance.

**8.1.3.3 Create()** [3/4]

```
static std::shared_ptr<API> mynteye::API::Create (
    int argc,
    char * argv[] ) [static]
```

Create the [API](#) instance.

**Parameters**

<i>argc</i>	the arg count.
<i>argv</i>	the arg values.

**Returns**

the [API](#) instance.

**Note**

This will init glog with args and call [device::select\(\)](#) to select a device.

**8.1.3.4 Create()** [4/4]

```
static std::shared_ptr<API> mynteye::API::Create (
    int argc,
    char * argv[],
    std::shared_ptr< Device > device ) [static]
```

Create the [API](#) instance.

**Parameters**

<i>argc</i>	the arg count.
<i>argv</i>	the arg values.
<i>device</i>	the selected device.

**Returns**

the [API](#) instance.

**Note**

This will init glog with args.

**8.1.3.5 EnableStreamData()**

```
void mynteye::API::EnableStreamData (
    const Stream & stream )
```

Enable the data of stream.

**Note**

must enable the stream if it's a synthetic one. This means the stream is not native, the device has the capability to provide this stream, but still support this stream.

**8.1.3.6 GetStreamDatas()**

```
std::vector<api::StreamData> mynteye::API::GetStreamDatas (
    const Stream & stream )
```

Get the datas of stream.

**Note**

default cache 4 datas at most.

## 8.2 mynteye::AsyncCallback< Data > Class Template Reference

## 8.3 mynteye::Context Class Reference

The context about devices.

**Public Member Functions**

- `std::vector< std::shared_ptr< Device > > devices () const`  
*Get all devices now.*

### 8.3.1 Detailed Description

The context about devices.

### 8.3.2 Member Function Documentation

#### 8.3.2.1 devices()

```
std::vector<std::shared_ptr<Device> > mynteye::Context::devices ( ) const [inline]
```

Get all devices now.

#### Returns

a vector of all devices.

## 8.4 mynteye::Device Class Reference

The [Device](#) class to communicate with MYNT® EYE device.

### Public Types

- using [stream\\_callback\\_t](#) = device::StreamCallback  
*The [device::StreamData](#) callback.*
- using [motion\\_callback\\_t](#) = device::MotionCallback  
*The [device::MotionData](#) callback.*

### Public Member Functions

- [Model GetModel](#) ( ) const  
*Get the model.*
- bool [Supports](#) (const [Stream](#) &stream) const  
*Supports the stream or not.*
- bool [Supports](#) (const [Capabilities](#) &capability) const  
*Supports the capability or not.*
- bool [Supports](#) (const [Option](#) &option) const  
*Supports the option or not.*
- bool [Supports](#) (const [AddOns](#) &addon) const  
*Supports the addon or not.*
- const std::vector< [StreamRequest](#) > & [GetStreamRequests](#) (const [Capabilities](#) &capability) const  
*Get all stream requests of the capability.*
- void [ConfigStreamRequest](#) (const [Capabilities](#) &capability, const [StreamRequest](#) &request)  
*Config the stream request to the capability.*
- std::shared\_ptr< [DeviceInfo](#) > [GetInfo](#) ( ) const

- Get the device info.*

  - `std::string GetInfo` (const `Info` &info) const
- Get the device info of a field.*

  - `Intrinsics GetIntrinsics` (const `Stream` &stream) const
- Get the intrinsics of stream.*

  - `Extrinsics GetExtrinsics` (const `Stream` &from, const `Stream` &to) const
- Get the extrinsics from one stream to another.*

  - `MotionIntrinsics GetMotionIntrinsics` () const
- Get the intrinsics of motion.*

  - `Extrinsics GetMotionExtrinsics` (const `Stream` &from) const
- Get the extrinsics from one stream to motion.*

  - `Intrinsics GetIntrinsics` (const `Stream` &stream, bool \*ok) const
- Get the intrinsics of stream.*

  - `Extrinsics GetExtrinsics` (const `Stream` &from, const `Stream` &to, bool \*ok) const
- Get the extrinsics from one stream to another.*

  - `MotionIntrinsics GetMotionIntrinsics` (bool \*ok) const
- Get the intrinsics of motion.*

  - `Extrinsics GetMotionExtrinsics` (const `Stream` &from, bool \*ok) const
- Get the extrinsics from one stream to motion.*

  - void `SetIntrinsics` (const `Stream` &stream, const `Intrinsics` &in)
- Set the intrinsics of stream.*

  - void `SetExtrinsics` (const `Stream` &from, const `Stream` &to, const `Extrinsics` &ex)
- Set the extrinsics from one stream to another.*

  - void `SetMotionIntrinsics` (const `MotionIntrinsics` &in)
- Set the intrinsics of motion.*

  - void `SetMotionExtrinsics` (const `Stream` &from, const `Extrinsics` &ex)
- Set the extrinsics from one stream to motion.*

  - void `LogOptionInfos` () const
- Log all option infos.*

  - `OptionInfo GetOptionInfo` (const `Option` &option) const
- Get the option info.*

  - `std::int32_t GetOptionValue` (const `Option` &option) const
- Get the option value.*

  - void `SetOptionValue` (const `Option` &option, `std::int32_t` value)
- Set the option value.*

  - bool `RunOptionAction` (const `Option` &option) const
- Run the option action.*

  - void `SetStreamCallback` (const `Stream` &stream, `stream_callback_t` callback, bool async=false)
- Set the callback of stream.*

  - void `SetMotionCallback` (`motion_callback_t` callback, bool async=false)
- Set the callback of motion.*

  - bool `HasStreamCallback` (const `Stream` &stream) const
- Has the callback of stream.*

  - bool `HasMotionCallback` () const
- Has the callback of motion.*

  - virtual void `Start` (const `Source` &source)
- Start capturing the source.*

  - virtual void `Stop` (const `Source` &source)
- Stop capturing the source.*

  - void `WaitForStreams` ()
- Wait the streams are ready.*

- `std::vector< device::StreamData > GetStreamDatas (const Stream &stream)`  
*Get the datas of stream.*
- `device::StreamData GetLatestStreamData (const Stream &stream)`  
*Get the latest data of stream.*
- `void EnableMotionDatas (std::size_t max_size=std::numeric_limits< std::size_t >::max())`  
*Enable cache motion datas.*
- `std::vector< device::MotionData > GetMotionDatas ()`  
*Get the motion datas.*

## Static Public Member Functions

- `static std::shared_ptr< Device > Create (const std::string &name, std::shared_ptr< uvc::device > device)`  
*Create the [Device](#) instance.*

### 8.4.1 Detailed Description

The [Device](#) class to communicate with MYNT® EYE device.

### 8.4.2 Member Typedef Documentation

#### 8.4.2.1 `motion_callback_t`

```
using mynteye::Device::motion\_callback\_t = device::MotionCallback
```

The [device::MotionData](#) callback.

#### 8.4.2.2 `stream_callback_t`

```
using mynteye::Device::stream\_callback\_t = device::StreamCallback
```

The [device::StreamData](#) callback.

### 8.4.3 Member Function Documentation

#### 8.4.3.1 `Create()`

```
static std::shared_ptr<Device> mynteye::Device::Create (
    const std::string & name,
    std::shared_ptr< uvc::device > device ) [static]
```

Create the [Device](#) instance.



## Parameters

<i>name</i>	the device name.
<i>device</i>	the device from uvc.

## Returns

the [Device](#) instance.

## 8.4.3.2 GetStreamDatas()

```
std::vector<device::StreamData> mynteye::Device::GetStreamDatas (
    const Stream & stream )
```

Get the datas of stream.

## Note

default cache 4 datas at most.

## 8.5 mynteye::Extrinsics Struct Reference

[Extrinsics](#), represent how the different datas are connected.

### Public Member Functions

- [Extrinsics Inverse](#) () const  
*Inverse this extrinsics.*

### Public Attributes

- double [rotation](#) [3][3]  
*Rotation matrix.*
- double [translation](#) [3]  
*Translation vector.*

#### 8.5.1 Detailed Description

[Extrinsics](#), represent how the different datas are connected.

#### 8.5.2 Member Function Documentation

### 8.5.2.1 Inverse()

```
Extrinsics mynteye::Extrinsics::Inverse ( ) const [inline]
```

Inverse this extrinsics.

#### Returns

the inversed extrinsics.

## 8.6 mynteye::device::Frame Class Reference

[Frame](#) with raw data.

### Public Member Functions

- [Frame](#) (const [StreamRequest](#) &request, const void \*data)  
*Construct the frame with [StreamRequest](#) and raw data.*
- [Frame](#) (std::uint16\_t width, std::uint16\_t height, [Format](#) format, const void \*data)  
*Construct the frame with stream info and raw data.*
- std::uint16\_t width () const  
*Get the width.*
- std::uint16\_t height () const  
*Get the height.*
- [Format](#) format () const  
*Get the format.*
- std::uint8\_t \* data ()  
*Get the data.*
- const std::uint8\_t \* data () const  
*Get the const data.*
- std::size\_t size () const  
*Get the size of data.*
- [Frame](#) clone () const  
*Clone a new frame.*

### 8.6.1 Detailed Description

[Frame](#) with raw data.

### 8.6.2 Member Function Documentation

### 8.6.2.1 clone()

```
Frame mynteye::device::Frame::clone ( ) const [inline]
```

Clone a new frame.

### 8.6.2.2 data() [1/2]

```
std::uint8_t* mynteye::device::Frame::data ( ) [inline]
```

Get the data.

### 8.6.2.3 data() [2/2]

```
const std::uint8_t* mynteye::device::Frame::data ( ) const [inline]
```

Get the const data.

### 8.6.2.4 format()

```
Format mynteye::device::Frame::format ( ) const [inline]
```

Get the format.

### 8.6.2.5 height()

```
std::uint16_t mynteye::device::Frame::height ( ) const [inline]
```

Get the height.

### 8.6.2.6 size()

```
std::size_t mynteye::device::Frame::size ( ) const [inline]
```

Get the size of data.

### 8.6.2.7 width()

```
std::uint16_t mynteye::device::Frame::width ( ) const [inline]
```

Get the width.

## 8.7 mynteye::ImgData Struct Reference

Image data.

### Public Attributes

- `std::uint16_t` [frame\\_id](#)  
*Image frame id.*
- `std::uint32_t` [timestamp](#)  
*Image timestamp in 0.01ms.*
- `std::uint16_t` [exposure\\_time](#)  
*Image exposure time, virtual value in [1, 480].*

#### 8.7.1 Detailed Description

Image data.

## 8.8 mynteye::ImuData Struct Reference

IMU data.

### Public Attributes

- `std::uint16_t` [frame\\_id](#)  
*Image frame id.*
- `std::uint32_t` [timestamp](#)  
*IMU timestamp in 0.01ms.*
- `double` [accel](#) [3]  
*IMU accelerometer data for 3-axis: X, Y, Z.*
- `double` [gyro](#) [3]  
*IMU gyroscope data for 3-axis: X, Y, Z.*
- `double` [temperature](#)  
*IMU temperature.*

#### 8.8.1 Detailed Description

IMU data.

## 8.8.2 Member Data Documentation

### 8.8.2.1 accel

```
double mynteye::ImuData::accel[3]
```

IMU accelerometer data for 3-axis: X, Y, Z.

### 8.8.2.2 gyro

```
double mynteye::ImuData::gyro[3]
```

IMU gyroscope data for 3-axis: X, Y, Z.

## 8.9 mynteye::ImuIntrinsics Struct Reference

IMU intrinsics: scale, drift and variances.

### Public Attributes

- double [scale](#) [3][3]  
*Scale matrix.*
- double [noise](#) [3]  
*Noise density variances.*
- double [bias](#) [3]  
*Random walk variances.*

### 8.9.1 Detailed Description

IMU intrinsics: scale, drift and variances.

## 8.9.2 Member Data Documentation

### 8.9.2.1 scale

```
double mynteye::ImuIntrinsics::scale[3][3]
```

Scale matrix.

```
Scale X      cross axis  cross axis
cross axis  Scale Y      cross axis
cross axis  cross axis  Scale Z
```

## 8.10 mynteye::Intrinsics Struct Reference

Stream intrinsics,.

### Public Attributes

- `std::uint16_t width`  
*The width of the image in pixels.*
- `std::uint16_t height`  
*The height of the image in pixels.*
- `double fx`  
*The focal length of the image plane, as a multiple of pixel width.*
- `double fy`  
*The focal length of the image plane, as a multiple of pixel height.*
- `double cx`  
*The horizontal coordinate of the principal point of the image.*
- `double cy`  
*The vertical coordinate of the principal point of the image.*
- `std::uint8_t model`  
*The distortion model of the image.*
- `double coeffs [5]`  
*The distortion coefficients:  $k_1, k_2, p_1, p_2, k_3$ .*

### 8.10.1 Detailed Description

Stream intrinsics,.

## 8.11 mynteye::device::MotionData Struct Reference

[Device](#) motion data.

### Public Attributes

- `std::shared_ptr< ImuData > imu`  
*[ImuData](#).*

### 8.11.1 Detailed Description

[Device](#) motion data.

### 8.11.2 Member Data Documentation

### 8.11.2.1 imu

```
std::shared_ptr<ImuData> mynteye::device::MotionData::imu
```

[ImuData](#).

## 8.12 mynteye::api::MotionData Struct Reference

[API](#) motion data.

### Public Attributes

- `std::shared_ptr< ImuData > imu`  
*[ImuData](#)*.

### 8.12.1 Detailed Description

[API](#) motion data.

### 8.12.2 Member Data Documentation

#### 8.12.2.1 imu

```
std::shared_ptr<ImuData> mynteye::api::MotionData::imu
```

[ImuData](#).

## 8.13 mynteye::MotionIntrinsics Struct Reference

Motion intrinsics, including accelerometer and gyroscope.

### Public Attributes

- `ImuIntrinsics accel`  
*Accelerometer intrinsics.*
- `ImuIntrinsics gyro`  
*Gyroscope intrinsics.*

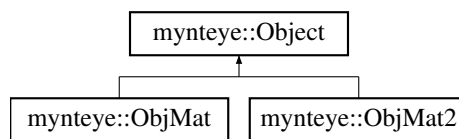
### 8.13.1 Detailed Description

Motion intrinsics, including accelerometer and gyroscope.

## 8.14 mynteye::Object Struct Reference

Input & output object.

Inheritance diagram for mynteye::Object:



### Static Public Member Functions

- `template<typename T >`  
`static T * Cast (Object *obj)`  
*Cast the obj to T pointer.*
- `template<typename T >`  
`static const T * Cast (const Object *obj)`  
*Cast the obj to const T pointer.*

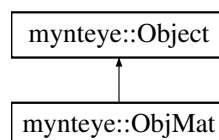
### 8.14.1 Detailed Description

Input & output object.

## 8.15 mynteye::ObjMat Struct Reference

Input & output object of one `cv::Mat`.

Inheritance diagram for mynteye::ObjMat:



### Public Attributes

- `cv::Mat value`  
*The value.*



## Additional Inherited Members

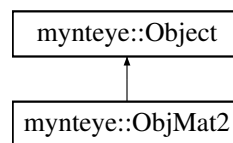
### 8.15.1 Detailed Description

Input & output object of one cv::Mat.

## 8.16 mynteye::ObjMat2 Struct Reference

Input & output object of two cv::Mat.

Inheritance diagram for mynteye::ObjMat2:



## Public Attributes

- cv::Mat [first](#)  
*The first value.*
- cv::Mat [second](#)  
*The second value.*

## Additional Inherited Members

### 8.16.1 Detailed Description

Input & output object of two cv::Mat.

## 8.17 mynteye::OptionInfo Struct Reference

Option info.

## Public Attributes

- std::int32\_t [min](#)  
*Minimum value.*
- std::int32\_t [max](#)  
*Maximum value.*
- std::int32\_t [def](#)  
*Default value.*

### 8.17.1 Detailed Description

Option info.

## 8.18 mynteye::Plugin Class Reference

The plugin which could implement processing by yourself.

### Public Member Functions

- virtual void [OnCreate](#) ([API](#) \*api)  
*Called when plugin created.*
- virtual bool [OnRectifyProcess](#) ([Object](#) \*const in, [Object](#) \*const out)  
*Called when process rectify.*
- virtual bool [OnDisparityProcess](#) ([Object](#) \*const in, [Object](#) \*const out)  
*Called when process disparity.*
- virtual bool [OnDisparityNormalizedProcess](#) ([Object](#) \*const in, [Object](#) \*const out)  
*Called when process normalized disparity.*
- virtual bool [OnPointsProcess](#) ([Object](#) \*const in, [Object](#) \*const out)  
*Called when process points.*
- virtual bool [OnDepthProcess](#) ([Object](#) \*const in, [Object](#) \*const out)  
*Called when process depth.*

### 8.18.1 Detailed Description

The plugin which could implement processing by yourself.

### 8.18.2 Member Function Documentation

#### 8.18.2.1 OnCreate()

```
virtual void mynteye::Plugin::OnCreate (  
    API * api ) [inline], [virtual]
```

Called when plugin created.

#### Parameters

<i>api</i>	the <a href="#">API</a> instacne.
------------	-----------------------------------

### 8.18.2.2 OnDepthProcess()

```
virtual bool mynteye::Plugin::OnDepthProcess (
    Object *const in,
    Object *const out ) [inline], [virtual]
```

Called when process depth.

#### Parameters

<i>in</i>	input object.
<i>out</i>	output object.

#### Returns

`true` if you process depth.

### 8.18.2.3 OnDisparityNormalizedProcess()

```
virtual bool mynteye::Plugin::OnDisparityNormalizedProcess (
    Object *const in,
    Object *const out ) [inline], [virtual]
```

Called when process normalized disparity.

#### Parameters

<i>in</i>	input object.
<i>out</i>	output object.

#### Returns

`true` if you process normalized disparity.

### 8.18.2.4 OnDisparityProcess()

```
virtual bool mynteye::Plugin::OnDisparityProcess (
    Object *const in,
    Object *const out ) [inline], [virtual]
```

Called when process disparity.

#### Parameters

<i>in</i>	input object.
<i>out</i>	output object.

**Returns**

`true` if you process disparity.

**8.18.2.5 OnPointsProcess()**

```
virtual bool mynteye::Plugin::OnPointsProcess (
    Object *const in,
    Object *const out ) [inline], [virtual]
```

Called when process points.

**Parameters**

<i>in</i>	input object.
<i>out</i>	output object.

**Returns**

`true` if you process points.

**8.18.2.6 OnRectifyProcess()**

```
virtual bool mynteye::Plugin::OnRectifyProcess (
    Object *const in,
    Object *const out ) [inline], [virtual]
```

Called when process rectify.

**Parameters**

<i>in</i>	input object.
<i>out</i>	output object.

**Returns**

`true` if you process rectify.

**8.19 mynteye::device::StreamData Struct Reference**

[Device](#) stream data.

## Public Attributes

- `std::shared_ptr< ImgData > img`  
*ImgData.*
- `std::shared_ptr< Frame > frame`  
*Frame.*

### 8.19.1 Detailed Description

[Device](#) stream data.

### 8.19.2 Member Data Documentation

#### 8.19.2.1 `frame`

`std::shared_ptr<Frame> mynteye::device::StreamData::frame`

[Frame](#).

#### 8.19.2.2 `img`

`std::shared_ptr<ImgData> mynteye::device::StreamData::img`

[ImgData](#).

## 8.20 mynteye::api::StreamData Struct Reference

[API](#) stream data.

## Public Attributes

- `std::shared_ptr< ImgData > img`  
*ImgData.*
- `cv::Mat frame`  
*Frame.*
- `std::shared_ptr< device::Frame > frame_raw`  
*Raw frame.*

### 8.20.1 Detailed Description

[API](#) stream data.

## 8.20.2 Member Data Documentation

### 8.20.2.1 frame

```
cv::Mat mynteye::api::StreamData::frame
```

Frame.

### 8.20.2.2 frame\_raw

```
std::shared_ptr<device::Frame> mynteye::api::StreamData::frame_raw
```

Raw frame.

### 8.20.2.3 img

```
std::shared_ptr<ImgData> mynteye::api::StreamData::img
```

[ImgData](#).

## 8.21 mynteye::StreamRequest Struct Reference

Stream request.

### Public Attributes

- [std::uint16\\_t width](#)  
*Stream width in pixels.*
- [std::uint16\\_t height](#)  
*Stream height in pixels.*
- [Format format](#)  
*Stream pixel format.*
- [std::uint16\\_t fps](#)  
*Stream frames per second (unused)*

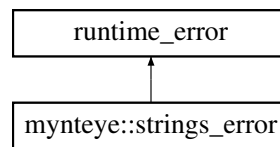
### 8.21.1 Detailed Description

Stream request.

## 8.22 mynteye::strings\_error Class Reference

The strings error.

Inheritance diagram for mynteye::strings\_error:



### 8.22.1 Detailed Description

The strings error.





# Index

- accel
  - [mynteye::ImuData, 37](#)
- AddOns
  - [Enumerations, 18](#)
- Capabilities
  - [Enumerations, 18](#)
- clone
  - [mynteye::device::Frame, 34](#)
- Create
  - [mynteye::API, 27, 28](#)
  - [mynteye::Device, 32](#)
- data
  - [mynteye::device::Frame, 35](#)
- Datatypes, [23](#)
- devices
  - [mynteye::Context, 30](#)
- EnableStreamData
  - [mynteye::API, 29](#)
- Enumerations, [17](#)
  - [AddOns, 18](#)
  - [Capabilities, 18](#)
  - [Format, 19](#)
  - [Info, 19](#)
  - [Model, 19](#)
  - [Option, 19](#)
  - [Source, 20](#)
  - [Stream, 20](#)
- Format
  - [Enumerations, 19](#)
- format
  - [mynteye::device::Frame, 35](#)
- frame
  - [mynteye::api::StreamData, 46](#)
  - [mynteye::device::StreamData, 45](#)
- frame\_raw
  - [mynteye::api::StreamData, 46](#)
- get\_real\_exposure\_time
  - [Utilities, 24](#)
- GetStreamDatas
  - [mynteye::API, 29](#)
  - [mynteye::Device, 33](#)
- gyro
  - [mynteye::ImuData, 37](#)
- height
  - [mynteye::device::Frame, 35](#)
- img
  - [mynteye::api::StreamData, 46](#)
  - [mynteye::device::StreamData, 45](#)
- imu
  - [mynteye::api::MotionData, 39](#)
  - [mynteye::device::MotionData, 38](#)
- Info
  - [Enumerations, 19](#)
- Intrinsics & Extrinsics, [22](#)
- Inverse
  - [mynteye::Extrinsics, 33](#)
- Model
  - [Enumerations, 19](#)
- motion\_callback\_t
  - [mynteye::API, 27](#)
  - [mynteye::Device, 32](#)
- mynteye::API, [25](#)
  - [Create, 27, 28](#)
  - [EnableStreamData, 29](#)
  - [GetStreamDatas, 29](#)
  - [motion\\_callback\\_t, 27](#)
  - [stream\\_callback\\_t, 27](#)
- [mynteye::AsyncCallback< Data >, 29](#)
- [mynteye::Context, 29](#)
  - [devices, 30](#)
- [mynteye::Device, 30](#)
  - [Create, 32](#)
  - [GetStreamDatas, 33](#)
  - [motion\\_callback\\_t, 32](#)
  - [stream\\_callback\\_t, 32](#)
- [mynteye::Extrinsics, 33](#)
  - [Inverse, 33](#)
- [mynteye::ImgData, 36](#)
- [mynteye::ImuData, 36](#)
  - [accel, 37](#)
  - [gyro, 37](#)
- [mynteye::ImuIntrinsics, 37](#)
  - [scale, 37](#)
- [mynteye::Intrinsics, 38](#)
- [mynteye::MotionIntrinsics, 39](#)
- [mynteye::ObjMat, 40](#)
- [mynteye::ObjMat2, 41](#)
- [mynteye::Object, 40](#)
- [mynteye::OptionInfo, 41](#)
- [mynteye::Plugin, 42](#)
  - [OnCreate, 42](#)
  - [OnDepthProcess, 42](#)
  - [OnDisparityNormalizedProcess, 43](#)
  - [OnDisparityProcess, 43](#)

- OnPointsProcess, [44](#)
- OnRectifyProcess, [44](#)
- mynteye::StreamRequest, [46](#)
- mynteye::api::MotionData, [39](#)
  - imu, [39](#)
- mynteye::api::StreamData, [45](#)
  - frame, [46](#)
  - frame\_raw, [46](#)
  - img, [46](#)
- mynteye::device::Frame, [34](#)
  - clone, [34](#)
  - data, [35](#)
  - format, [35](#)
  - height, [35](#)
  - size, [35](#)
  - width, [35](#)
- mynteye::device::MotionData, [38](#)
  - imu, [38](#)
- mynteye::device::StreamData, [44](#)
  - frame, [45](#)
  - img, [45](#)
- mynteye::strings\_error, [47](#)
  
- OnCreate
  - mynteye::Plugin, [42](#)
- OnDepthProcess
  - mynteye::Plugin, [42](#)
- OnDisparityNormalizedProcess
  - mynteye::Plugin, [43](#)
- OnDisparityProcess
  - mynteye::Plugin, [43](#)
- OnPointsProcess
  - mynteye::Plugin, [44](#)
- OnRectifyProcess
  - mynteye::Plugin, [44](#)
- Option
  - Enumerations, [19](#)
  
- scale
  - mynteye::ImuIntrinsics, [37](#)
- select
  - Utilities, [24](#)
- size
  - mynteye::device::Frame, [35](#)
- Source
  - Enumerations, [20](#)
- Stream
  - Enumerations, [20](#)
- stream\_callback\_t
  - mynteye::API, [27](#)
  - mynteye::Device, [32](#)
  
- Utilities, [24](#)
  - get\_real\_exposure\_time, [24](#)
  - select, [24](#)
  
- width
  - mynteye::device::Frame, [35](#)