

# MYNT EYE S SDK

2.3.3

Generated by Doxygen 1.8.15



---

<b>1 MYNT EYE S SDK</b>	<b>1</b>
<b>2 Device Data Specification</b>	<b>3</b>
2.1 Device Information . . . . .	3
2.2 Image Params . . . . .	4
2.3 IMU Params . . . . .	4
2.4 Image Data . . . . .	5
2.5 IMU Data . . . . .	5
<b>3 Device Control Specification</b>	<b>7</b>
3.1 Control Protocols . . . . .	7
3.2 Control Channels . . . . .	8
<b>4 Deprecated List</b>	<b>11</b>
<b>5 Module Index</b>	<b>13</b>
5.1 Modules . . . . .	13
<b>6 Hierarchical Index</b>	<b>15</b>
6.1 Class Hierarchy . . . . .	15
<b>7 Class Index</b>	<b>17</b>
7.1 Class List . . . . .	17
<b>8 Module Documentation</b>	<b>19</b>
8.1 Utilities . . . . .	19
8.1.1 Detailed Description . . . . .	19
8.1.2 Function Documentation . . . . .	19
8.1.2.1 get_real_exposure_time() . . . . .	19
8.1.2.2 select() . . . . .	20
8.1.2.3 select_request() . . . . .	20
8.2 Enumerations . . . . .	21
8.2.1 Detailed Description . . . . .	22
8.2.2 Enumeration Type Documentation . . . . .	22
8.2.2.1 AddOns . . . . .	22
8.2.2.2 Capabilities . . . . .	22
8.2.2.3 Format . . . . .	22
8.2.2.4 Info . . . . .	23
8.2.2.5 Model . . . . .	23
8.2.2.6 Option . . . . .	23
8.2.2.7 Source . . . . .	25
8.2.2.8 Stream . . . . .	26
8.3 Intrinsic & Extrinsic . . . . .	27
8.3.1 Detailed Description . . . . .	27
8.3.2 Enumeration Type Documentation . . . . .	27

---

---

8.3.2.1 CalibrationModel	27
8.4 Datatypes	28
8.4.1 Detailed Description	28
<b>9 Class Documentation</b>	<b>29</b>
9.1 mynteye::API Class Reference	29
9.1.1 Detailed Description	31
9.1.2 Member Typedef Documentation	31
9.1.2.1 motion_callback_t	31
9.1.2.2 stream_callback_t	31
9.1.3 Member Function Documentation	31
9.1.3.1 Create() [1/3]	31
9.1.3.2 Create() [2/3]	32
9.1.3.3 Create() [3/3]	32
9.1.3.4 EnableStreamData()	33
9.1.3.5 GetIntrinsics()	33
9.1.3.6 GetStreamDatas()	33
9.2 mynteye::AsyncCallback< Data > Class Template Reference	33
9.3 mynteye::Context Class Reference	33
9.3.1 Detailed Description	34
9.3.2 Member Function Documentation	34
9.3.2.1 devices()	34
9.4 mynteye::Device Class Reference	34
9.4.1 Detailed Description	36
9.4.2 Member Typedef Documentation	36
9.4.2.1 motion_callback_t	37
9.4.2.2 stream_callback_t	37
9.4.3 Member Function Documentation	37
9.4.3.1 Create()	37
9.4.3.2 GetLatestStreamData()	37
9.4.3.3 GetStreamDatas()	38
9.5 mynteye::DeviceInfo Struct Reference	38
9.5.1 Detailed Description	38
9.6 mynteye::Extrinsics Struct Reference	38
9.6.1 Detailed Description	38
9.6.2 Member Function Documentation	38
9.6.2.1 Inverse()	39
9.7 mynteye::device::Frame Class Reference	39
9.7.1 Detailed Description	39
9.7.2 Member Function Documentation	39
9.7.2.1 clone()	40
9.7.2.2 data() [1/2]	40

---

---

9.7.2.3 data() [2/2]	40
9.7.2.4 format()	40
9.7.2.5 height()	40
9.7.2.6 size()	40
9.7.2.7 width()	41
9.8 mynteye::HardwareVersion Class Reference	41
9.8.1 Detailed Description	41
9.9 mynteye::ImgData Struct Reference	41
9.9.1 Detailed Description	41
9.10 mynteye::device::ImgParams Struct Reference	41
9.11 mynteye::ImuData Struct Reference	41
9.11.1 Detailed Description	42
9.11.2 Member Data Documentation	42
9.11.2.1 accel	42
9.11.2.2 flag	42
9.11.2.3 gyro	42
9.12 mynteye::ImuIntrinsics Struct Reference	43
9.12.1 Detailed Description	43
9.12.2 Member Data Documentation	43
9.12.2.1 scale	43
9.13 mynteye::device::ImuParams Struct Reference	43
9.14 mynteye::IntrinsicsBase Struct Reference	43
9.15 mynteye::IntrinsicsEquidistant Struct Reference	44
9.15.1 Detailed Description	44
9.16 mynteye::IntrinsicsPinhole Struct Reference	44
9.16.1 Detailed Description	45
9.16.2 Member Data Documentation	45
9.16.2.1 model	45
9.17 mynteye::device::MotionData Struct Reference	45
9.17.1 Detailed Description	45
9.17.2 Member Data Documentation	46
9.17.2.1 imu	46
9.18 mynteye::api::MotionData Struct Reference	46
9.18.1 Detailed Description	46
9.18.2 Member Data Documentation	46
9.18.2.1 imu	46
9.19 mynteye::MotionIntrinsics Struct Reference	46
9.19.1 Detailed Description	47
9.20 mynteye::Object Struct Reference	47
9.20.1 Detailed Description	47
9.21 mynteye::ObjMat Struct Reference	47
9.21.1 Detailed Description	48

9.22 mynteye::ObjMat2 Struct Reference	48
9.22.1 Detailed Description	48
9.23 mynteye::OptionInfo Struct Reference	48
9.23.1 Detailed Description	49
9.24 mynteye::Plugin Class Reference	49
9.24.1 Detailed Description	49
9.24.2 Member Function Documentation	49
9.24.2.1 OnCreate()	49
9.24.2.2 OnDepthProcess()	50
9.24.2.3 OnDisparityNormalizedProcess()	50
9.24.2.4 OnDisparityProcess()	50
9.24.2.5 OnPointsProcess()	51
9.24.2.6 OnRectifyProcess()	51
9.25 mynteye::Resolution Struct Reference	52
9.25.1 Detailed Description	52
9.26 mynteye::api::StreamData Struct Reference	52
9.26.1 Detailed Description	52
9.26.2 Member Data Documentation	52
9.26.2.1 frame	52
9.26.2.2 frame_id	53
9.26.2.3 frame_raw	53
9.26.2.4 img	53
9.27 mynteye::device::StreamData Struct Reference	53
9.27.1 Detailed Description	53
9.27.2 Member Data Documentation	53
9.27.2.1 frame	54
9.27.2.2 frame_id	54
9.27.2.3 img	54
9.28 mynteye::StreamRequest Struct Reference	54
9.28.1 Detailed Description	54
9.29 mynteye::strings_error Class Reference	55
9.29.1 Detailed Description	55
9.30 mynteye::Type Class Reference	55
9.30.1 Detailed Description	55
9.31 mynteye::Version Class Reference	55
9.31.1 Detailed Description	55

# 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
Auxiliary chip version	auxiliary_↔ chip_version	-	×	✓ Get	2	major,minor
isp version	isp_version	-	×	✓ Get	2	major,minor

## 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}
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	Checksum
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

# Deprecated List

**Member [mynteye::API::GetIntrinsics](#) (const Stream &stream) const**

Get the intrinsics (pinhole) of stream.

**Member [mynteye::Device::GetLatestStreamData](#) (const Stream &stream)**

Replaced by [GetStreamData\(const Stream &stream\)](#)

**Member [mynteye::IntrinsicsPinhole::model](#)**

Replaced by [calib\\_model\\_](#).



# Chapter 5

## Module Index

### 5.1 Modules

Here is a list of all modules:

Utilities . . . . .	19
Enumerations . . . . .	21
Intrinsics & Extrinsics . . . . .	27
Datatypes . . . . .	28



# Chapter 6

## Hierarchical Index

### 6.1 Class Hierarchy

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

mynteye::API	29
mynteye::AsyncCallback< Data >	33
mynteye::Context	33
mynteye::Device	34
mynteye::DeviceInfo	38
mynteye::Extrinsics	38
mynteye::device::Frame	39
mynteye::ImgData	41
mynteye::device::ImgParams	41
mynteye::ImuData	41
mynteye::ImuIntrinsics	43
mynteye::device::ImuParams	43
mynteye::IntrinsicsBase	43
mynteye::IntrinsicsEquidistant	44
mynteye::IntrinsicsPinhole	44
mynteye::device::MotionData	45
mynteye::api::MotionData	46
mynteye::MotionIntrinsics	46
mynteye::Object	47
mynteye::ObjMat	47
mynteye::ObjMat2	48
mynteye::OptionInfo	48
mynteye::Plugin	49
mynteye::Resolution	52
runtime_error	
mynteye::strings_error	55
mynteye::api::StreamData	52
mynteye::device::StreamData	53
mynteye::StreamRequest	54
mynteye::Type	55
mynteye::Version	55
mynteye::HardwareVersion	41



# Chapter 7

## Class Index

### 7.1 Class List

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

<a href="#">mynteye::API</a>	To communicate with MYNT® EYE device	29
<a href="#">mynteye::AsyncCallback&lt; Data &gt;</a>		33
<a href="#">mynteye::Context</a>	The context about devices	33
<a href="#">mynteye::Device</a>	To communicate with MYNT® EYE device	34
<a href="#">mynteye::DeviceInfo</a>	Device information	38
<a href="#">mynteye::Extrinsics</a>	Extrinsics, represent how the different datas are connected	38
<a href="#">mynteye::device::Frame</a>	Frame with raw data	39
<a href="#">mynteye::HardwareVersion</a>	Hardware version	41
<a href="#">mynteye::ImgData</a>	Image data	41
<a href="#">mynteye::device::ImgParams</a>		41
<a href="#">mynteye::ImuData</a>	IMU data	41
<a href="#">mynteye::ImuIntrinsics</a>	IMU intrinsics: scale, drift and variances	43
<a href="#">mynteye::device::ImuParams</a>		43
<a href="#">mynteye::IntrinsicsBase</a>		43
<a href="#">mynteye::IntrinsicsEquidistant</a>	Stream intrinsics (Equidistant: KANNALA_BRANDT)	44
<a href="#">mynteye::IntrinsicsPinhole</a>	Stream intrinsics (Pinhole)	44
<a href="#">mynteye::device::MotionData</a>	Device motion data	45
<a href="#">mynteye::api::MotionData</a>	API motion data	46
<a href="#">mynteye::MotionIntrinsics</a>	Motion intrinsics, including accelerometer and gyroscope	46
<a href="#">mynteye::Object</a>	Input & output object	47

<a href="#">mynteye::ObjMat</a>	
Input & output object of one cv::Mat	47
<a href="#">mynteye::ObjMat2</a>	
Input & output object of two cv::Mat	48
<a href="#">mynteye::OptionInfo</a>	
Option info	48
<a href="#">mynteye::Plugin</a>	
The plugin which could implement processing by yourself	49
<a href="#">mynteye::Resolution</a>	
Resolution	52
<a href="#">mynteye::api::StreamData</a>	
API stream data	52
<a href="#">mynteye::device::StreamData</a>	
Device stream data	53
<a href="#">mynteye::StreamRequest</a>	
Stream request	54
<a href="#">mynteye::strings_error</a>	
The strings error	55
<a href="#">mynteye::Type</a>	
Type	55
<a href="#">mynteye::Version</a>	
Version	55



# Chapter 8

## Module Documentation

### 8.1 Utilities

#### Functions

- MYNTEYE\_API `std::shared_ptr< Device > mynteye::device::select ()`  
*Detecting MYNT EYE devices and prompt user to select one.*
- MYNTEYE\_API `MYNTEYE_NAMESPACE::StreamRequest mynteye::device::select_request (const std::shared_ptr< Device > &device, bool *ok)`  
*List stream requests 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.*
- MYNTEYE\_API `std::string mynteye::utils::get_sdk_root_dir ()`  
*Get sdk root dir.*
- MYNTEYE\_API `std::string mynteye::utils::get_sdk_install_dir ()`  
*Get sdk install dir.*

#### 8.1.1 Detailed Description

#### 8.1.2 Function Documentation

##### 8.1.2.1 `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.

**8.1.2.2 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.

**8.1.2.3 select\_request()**

```
MYNTEYE_API MYNTEYE_NAMESPACE::StreamRequest mynteye::device::select_request (
    const std::shared_ptr< Device > & device,
    bool * ok )
```

List stream requests and prompt user to select one.

**Returns**

the selected request.

## 8.2 Enumerations

Public enumeration types.

### Enumerations

- enum `mynteye::Model` : `std::uint8_t` { `mynteye::Model::STANDARD`, `mynteye::Model::STANDARD2`, `mynteye::Model::STANDARD210A`, `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::STEREO_COLOR`, `mynteye::Capabilities::COLOR`, `mynteye::Capabilities::DEPTH`, `mynteye::Capabilities::POINTS`, `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::MIN_EXPOSURE_TIME`, `mynteye::Option::DESIRED_BRIGHTNESS`, `mynteye::Option::IR_CONTROL`, `mynteye::Option::HDR_MODE`, `mynteye::Option::ACCELEROMETER_RANGE`, `mynteye::Option::GYROSCOPE_RANGE`, `mynteye::Option::ACCELEROMET`, `mynteye::Option::GYROSCOPE_LOW_PASS_FILTER`, `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::BGR888` =  $((\text{std::uint32\_t}('B') \mid (\text{std::uint32\_t}('G') \ll 8) \mid (\text{std::uint32\_t}('R') \ll 16) \mid (\text{std::uint32\_t}('3') \ll 24))$ , `mynteye::Format::RGB888` =  $((\text{std::uint32\_t}('R') \mid (\text{std::uint32\_t}('G') \ll 8) \mid (\text{std::uint32\_t}('B') \ll 16) \mid (\text{std::uint32\_t}('3') \ll 24))$ , `mynteye::Format::LAST` }

*Formats define how each stream can be encoded.*

## 8.2.1 Detailed Description

Public enumeration types.

## 8.2.2 Enumeration Type Documentation

### 8.2.2.1 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.

### 8.2.2.2 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.
STEREO_COLOR	Provide stereo color 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.

### 8.2.2.3 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.
BGR888	BGR 8:8:8, 24 bits per pixel.
RGB888	RGB 8:8:8, 24 bits per pixel.
LAST	Last guard.

## 8.2.2.4 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.

## 8.2.2.5 Model

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

[Device](#) model.

## Enumerator

STANDARD	Standard.
STANDARD2	Standard 2.
STANDARD210A	Standard 210a.
LAST	Last guard.

## 8.2.2.6 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: {10,15,20,25,30,35,40,45,50,55,60}, 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 of standard 1: [0,48], default: 48 range of standard 2: [0,255], default: 8
MAX_EXPOSURE_TIME	Max exposure time, valid if auto-exposure. range of standard 1: [0,240], default: 240 range of standard 2: [0,1000], default: 333
MIN_EXPOSURE_TIME	min exposure time, valid if auto-exposure range: [0,1000], default: 0
DESIRED_BRIGHTNESS	Desired brightness, valid if auto-exposure. range of standard 1: [0,255], default: 192 range of standard 2: [1,255], default: 122
IR_CONTROL	IR control. range: [0,160], default: 0
HDR_MODE	HDR mode. 0: 10-bit 1: 12-bit
ACCELEROMETER_RANGE	The range of accelerometer. value of standard 1: {4,8,16,32}, default: 8 value of standard 2: {6,12,24,48}, default: 12
GYROSCOPE_RANGE	The range of gyroscope. value of standard 1: {500,1000,2000,4000}, default: 1000 value of standard 2: {250,500,1000,2000,4000}, default: 1000
ACCELEROMETER_LOW_PASS_FILTER	The parameter of accelerometer low pass filter. values: {0,1,2}, default: 2
GYROSCOPE_LOW_PASS_FILTER	The parameter of gyroscope low pass filter. values: {23,64}, default: 64
ZERO_DRIFT_CALIBRATION	Zero drift calibration.
ERASE_CHIP	Erase chip.
LAST	Last guard.

## 8.2.2.7 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.

### 8.2.2.8 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.



## 8.3 Intrinsic & Extrinsic

Intrinsic and extrinsic properties.

### Classes

- struct `mynteye::IntrinsicPinhole`  
*Stream intrinsic (Pinhole)*
- struct `mynteye::IntrinsicEquidistant`  
*Stream intrinsic (Equidistant: KANNALA\_BRANDT)*
- 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.*

### Enumerations

- enum `mynteye::CalibrationModel` : `std::uint8_t { mynteye::CalibrationModel::PINHOLE = 0, mynteye::CalibrationModel::KANNALA_BRANDT = 1, mynteye::CalibrationModel::UNKNOWN }`  
*Camera calibration model.*

#### 8.3.1 Detailed Description

Intrinsic and extrinsic properties.

#### 8.3.2 Enumeration Type Documentation

##### 8.3.2.1 CalibrationModel

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

Camera calibration model.

##### Enumerator

PINHOLE	Pinhole.
KANNALA_BRANDT	Equidistant: KANNALA_BRANDT.
UNKNOWN	Unknow.

## 8.4 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::DeviceInfo](#)  
*Device information.*
- struct [mynteye::ImgData](#)  
*Image data.*
- struct [mynteye::ImuData](#)  
*IMU data.*
- struct [mynteye::OptionInfo](#)  
*Option info.*

### 8.4.1 Detailed Description

Public data types.

# Chapter 9

## Class Documentation

### 9.1 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.*
- [StreamRequest SelectStreamRequest](#) (bool \*ok) const  
*Log all stream requests and prompt user to select one.*
- 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.*
- const [StreamRequest](#) & [GetStreamRequest](#) (const [Capabilities](#) &capability) const  
*Get the config stream requests of the capability.*
- const std::vector< [StreamRequest](#) > & [GetStreamRequests](#) () const  
*Get all stream requests of the key stream capability.*

- void [ConfigStreamRequest](#) (const [StreamRequest](#) &request)  
*Config the stream request to the key stream capability.*
- const [StreamRequest](#) & [GetStreamRequest](#) () const  
*Get the config stream requests of the key stream capability.*
- std::shared\_ptr< [DeviceInfo](#) > [GetInfo](#) () const  
*Get the device info.*
- std::string [GetInfo](#) (const [Info](#) &info) const  
*Get the device info.*
- [IntrinsicsPinhole](#) [GetIntrinsics](#) (const [Stream](#) &stream) const
- template<typename T >  
T [GetIntrinsics](#) (const [Stream](#) &stream) const  
*Get the intrinsics of stream.*
- std::shared\_ptr< [IntrinsicsBase](#) > [GetIntrinsicsBase](#) (const [Stream](#) &stream) const  
*Get the intrinsics base 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](#) (int argc, char \*argv[])  
*Create the [API](#) instance.*
- static std::shared\_ptr< [API](#) > [Create](#) (int argc, char \*argv[], const std::shared\_ptr< [Device](#) > &device)  
*Create the [API](#) instance.*
- static std::shared\_ptr< [API](#) > [Create](#) (const std::shared\_ptr< [Device](#) > &device)  
*Create the [API](#) instance.*

### 9.1.1 Detailed Description

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

### 9.1.2 Member Typedef Documentation

#### 9.1.2.1 motion\_callback\_t

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

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

#### 9.1.2.2 stream\_callback\_t

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

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

### 9.1.3 Member Function Documentation

#### 9.1.3.1 Create() [1/3]

```
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.

**9.1.3.2 Create()** [2/3]

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

**9.1.3.3 Create()** [3/3]

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

Create the [API](#) instance.

**Parameters**

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

### Returns

the [API](#) instance.

#### 9.1.3.4 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.

#### 9.1.3.5 GetIntrinsics()

```
IntrinsicsPinhole mynteye::API::GetIntrinsics (
    const Stream & stream ) const
```

**Deprecated** Get the intrinsics (pinhole) of stream.

#### 9.1.3.6 GetStreamDatas()

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

Get the datas of stream.

### Note

default cache 4 datas at most.

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

## 9.3 mynteye::Context Class Reference

The context about devices.

## Public Member Functions

- `std::vector< std::shared_ptr< Device > > devices () const`

*Get all devices now.*

### 9.3.1 Detailed Description

The context about devices.

### 9.3.2 Member Function Documentation

#### 9.3.2.1 devices()

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

Get all devices now.

#### Returns

a vector of all devices.

## 9.4 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.*
- const [StreamRequest](#) & [GetStreamRequest](#) (const [Capabilities](#) &capability) const  
*Get the config stream requests of the capability.*
- const std::vector< [StreamRequest](#) > & [GetStreamRequests](#) () const  
*Get all stream requests of the key stream capability.*
- void [ConfigStreamRequest](#) (const [StreamRequest](#) &request)  
*Config the stream request to the key stream capability.*
- const [StreamRequest](#) & [GetStreamRequest](#) () const  
*Get the config stream requests of the key stream 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.*
- std::shared\_ptr< [IntrinsicsBase](#) > [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.*
- std::shared\_ptr< [IntrinsicsBase](#) > [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 std::shared\_ptr< [IntrinsicsBase](#) > &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.*
  - [device::StreamData GetStreamData](#) (const [Stream](#) &stream)  
*Get the latest data of stream.*
  - [device::StreamData GetLatestStreamData](#) (const [Stream](#) &stream)
  - std::vector< [device::StreamData](#) > [GetStreamDatas](#) (const [Stream](#) &stream)  
*Get the datas of stream.*
  - void [EnableMotionDatas](#) ()  
*Enable cache motion datas.*
  - void [EnableMotionDatas](#) (std::size\_t max\_size)  
*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.*

#### 9.4.1 Detailed Description

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

#### 9.4.2 Member Typedef Documentation

#### 9.4.2.1 motion\_callback\_t

```
using mynteye::Device::motion_callback_t = device::MotionCallback
```

The `device::MotionData` callback.

#### 9.4.2.2 stream\_callback\_t

```
using mynteye::Device::stream_callback_t = device::StreamCallback
```

The `device::StreamData` callback.

### 9.4.3 Member Function Documentation

#### 9.4.3.1 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.

#### 9.4.3.2 GetLatestStreamData()

```
device::StreamData mynteye::Device::GetLatestStreamData (
    const Stream & stream )
```

**Deprecated** Replaced by `GetStreamData(const Stream &stream)`

### 9.4.3.3 GetStreamDatas()

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

Get the datas of stream.

#### Note

default cache 4 datas at most.

## 9.5 mynteye::DeviceInfo Struct Reference

[Device](#) infomation.

### 9.5.1 Detailed Description

[Device](#) infomation.

## 9.6 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.*

### 9.6.1 Detailed Description

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

### 9.6.2 Member Function Documentation

### 9.6.2.1 Inverse()

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

Inverse this extrinsics.

#### Returns

the inversed extrinsics.

## 9.7 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.*

### 9.7.1 Detailed Description

[Frame](#) with raw data.

### 9.7.2 Member Function Documentation

### 9.7.2.1 clone()

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

Clone a new frame.

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

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

Get the data.

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

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

Get the const data.

### 9.7.2.4 format()

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

Get the format.

### 9.7.2.5 height()

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

Get the height.

### 9.7.2.6 size()

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

Get the size of data.

## 9.7.2.7 width()

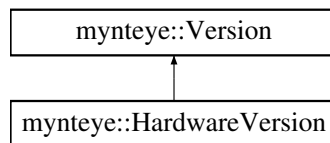
```
std::uint16_t mynteye::device::Frame::width ( ) const [inline]
```

Get the width.

## 9.8 mynteye::HardwareVersion Class Reference

Hardware version.

Inheritance diagram for mynteye::HardwareVersion:



### 9.8.1 Detailed Description

Hardware version.

## 9.9 mynteye::ImgData Struct Reference

Image data.

### Public Attributes

- `std::uint16_t frame_id`  
*Image frame id.*
- `std::uint64_t timestamp`  
*Image timestamp in 1us.*
- `std::uint16_t exposure_time`  
*Image exposure time, virtual value in [1, 480].*

### 9.9.1 Detailed Description

Image data.

## 9.10 mynteye::device::ImgParams Struct Reference

## 9.11 mynteye::ImuData Struct Reference

IMU data.

## Public Attributes

- `std::uint32_t frame_id`  
*IMU frame id.*
- `std::uint8_t flag`  
*IMU accel or gyro flag.*
- `std::uint64_t timestamp`  
*IMU timestamp in 1us.*
- `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.*

### 9.11.1 Detailed Description

IMU data.

### 9.11.2 Member Data Documentation

#### 9.11.2.1 accel

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

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

#### 9.11.2.2 flag

```
std::uint8_t mynteye::ImuData::flag
```

IMU accel or gyro flag.

0: accel and gyro are both valid 1: accel is valid 2: gyro is valid

#### 9.11.2.3 gyro

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

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



## 9.12 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.*

### 9.12.1 Detailed Description

IMU intrinsics: scale, drift and variances.

### 9.12.2 Member Data Documentation

#### 9.12.2.1 scale

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

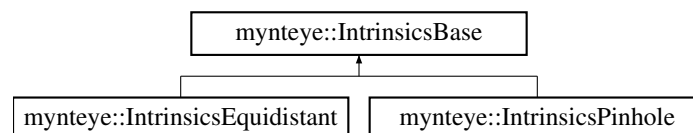
Scale matrix.

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

## 9.13 mynteye::device::ImuParams Struct Reference

## 9.14 mynteye::IntrinsicsBase Struct Reference

Inheritance diagram for mynteye::IntrinsicsBase:



### Public Member Functions

- [CalibrationModel calib\\_model](#) () const  
*The calibration model.*

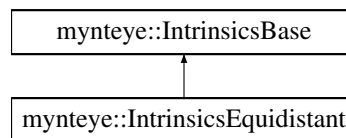
## Public Attributes

- `std::uint16_t width`  
*The width of the image in pixels.*
- `std::uint16_t height`  
*The height of the image in pixels.*

## 9.15 mynteye::IntrinsicsEquidistant Struct Reference

Stream intrinsics (Equidistant: KANNALA\_BRANDT)

Inheritance diagram for `mynteye::IntrinsicsEquidistant`:



## Public Attributes

- `double coeffs [8]`  
*The distortion coefficients:  $k_2, k_3, k_4, k_5, \mu, \nu, u_0, v_0$ .*

## Additional Inherited Members

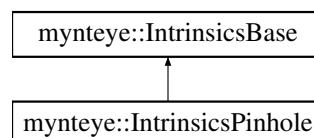
### 9.15.1 Detailed Description

Stream intrinsics (Equidistant: KANNALA\_BRANDT)

## 9.16 mynteye::IntrinsicsPinhole Struct Reference

Stream intrinsics (Pinhole)

Inheritance diagram for `mynteye::IntrinsicsPinhole`:



## Public Attributes

- 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](#)
- double [coeffs](#) [5]  
*The distortion coefficients:  $k_1, k_2, p_1, p_2, k_3$ .*

## Additional Inherited Members

### 9.16.1 Detailed Description

Stream intrinsics (Pinhole)

### 9.16.2 Member Data Documentation

#### 9.16.2.1 model

```
std::uint8_t mynteye::IntrinsicsPinhole::model
```

**Deprecated** Replaced by `calib_model_`.

The distortion model of the image

## 9.17 mynteye::device::MotionData Struct Reference

[Device](#) motion data.

## Public Attributes

- std::shared\_ptr< [ImuData](#) > [imu](#)  
*[ImuData](#).*

### 9.17.1 Detailed Description

[Device](#) motion data.

## 9.17.2 Member Data Documentation

### 9.17.2.1 imu

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

[ImuData](#).

## 9.18 mynteye::api::MotionData Struct Reference

[API](#) motion data.

### Public Attributes

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

### 9.18.1 Detailed Description

[API](#) motion data.

## 9.18.2 Member Data Documentation

### 9.18.2.1 imu

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

[ImuData](#).

## 9.19 mynteye::MotionIntrinsics Struct Reference

Motion intrinsics, including accelerometer and gyroscope.

### Public Attributes

- [ImuIntrinsics accel](#)  
*Accelerometer intrinsics.*
- [ImuIntrinsics gyro](#)  
*Gyroscope intrinsics.*

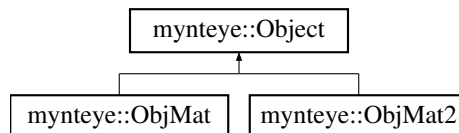
### 9.19.1 Detailed Description

Motion intrinsics, including accelerometer and gyroscope.

## 9.20 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.*

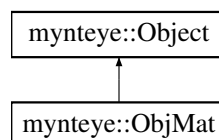
### 9.20.1 Detailed Description

Input & output object.

## 9.21 mynteye::ObjMat Struct Reference

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

Inheritance diagram for mynteye::ObjMat:



### Public Attributes

- `cv::Mat value`  
*The value.*
- `std::uint16_t id`  
*The id.*
- `std::shared_ptr< ImgData > data`  
*The data.*

## Additional Inherited Members

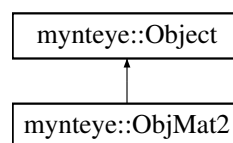
### 9.21.1 Detailed Description

Input & output object of one cv::Mat.

## 9.22 mynteye::ObjMat2 Struct Reference

Input & output object of two cv::Mat.

Inheritance diagram for mynteye::ObjMat2:



## Public Attributes

- cv::Mat [first](#)  
*The first value.*
- std::uint16\_t [first\\_id](#)  
*The first id.*
- std::shared\_ptr< [ImgData](#) > [first\\_data](#)  
*The first data.*
- cv::Mat [second](#)  
*The second value.*
- std::uint16\_t [second\\_id](#)  
*The second id.*
- std::shared\_ptr< [ImgData](#) > [second\\_data](#)  
*The second data.*

## Additional Inherited Members

### 9.22.1 Detailed Description

Input & output object of two cv::Mat.

## 9.23 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.*

### 9.23.1 Detailed Description

Option info.

## 9.24 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.*

### 9.24.1 Detailed Description

The plugin which could implement processing by yourself.

### 9.24.2 Member Function Documentation

#### 9.24.2.1 OnCreate()

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

Called when plugin created.

**Parameters**

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

**9.24.2.2 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.

**9.24.2.3 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.

**9.24.2.4 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.

**9.24.2.5 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.

**9.24.2.6 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.

## 9.25 mynteye::Resolution Struct Reference

[Resolution.](#)

### Public Attributes

- `std::uint16_t width`  
*Width.*
- `std::uint16_t height`  
*Height.*

### 9.25.1 Detailed Description

[Resolution.](#)

## 9.26 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.*
- `std::uint16_t frame_id`  
*Frame ID.*

### 9.26.1 Detailed Description

[API](#) stream data.

### 9.26.2 Member Data Documentation

#### 9.26.2.1 frame

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

Frame.

### 9.26.2.2 frame\_id

```
std::uint16_t mynteye::api::StreamData::frame_id
```

Frame ID.

### 9.26.2.3 frame\_raw

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

Raw frame.

### 9.26.2.4 img

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

[ImgData](#).

## 9.27 mynteye::device::StreamData Struct Reference

[Device](#) stream data.

### Public Attributes

- `std::shared_ptr< ImgData > img`  
[ImgData](#).
- `std::shared_ptr< Frame > frame`  
[Frame](#).
- `std::uint16_t frame_id`  
[Frame ID](#).

### 9.27.1 Detailed Description

[Device](#) stream data.

### 9.27.2 Member Data Documentation

### 9.27.2.1 frame

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

[Frame](#).

### 9.27.2.2 frame\_id

```
std::uint16_t mynteye::device::StreamData::frame_id
```

[Frame ID](#).

### 9.27.2.3 img

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

[ImgData](#).

## 9.28 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.*

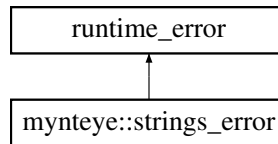
### 9.28.1 Detailed Description

Stream request.

## 9.29 mynteye::strings\_error Class Reference

The strings error.

Inheritance diagram for mynteye::strings\_error:



### 9.29.1 Detailed Description

The strings error.

## 9.30 mynteye::Type Class Reference

Type.

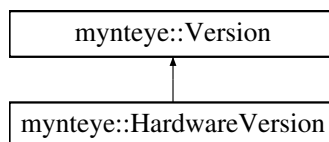
### 9.30.1 Detailed Description

Type.

## 9.31 mynteye::Version Class Reference

Version.

Inheritance diagram for mynteye::Version:



### 9.31.1 Detailed Description

Version.



# Index

- accel
  - mynteye::ImuData, [42](#)
- ACCELEROMETER\_LOW\_PASS\_FILTER
  - Enumerations, [25](#)
- ACCELEROMETER\_RANGE
  - Enumerations, [25](#)
- AddOns
  - Enumerations, [22](#)
- ALL
  - Enumerations, [25](#)
- BGR888
  - Enumerations, [23](#)
- BRIGHTNESS
  - Enumerations, [25](#)
- CalibrationModel
  - Intrinsics & Extrinsics, [27](#)
- Capabilities
  - Enumerations, [22](#)
- clone
  - mynteye::device::Frame, [39](#)
- COLOR
  - Enumerations, [22](#)
- CONTRAST
  - Enumerations, [25](#)
- Create
  - mynteye::API, [31](#), [32](#)
  - mynteye::Device, [37](#)
- data
  - mynteye::device::Frame, [40](#)
- Datatypes, [28](#)
- DEPTH
  - Enumerations, [22](#), [26](#)
- DESIRED\_BRIGHTNESS
  - Enumerations, [25](#)
- DEVICE\_NAME
  - Enumerations, [23](#)
- devices
  - mynteye::Context, [34](#)
- DISPARITY
  - Enumerations, [26](#)
- DISPARITY\_NORMALIZED
  - Enumerations, [26](#)
- EnableStreamData
  - mynteye::API, [33](#)
- Enumerations, [21](#)
  - ACCELEROMETER\_LOW\_PASS\_FILTER, [25](#)
  - ACCELEROMETER\_RANGE, [25](#)
  - AddOns, [22](#)
  - ALL, [25](#)
  - BGR888, [23](#)
  - BRIGHTNESS, [25](#)
  - Capabilities, [22](#)
  - COLOR, [22](#)
  - CONTRAST, [25](#)
  - DEPTH, [22](#), [26](#)
  - DESIRED\_BRIGHTNESS, [25](#)
  - DEVICE\_NAME, [23](#)
  - DISPARITY, [26](#)
  - DISPARITY\_NORMALIZED, [26](#)
  - ERASE\_CHIP, [25](#)
  - EXPOSURE\_MODE, [25](#)
  - FIRMWARE\_VERSION, [23](#)
  - FISHEYE, [22](#)
  - Format, [22](#)
  - FRAME\_RATE, [25](#)
  - GAIN, [25](#)
  - GREY, [23](#)
  - GYROSCOPE\_LOW\_PASS\_FILTER, [25](#)
  - GYROSCOPE\_RANGE, [25](#)
  - HARDWARE\_VERSION, [23](#)
  - HDR\_MODE, [25](#)
  - IMU, [22](#)
  - IMU\_FREQUENCY, [25](#)
  - IMU\_TYPE, [23](#)
  - Info, [23](#)
  - INFRARED, [22](#)
  - INFRARED2, [22](#)
  - IR\_CONTROL, [25](#)
  - LAST, [22](#), [23](#), [25](#), [26](#)
  - LEFT, [26](#)
  - LEFT\_RECTIFIED, [26](#)
  - LENS\_TYPE, [23](#)
  - MAX\_EXPOSURE\_TIME, [25](#)
  - MAX\_GAIN, [25](#)
  - MIN\_EXPOSURE\_TIME, [25](#)
  - Model, [23](#)
  - MOTION\_TRACKING, [25](#)
  - NOMINAL\_BASELINE, [23](#)
  - Option, [23](#)
  - POINTS, [22](#), [26](#)
  - RGB888, [23](#)
  - RIGHT, [26](#)
  - RIGHT\_RECTIFIED, [26](#)
  - SERIAL\_NUMBER, [23](#)
  - Source, [25](#)

- SPEC\_VERSION, [23](#)
- STANDARD, [23](#)
- STANDARD2, [23](#)
- STANDARD210A, [23](#)
- STEREO, [22](#)
- STEREO\_COLOR, [22](#)
- Stream, [26](#)
- VIDEO\_STREAMING, [25](#)
- YUYV, [23](#)
- ZERO\_DRIFT\_CALIBRATION, [25](#)
- ERASE\_CHIP
  - Enumerations, [25](#)
- EXPOSURE\_MODE
  - Enumerations, [25](#)
- FIRMWARE\_VERSION
  - Enumerations, [23](#)
- FISHEYE
  - Enumerations, [22](#)
- flag
  - mynteye::ImuData, [42](#)
- Format
  - Enumerations, [22](#)
- format
  - mynteye::device::Frame, [40](#)
- frame
  - mynteye::api::StreamData, [52](#)
  - mynteye::device::StreamData, [53](#)
- frame\_id
  - mynteye::api::StreamData, [52](#)
  - mynteye::device::StreamData, [54](#)
- FRAME\_RATE
  - Enumerations, [25](#)
- frame\_raw
  - mynteye::api::StreamData, [53](#)
- GAIN
  - Enumerations, [25](#)
- get\_real\_exposure\_time
  - Utilities, [19](#)
- GetIntrinsics
  - mynteye::API, [33](#)
- GetLatestStreamData
  - mynteye::Device, [37](#)
- GetStreamDatas
  - mynteye::API, [33](#)
  - mynteye::Device, [37](#)
- GREY
  - Enumerations, [23](#)
- gyro
  - mynteye::ImuData, [42](#)
- GYROSCOPE\_LOW\_PASS\_FILTER
  - Enumerations, [25](#)
- GYROSCOPE\_RANGE
  - Enumerations, [25](#)
- HARDWARE\_VERSION
  - Enumerations, [23](#)
- HDR\_MODE
  - Enumerations, [25](#)
- Enumerations, [25](#)
- height
  - mynteye::device::Frame, [40](#)
- img
  - mynteye::api::StreamData, [53](#)
  - mynteye::device::StreamData, [54](#)
- IMU
  - Enumerations, [22](#)
- imu
  - mynteye::api::MotionData, [46](#)
  - mynteye::device::MotionData, [46](#)
- IMU\_FREQUENCY
  - Enumerations, [25](#)
- IMU\_TYPE
  - Enumerations, [23](#)
- Info
  - Enumerations, [23](#)
- INFRARED
  - Enumerations, [22](#)
- INFRARED2
  - Enumerations, [22](#)
- Intrinsics & Extrinsics, [27](#)
  - CalibrationModel, [27](#)
  - KANNALA\_BRANDT, [27](#)
  - PINHOLE, [27](#)
  - UNKNOWN, [27](#)
- Inverse
  - mynteye::Extrinsics, [38](#)
- IR\_CONTROL
  - Enumerations, [25](#)
- KANNALA\_BRANDT
  - Intrinsics & Extrinsics, [27](#)
- LAST
  - Enumerations, [22](#), [23](#), [25](#), [26](#)
- LEFT
  - Enumerations, [26](#)
- LEFT\_RECTIFIED
  - Enumerations, [26](#)
- LENS\_TYPE
  - Enumerations, [23](#)
- MAX\_EXPOSURE\_TIME
  - Enumerations, [25](#)
- MAX\_GAIN
  - Enumerations, [25](#)
- MIN\_EXPOSURE\_TIME
  - Enumerations, [25](#)
- Model
  - Enumerations, [23](#)
- model
  - mynteye::IntrinsicsPinhole, [45](#)
- motion\_callback\_t
  - mynteye::API, [31](#)
  - mynteye::Device, [36](#)
- MOTION\_TRACKING
  - Enumerations, [25](#)



- mynteye::API, 29
  - Create, 31, 32
  - EnableStreamData, 33
  - GetIntrinsics, 33
  - GetStreamDatas, 33
  - motion\_callback\_t, 31
  - stream\_callback\_t, 31
- mynteye::api::MotionData, 46
  - imu, 46
- mynteye::api::StreamData, 52
  - frame, 52
  - frame\_id, 52
  - frame\_raw, 53
  - img, 53
- mynteye::AsyncCallback< Data >, 33
- mynteye::Context, 33
  - devices, 34
- mynteye::Device, 34
  - Create, 37
  - GetLatestStreamData, 37
  - GetStreamDatas, 37
  - motion\_callback\_t, 36
  - stream\_callback\_t, 37
- mynteye::device::Frame, 39
  - clone, 39
  - data, 40
  - format, 40
  - height, 40
  - size, 40
  - width, 40
- mynteye::device::ImgParams, 41
- mynteye::device::ImuParams, 43
- mynteye::device::MotionData, 45
  - imu, 46
- mynteye::device::StreamData, 53
  - frame, 53
  - frame\_id, 54
  - img, 54
- mynteye::DeviceInfo, 38
- mynteye::Extrinsics, 38
  - Inverse, 38
- mynteye::HardwareVersion, 41
- mynteye::ImgData, 41
- mynteye::ImuData, 41
  - accel, 42
  - flag, 42
  - gyro, 42
- mynteye::ImuIntrinsics, 43
  - scale, 43
- mynteye::IntrinsicsBase, 43
- mynteye::IntrinsicsEquidistant, 44
- mynteye::IntrinsicsPinhole, 44
  - model, 45
- mynteye::MotionIntrinsics, 46
- mynteye::Object, 47
- mynteye::ObjMat, 47
- mynteye::ObjMat2, 48
- mynteye::OptionInfo, 48
- mynteye::Plugin, 49
  - OnCreate, 49
  - OnDepthProcess, 50
  - OnDisparityNormalizedProcess, 50
  - OnDisparityProcess, 50
  - OnPointsProcess, 51
  - OnRectifyProcess, 51
- mynteye::Resolution, 52
- mynteye::StreamRequest, 54
- mynteye::strings\_error, 55
- mynteye::Type, 55
- mynteye::Version, 55
- NOMINAL\_BASELINE
  - Enumerations, 23
- OnCreate
  - mynteye::Plugin, 49
- OnDepthProcess
  - mynteye::Plugin, 50
- OnDisparityNormalizedProcess
  - mynteye::Plugin, 50
- OnDisparityProcess
  - mynteye::Plugin, 50
- OnPointsProcess
  - mynteye::Plugin, 51
- OnRectifyProcess
  - mynteye::Plugin, 51
- Option
  - Enumerations, 23
- PINHOLE
  - Intrinsics & Extrinsics, 27
- POINTS
  - Enumerations, 22, 26
- RGB888
  - Enumerations, 23
- RIGHT
  - Enumerations, 26
- RIGHT\_RECTIFIED
  - Enumerations, 26
- scale
  - mynteye::ImuIntrinsics, 43
- select
  - Utilities, 20
- select\_request
  - Utilities, 20
- SERIAL\_NUMBER
  - Enumerations, 23
- size
  - mynteye::device::Frame, 40
- Source
  - Enumerations, 25
- SPEC\_VERSION
  - Enumerations, 23
- STANDARD
  - Enumerations, 23

---

STANDARD2  
Enumerations, [23](#)

STANDARD210A  
Enumerations, [23](#)

STEREO  
Enumerations, [22](#)

STEREO\_COLOR  
Enumerations, [22](#)

Stream  
Enumerations, [26](#)

stream\_callback\_t  
mynteye::API, [31](#)  
mynteye::Device, [37](#)

UNKNOWN  
Intrinsics & Extrinsics, [27](#)

Utilities, [19](#)  
get\_real\_exposure\_time, [19](#)  
select, [20](#)  
select\_request, [20](#)

VIDEO\_STREAMING  
Enumerations, [25](#)

width  
mynteye::device::Frame, [40](#)

YUYV  
Enumerations, [23](#)

ZERO\_DRIFT\_CALIBRATION  
Enumerations, [25](#)