Camport2 Project

2

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 DepthEnhenceParameters Struct Reference

Public Attributes

- float **sigma_s**
- float **sigma_r**
- int **outlier_win_sz**
- float **outlier_rate**

3.1.1 Detailed Description

Definition at line 43 of file TYImageProc.h.

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

- TYImageProc.h

3.2 DepthSpeckleFilterParameters Struct Reference

Public Attributes

- int **max_speckle_size**
- int **max_speckle_diff**

3.2.1 Detailed Description

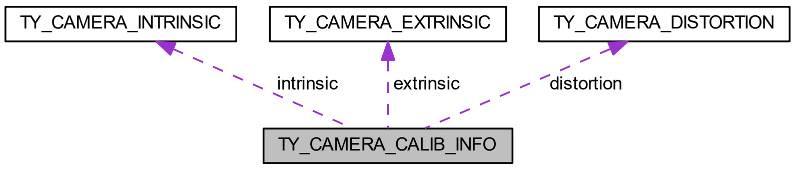
Definition at line 25 of file TYImageProc.h.

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

- TYImageProc.h

3.3 TY_CAMERA_CALIB_INFO Struct Reference

Collaboration diagram for TY_CAMERA_CALIB_INFO:



Public Attributes

- int32_t **intrinsicWidth**
- int32_t **intrinsicHeight**
- TY_CAMERA_INTRINSIC **intrinsic**
- TY_CAMERA_EXTRINSIC **extrinsic**
- TY_CAMERA_DISTORTION **distortion**

3.3.1 Detailed Description

Definition at line 497 of file TYApi.h.

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

- TYApi.h

3.4 TY_CAMERA_DISTORTION Struct Reference

camera distortion parameters

```
#include <TYApi.h>
```

Public Attributes

- float data [12]

  k1,k2,p1,p2,k3,k4,k5,k6,s1,s2,s3,s4

3.4.1 Detailed Description

camera distortion parameters

Definition at line 491 of file TYApi.h.

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

- TYApi.h

3.5 TY_CAMERA_EXTRINSIC Struct Reference

```
#include <TYApi.h>
```

Public Attributes

- float **data** [4 ∗4]

3.5.1 Detailed Description

[r11, r12, r13, t1, r21, r22, r23, t2, r31, r32, r33, t3, 0, 0, 0, 1]

Definition at line 485 of file TYApi.h.

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

- TYApi.h

3.6 TY_CAMERA_INTRINSIC Struct Reference

```
#include <TYApi.h>
```

Public Attributes

- float **data** [3 ∗3]

3.6.1 Detailed Description

[fx, 0, cx, 0, fy, cy, 0, 0, 1]

Definition at line 476 of file TYApi.h.

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

- TYApi.h

3.7 TY_CAMERA_STATISTICS Struct Reference

Public Attributes

- int32_t **packetReceived**
- int32_t **packetLost**
- int32_t **imageOutputed**
- int32_t **imageDropped**
- uint8_t **rsvd** [1024]

3.7.1 Detailed Description

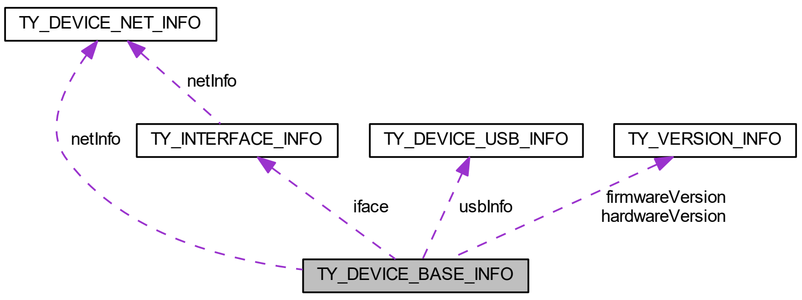
Definition at line 515 of file TYApi.h.

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

- TYApi.h

3.8 TY_DEVICE_BASE_INFO Struct Reference

Collaboration diagram for TY_DEVICE_BASE_INFO:



Public Attributes

- TY_INTERFACE_INFO **iface**
- char **id** [32]
- char **vendorName** [32]
- char **modelName** [32]
- TY_VERSION_INFO **hardwareVersion**
- TY_VERSION_INFO **firmwareVersion**
-

```
union {
   TY_DEVICE_NET_INFO netInfo
   TY_DEVICE_USB_INFO usbInfo
};
```

- char **reserved** [256]

3.8.1 Detailed Description

Definition at line 414 of file TYApi.h.

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

- TYApi.h

3.9 TY_DEVICE_NET_INFO Struct Reference

Public Attributes

- char **mac** [32]
- char **ip** [32]
- char **netmask** [32]
- char **gateway** [32]
- char **broadcast** [32]
- char **reserved** [96]

3.9.1 Detailed Description

Definition at line 388 of file TYApi.h.

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

- TYApi.h

3.10 TY_DEVICE_USB_INFO Struct Reference

Public Attributes

- int **bus**
- int **addr**
- char **reserved** [248]

3.10.1 Detailed Description

Definition at line 398 of file TYApi.h.

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

- TYApi.h

3.11 TY_ENUM_ENTRY Struct Reference

Public Attributes

- char **description** [64]
- int32_t **value**
- int32_t **reserved** [3]

3.11.1 Detailed Description

Definition at line 459 of file TYApi.h.

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

- TYApi.h

3.12 TY_EVENT_INFO Struct Reference

Public Attributes

- TY_EVENT **eventId**
- char **message** [124]

3.12.1 Detailed Description

Definition at line 553 of file TYApi.h.

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

- TYApi.h

3.13 TY_FEATURE_INFO Struct Reference

Public Attributes

- bool isValid

    true if feature exists, false otherwise
- TY_ACCESS_MODE accessMode

    feature access mode
- bool writableAtRun

    feature can be written while capturing
- char **reserved0** [1]
- TY_COMPONENT_ID **componentID**
- TY_FEATURE_ID **featureID**
- char **name** [32]
- int32_t bindComponentID

    component ID current feature bind to
- int32_t bindFeatureID

    feature ID current feature bind to
- char **reserved** [252]

3.13.1 Detailed Description

Definition at line 429 of file TYApi.h.

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

- TYApi.h

3.14 TY_FLOAT_RANGE Struct Reference

Public Attributes

- float **min**
- float **max**
- float **inc**
- float **reserved** [1]

3.14.1 Detailed Description

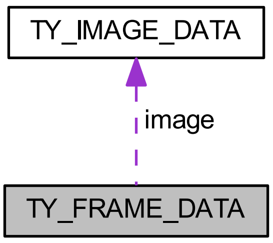
Definition at line 451 of file TYApi.h.

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

- TYApi.h

3.15 TY_FRAME_DATA Struct Reference

Collaboration diagram for TY_FRAME_DATA:



Public Attributes

- void ∗ userBuffer

    Pointer to user enqueued buffer, user should enqueue this buffer in the end of callback.
- int32_t bufferSize

    Size of userBuffer.
- int32_t validCount

    Number of valid data.
- int32_t reserved [6]

    Reserved.
- TY_IMAGE_DATA image [10]

    Buffer data, max to 10 images per frame, each buffer data could be an image or something else.

3.15.1 Detailed Description

Definition at line 543 of file TYApi.h.

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

- TYApi.h

3.16 TY_IMAGE_DATA Struct Reference

Public Attributes

- uint64_t timestamp

    Timestamp in microseconds.
- int32_t imageIndex

    image index used in trigger mode
- int32_t status

    Status of this buffer.
- int32_t componentID

    Where current data come from.
- int32_t size

    Buffer size.
- void ∗ buffer

    Pointer to data buffer.
- int32_t width

    Image width in pixels.
- int32_t height

    Image height in pixels.
- int32_t pixelFormat

    Pixel format, see TY_PIXEL_FORMAT_LIST.
- int32_t reserved [9]

    Reserved.

3.16.1 Detailed Description

Definition at line 528 of file TYApi.h.

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

- TYApi.h

3.17 TY_INT_RANGE Struct Reference

Public Attributes

- int32_t **min**
- int32_t **max**
- int32_t **inc**
- int32_t **reserved** [1]

3.17.1 Detailed Description

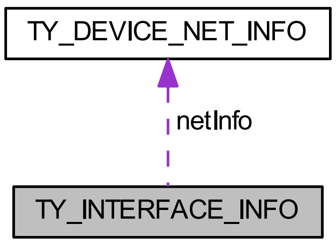
Definition at line 443 of file TYApi.h.

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

- TYApi.h

3.18 TY_INTERFACE_INFO Struct Reference

Collaboration diagram for TY_INTERFACE_INFO:



Public Attributes

- char **name** [32]
- char **id** [32]
- TY_INTERFACE_TYPE **type**
- char **reserved** [4]
- TY_DEVICE_NET_INFO **netInfo**

3.18.1 Detailed Description

Definition at line 405 of file TYApi.h.

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

- TYApi.h

3.19 TY_PIXEL_DESC Struct Reference

Public Attributes

- int16_t **x**
- int16_t **y**
- uint16_t **depth**
- uint16_t **rsvd**

3.19.1 Detailed Description

Definition at line 15 of file TYCoordinateMapper.h.

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

- TYCoordinateMapper.h

3.20 TY_TRIGGER_PARAM Struct Reference

Public Attributes

- TY_TRIGGER_MODE **mode**
- int8_t **fps**
- int8_t **rsvd**

3.20.1 Detailed Description

Definition at line 507 of file TYApi.h.

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

- TYApi.h

3.21 TY_VECT_3F Struct Reference

Public Attributes

- float **x**
- float **y**
- float **z**

3.21.1 Detailed Description

Definition at line 466 of file TYApi.h.

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

- TYApi.h

3.22 TY_VERSION_INFO Struct Reference

Public Attributes

- int32_t **major**
- int32_t **minor**
- int32_t **patch**
- int32_t **reserved**

3.22.1 Detailed Description

Definition at line 380 of file TYApi.h.

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

- TYApi.h

Chapter 4

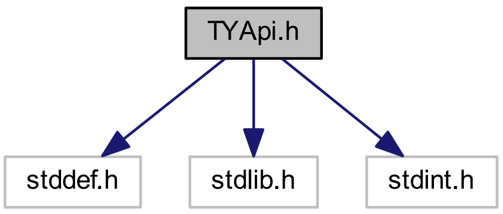
File Documentation

4.1 TYApi.h File Reference

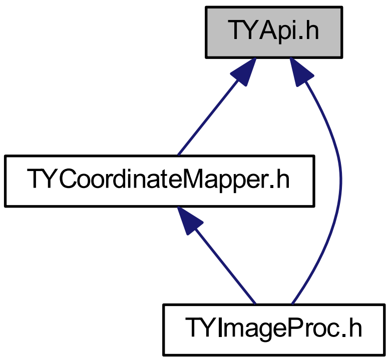
TYApi.h includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure
time, gain, working mode,etc.

```
#include <stddef.h>
#include <stdlib.h>
#include <stdint.h>
```

Include dependency graph for TYApi.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct TY_VERSION_INFO
- struct TY_DEVICE_NET_INFO
- struct TY_DEVICE_USB_INFO
- struct TY_INTERFACE_INFO
- struct TY_DEVICE_BASE_INFO
- struct TY_FEATURE_INFO
- struct TY_INT_RANGE
- struct TY_FLOAT_RANGE
- struct TY_ENUM_ENTRY
- struct TY_VECT_3F
- struct TY_CAMERA_INTRINSIC
- struct TY_CAMERA_EXTRINSIC
- struct TY_CAMERA_DISTORTION

    camera distortion parameters
- struct TY_CAMERA_CALIB_INFO
- struct TY_TRIGGER_PARAM
- struct TY_CAMERA_STATISTICS
- struct TY_IMAGE_DATA
- struct TY_FRAME_DATA
- struct TY_EVENT_INFO

Macros

- #define **_STDBOOL_H**
- #define **__bool_true_false_are_defined** 1
- #define **bool** _Bool
- #define **true** 1
- #define **false** 0
- #define **TY_DLLIMPORT** __attribute__((visibility("default")))
- #define **TY_DLLEXPORT** __attribute__((visibility("default")))

- #define **TY_STDC**
- #define **TY_CDEC**
- #define **TY_EXPORT** TY_DLLIMPORT
- #define **TY_EXTC**
- #define **TY_LIB_VERSION_MAJOR** 3
- #define **TY_LIB_VERSION_MINOR** 0
- #define **TY_LIB_VERSION_PATCH** 8
- #define **TY_DECLARE_IMAGE_MODE0**(pix, res) TY_IMAGE_MODE_##pix##_##res = TY_PIXEL_FOR↩
MAT_##pix | TY_RESOLUTION_MODE_##res
- #define **TY_DECLARE_IMAGE_MODE1**(pix)
- #define **TY_CAPI** TY_EXTC TY_EXPORT TY_STATUS TY_STDC

Typedefs

- typedef enum TY_STATUS_LIST **TY_STATUS_LIST**
- typedef int32_t **TY_STATUS**
- typedef enum TY_EVENT_LIST **TY_ENENT_LIST**
- typedef int32_t **TY_EVENT**
- typedef void ∗ **TY_INTERFACE_HANDLE**
- typedef void ∗ **TY_DEV_HANDLE**
- typedef enum TY_DEVICE_COMPONENT_LIST **TY_DEVICE_COMPONENT_LIST**
- typedef int32_t **TY_COMPONENT_ID**
- typedef enum TY_FEATURE_TYPE_LIST **TY_FEATURE_TYPE_LIST**
- typedef int32_t **TY_FEATURE_TYPE**
- typedef enum TY_FEATURE_ID_LIST **TY_FEATURE_ID_LIST**
- typedef int32_t **TY_FEATURE_ID**
- typedef enum TY_TRIGGER_ACTIVATION_LIST **TY_TRIGGER_ACTIVATION_LIST**
- typedef int32_t **TY_TRIGGER_ACTIVATION**
- typedef enum TY_INTERFACE_TYPE_LIST **TY_INTERFACE_TYPE_LIST**
- typedef int32_t **TY_INTERFACE_TYPE**
- typedef enum TY_ACCESS_MODE_LIST **TY_ACCESS_MODE_LIST**
- typedef int8_t **TY_ACCESS_MODE**
- typedef enum TY_PIXEL_BITS_LIST **TY_PIXEL_BITS_LIST**
- typedef enum TY_PIXEL_FORMAT_LIST **TY_PIXEL_FORMAT_LIST**
- typedef int32_t **TY_PIXEL_FORMAT**
- typedef enum TY_RESOLUTION_MODE_LIST **TY_RESOLUTION_MODE_LIST**
- typedef int32_t **TY_RESOLUTION_MODE**
- typedef enum TY_IMAGE_MODE_LIST **TY_IMAGE_MODE_LIST**
- typedef int32_t **TY_IMAGE_MODE**
- typedef enum TY_TRIGGER_MODE_LIST **TY_TRIGGER_MODE_LIST**
- typedef int16_t **TY_TRIGGER_MODE**
- typedef struct TY_VERSION_INFO **TY_VERSION_INFO**
- typedef struct TY_DEVICE_NET_INFO **TY_DEVICE_NET_INFO**
- typedef struct TY_DEVICE_USB_INFO **TY_DEVICE_USB_INFO**
- typedef struct TY_INTERFACE_INFO **TY_INTERFACE_INFO**
- typedef struct TY_DEVICE_BASE_INFO **TY_DEVICE_BASE_INFO**
- typedef struct TY_FEATURE_INFO **TY_FEATURE_INFO**
- typedef struct TY_INT_RANGE **TY_INT_RANGE**
- typedef struct TY_FLOAT_RANGE **TY_FLOAT_RANGE**
- typedef struct TY_ENUM_ENTRY **TY_ENUM_ENTRY**
- typedef struct TY_VECT_3F **TY_VECT_3F**
- typedef struct TY_CAMERA_INTRINSIC TY_CAMERA_INTRINSIC
- typedef struct TY_CAMERA_EXTRINSIC TY_CAMERA_EXTRINSIC
- typedef struct TY_CAMERA_DISTORTION TY_CAMERA_DISTORTION

camera distortion parameters

- typedef struct TY_CAMERA_CALIB_INFO **TY_CAMERA_CALIB_INFO**
- typedef struct TY_TRIGGER_PARAM **TY_TRIGGER_PARAM**
- typedef struct TY_CAMERA_STATISTICS **TY_CAMERA_STATISTICS**
- typedef struct TY_IMAGE_DATA **TY_IMAGE_DATA**
- typedef struct TY_FRAME_DATA **TY_FRAME_DATA**
- typedef struct TY_EVENT_INFO **TY_EVENT_INFO**
- typedef void(∗ **TY_EVENT_CALLBACK**) (TY_EVENT_INFO ∗, void ∗userdata)

Enumerations

- enum **TY_STATUS_LIST** {
  TY_STATUS_OK = 0, **TY_STATUS_ERROR** = -1001, **TY_STATUS_NOT_INITED** = -1002, **TY_STATUS↩**
  _NOT_IMPLEMENTED = -1003,
  TY_STATUS_NOT_PERMITTED = -1004, **TY_STATUS_DEVICE_ERROR** = -1005, **TY_STATUS_INVA↩**
  LID_PARAMETER = -1006, **TY_STATUS_INVALID_HANDLE** = -1007,
  TY_STATUS_INVALID_COMPONENT = -1008, **TY_STATUS_INVALID_FEATURE** = -1009, **TY_STATU↩**
  S_WRONG_TYPE = -1010, **TY_STATUS_WRONG_SIZE** = -1011,
  TY_STATUS_OUT_OF_MEMORY = -1012, **TY_STATUS_OUT_OF_RANGE** = -1013, **TY_STATUS_TIM↩**
  EOUT = -1014, **TY_STATUS_WRONG_MODE** = -1015,
  TY_STATUS_BUSY = -1016, **TY_STATUS_IDLE** = -1017, **TY_STATUS_NO_DATA** = -1018, **TY_STATU↩**
  S_NO_BUFFER = -1019,
  TY_STATUS_NULL_POINTER = -1020, **TY_STATUS_READONLY_FEATURE** = -1021, **TY_STATUS_I↩**
  NVALID_DESCRIPTOR = -1022, **TY_STATUS_INVALID_INTERFACE** = -1023,
  TY_STATUS_FIRMWARE_ERROR = -1024 }
- enum **TY_EVENT_LIST** { **TY_EVENT_DEVICE_OFFLINE** = -2001, **TY_EVENT_LICENSE_ERROR** = -2002, **TY_EVENT_FW_INIT_ERROR** = -2003 }
- enum TY_DEVICE_COMPONENT_LIST {
  TY_COMPONENT_DEVICE = 0x80000000, TY_COMPONENT_DEPTH_CAM = 0x00010000, TY_COMPONENT_IR_CAM_LE
  = 0x00040000, TY_COMPONENT_IR_CAM_RIGHT = 0x00080000,
  TY_COMPONENT_RGB_CAM_LEFT = 0x00100000, TY_COMPONENT_RGB_CAM_RIGHT = 0x00200000,
  TY_COMPONENT_LASER = 0x00400000, TY_COMPONENT_IMU = 0x00800000,
  TY_COMPONENT_BRIGHT_HISTO = 0x01000000, **TY_COMPONENT_RGB_CAM** = TY_COMPONENT↩
  _RGB_CAM_LEFT }
- enum **TY_FEATURE_TYPE_LIST** {
  TY_FEATURE_INT = 0x1000, **TY_FEATURE_FLOAT** = 0X2000, **TY_FEATURE_ENUM** = 0x3000, **TY_F↩**
  EATURE_BOOL = 0x4000,
  TY_FEATURE_STRING = 0x5000, **TY_FEATURE_BYTEARRAY** = 0x6000, **TY_FEATURE_STRUCT** = 0x7000 }
- enum TY_FEATURE_ID_LIST {
  TY_STRUCT_CAM_INTRINSIC = 0x0000 | TY_FEATURE_STRUCT, TY_STRUCT_EXTRINSIC_TO_LEFT_IR
  = 0x0001 | TY_FEATURE_STRUCT, TY_STRUCT_CAM_DISTORTION = 0x0006 | TY_FEATURE_STR↩
  UCT, TY_STRUCT_CAM_CALIB_DATA = 0x0007 | TY_FEATURE_STRUCT,
  TY_INT_PERSISTENT_IP = 0x0010 | TY_FEATURE_INT, **TY_INT_PERSISTENT_SUBMASK** = 0x0011 |
  TY_FEATURE_INT, **TY_INT_PERSISTENT_GATEWAY** = 0x0012 | TY_FEATURE_INT, **TY_BOOL_GVS↩**
  P_RESEND = 0x0013 | TY_FEATURE_BOOL,
  TY_INT_PACKET_DELAY = 0x0014 | TY_FEATURE_INT, **TY_INT_ACCEPTABLE_PERCENT** = 0x0015 |
  TY_FEATURE_INT, TY_STRUCT_CAM_STATISTICS = 0x00ff | TY_FEATURE_STRUCT, **TY_INT_WID↩**
  TH_MAX = 0x0100 | TY_FEATURE_INT,
  TY_INT_HEIGHT_MAX = 0x0101 | TY_FEATURE_INT, **TY_INT_OFFSET_X** = 0x0102 | TY_FEATURE_INT,
  TY_INT_OFFSET_Y = 0x0103 | TY_FEATURE_INT, **TY_INT_WIDTH** = 0x0104 | TY_FEATURE_INT,
  TY_INT_HEIGHT = 0x0105 | TY_FEATURE_INT, TY_ENUM_IMAGE_MODE = 0x0109 | TY_FEATURE_E↩
  NUM, TY_ENUM_TRIGGER_ACTIVATION = 0x0201 | TY_FEATURE_ENUM, TY_INT_FRAME_PER_TRIGGER
  = 0x0202 | TY_FEATURE_INT,
  TY_STRUCT_TRIGGER_PARAM = 0x0523 | TY_FEATURE_STRUCT, TY_BOOL_KEEP_ALIVE_ONOFF
  = 0x0203 | TY_FEATURE_BOOL, TY_INT_KEEP_ALIVE_TIMEOUT = 0x0204 | TY_FEATURE_INT,

TY_BOOL_CMOS_SYNC = 0x0205 | TY_FEATURE_BOOL,
TY_INT_TRIGGER_DELAY_US = 0x0206 | TY_FEATURE_INT, TY_BOOL_AUTO_EXPOSURE = 0x0300 | TY_FEATURE_BOOL, TY_INT_EXPOSURE_TIME = 0x0301 | TY_FEATURE_INT, TY_BOOL_AUTO_GAIN = 0x0302 | TY_FEATURE_BOOL,
TY_INT_GAIN = 0x0303 | TY_FEATURE_INT, TY_BOOL_AUTO_AWB = 0x0304 | TY_FEATURE_BOOL, TY_INT_LASER_POWER = 0x0500 | TY_FEATURE_INT, TY_BOOL_LASER_AUTO_CTRL = 0x0501 | TY_FEATURE_BOOL,
TY_BOOL_UNDISTORTION = 0x0510 | TY_FEATURE_BOOL, TY_BOOL_BRIGHTNESS_HISTOGRAM = 0x0511 | TY_FEATURE_BOOL, TY_BOOL_DEPTH_POSTPROC = 0x0512 | TY_FEATURE_BOOL, TY_INT_R_GAIN = 0x0520 | TY_FEATURE_INT,
TY_INT_G_GAIN = 0x0521 | TY_FEATURE_INT, TY_INT_B_GAIN = 0x0522 | TY_FEATURE_INT, TY_INT_ANALOG_GAIN = 0x0524 | TY_FEATURE_INT }

- enum **TY_TRIGGER_ACTIVATION_LIST** { **TY_TRIGGER_ACTIVATION_FALLINGEDGE** = 0, **TY_TRIG↩ GER_ACTIVATION_RISINGEDGE** = 1 }
- enum **TY_INTERFACE_TYPE_LIST** {
  TY_INTERFACE_UNKNOWN = 0, **TY_INTERFACE_RAW** = 1, **TY_INTERFACE_USB** = 2, **TY_INTERF↩ ACE_ETHERNET** = 4,
  TY_INTERFACE_IEEE80211 = 8, **TY_INTERFACE_ALL** = 0xffff }
- enum **TY_ACCESS_MODE_LIST** { **TY_ACCESS_READABLE** = 0x1, **TY_ACCESS_WRITABLE** = 0x2 }
- enum **TY_PIXEL_BITS_LIST** { **TY_PIXEL_8BIT** = 0x1 << 28, **TY_PIXEL_16BIT** = 0x2 << 28, **TY_PIX↩ EL_24BIT** = 0x3 << 28, **TY_PIXEL_32BIT** = 0x4 << 28 }
- enum TY_PIXEL_FORMAT_LIST {
  TY_PIXEL_FORMAT_UNDEFINED = 0, TY_PIXEL_FORMAT_MONO = (TY_PIXEL_8BIT | (0x0 << 24)), TY_PIXEL_FORMAT_BAYER8GB = (TY_PIXEL_8BIT | (0x1 << 24)), TY_PIXEL_FORMAT_DEPTH16 = (TY_PIXEL_16BIT | (0x0 << 24)),
  TY_PIXEL_FORMAT_YVYU = (TY_PIXEL_16BIT | (0x1 << 24)), TY_PIXEL_FORMAT_YUYV = (T↩ Y_PIXEL_16BIT | (0x2 << 24)), TY_PIXEL_FORMAT_RGB = (TY_PIXEL_24BIT | (0x0 << 24)), TY_PIXEL_FORMAT_BGR = (TY_PIXEL_24BIT | (0x1 << 24)),
  TY_PIXEL_FORMAT_JPEG = (TY_PIXEL_24BIT | (0x2 << 24)), TY_PIXEL_FORMAT_MJPG = (TY_PI↩ XEL_24BIT | (0x3 << 24)) }
- enum TY_RESOLUTION_MODE_LIST {
  TY_RESOLUTION_MODE_160x120 = (160<<12)+120, TY_RESOLUTION_MODE_320x240 = (320<<12)+240, TY_RESOLUTION_MODE_640x480 = (640<<12)+480, TY_RESOLUTION_MODE_1280x720 = (1280<<12)+720,
  TY_RESOLUTION_MODE_1280x960 = (1280<<12)+960, TY_RESOLUTION_MODE_2592x1944 = (2592<<12)+1944 }
- enum **TY_IMAGE_MODE_LIST**
- enum **TY_TRIGGER_MODE_LIST** { **TY_TRIGGER_MODE_OFF** = 0, **TY_TRIGGER_MODE_SLAVE** = 1, **TY_TRIGGER_MODE_M_SIG** = 2, **TY_TRIGGER_MODE_M_PER** = 3 }

Functions

- TY_EXTC TY_EXPORT const char ∗TY_STDC TYErrorString (TY_STATUS errorID)

  Get error information.
- TY_CAPI TYDeinitLib (void)

  Deinit this library.
- TY_CAPI TYLibVersion (TY_VERSION_INFO ∗version)

  Get current library version.
- TY_CAPI TYUpdateInterfaceList ()

  Update current interfaces.
- TY_CAPI TYGetInterfaceNumber (uint32_t ∗pNumIfaces)

  Get number of current interfaces.
- TY_CAPI TYGetInterfaceList (TY_INTERFACE_INFO ∗pIfaceInfos, uint32_t bufferCount, uint32_t ∗filled↩ Count)

  Get interface info list.
- TY_CAPI TYHasInterface (const char ∗ifaceID, bool ∗value)

Check if has interface.
- TY_CAPI TYOpenInterface (const char ∗ifaceID, TY_INTERFACE_HANDLE ∗outHandle)

  Open specified interface.
- TY_CAPI TYCloseInterface (TY_INTERFACE_HANDLE ifaceHandle)

  Close interface.
- TY_CAPI TYUpdateDeviceList (TY_INTERFACE_HANDLE ifaceHandle)

  Update current connected devices.
- TY_CAPI TYGetDeviceNumber (TY_INTERFACE_HANDLE ifaceHandle, uint32_t ∗deviceNumber)

  Get number of current connected devices.
- TY_CAPI TYGetDeviceList (TY_INTERFACE_HANDLE ifaceHandle, TY_DEVICE_BASE_INFO ∗device↩ Infos, uint32_t bufferCount, uint32_t ∗filledDeviceCount)

  Get device info list.
- TY_CAPI TYHasDevice (TY_INTERFACE_HANDLE ifaceHandle, const char ∗deviceID, bool ∗value)

  Check whether the interface has the specified device.
- TY_CAPI TYOpenDevice (TY_INTERFACE_HANDLE ifaceHandle, const char ∗deviceID, TY_DEV_HAN↩ DLE ∗outDeviceHandle)

  Open device by device ID.
- TY_CAPI TYOpenDeviceWithIP (TY_INTERFACE_HANDLE ifaceHandle, const char ∗IP, TY_DEV_HANDLE ∗deviceHandle)

  Open device by device IP, useful when device not listed.
- TY_CAPI TYGetDeviceInterface (TY_DEV_HANDLE hDevice, TY_INTERFACE_HANDLE ∗pIface)

  Get interface handle by device handle.
- TY_CAPI TYForceDeviceIP (TY_INTERFACE_HANDLE ifaceHandle, const char ∗MAC, const char ∗newIP, const char ∗newNetMask, const char ∗newGateway)

  Force device to use new IP address, useful when device use persistent IP and cannot be found.
- TY_CAPI TYCloseDevice (TY_DEV_HANDLE hDevice)

  Close device by device handle.
- TY_CAPI TYGetDeviceInfo (TY_DEV_HANDLE hDevice, TY_DEVICE_BASE_INFO ∗info)

  Get base info of the open device.
- TY_CAPI TYGetComponentIDs (TY_DEV_HANDLE hDevice, int32_t ∗componentIDs)

  Get all components IDs.
- TY_CAPI TYGetEnabledComponents (TY_DEV_HANDLE hDevice, int32_t ∗componentIDs)

  Get all enabled components IDs.
- TY_CAPI TYEnableComponents (TY_DEV_HANDLE hDevice, int32_t componentIDs)

  Enable components.
- TY_CAPI TYDisableComponents (TY_DEV_HANDLE hDevice, int32_t componentIDs)

  Disable components.
- TY_CAPI TYGetFrameBufferSize (TY_DEV_HANDLE hDevice, uint32_t ∗bufferSize)

  Get total buffer size of one frame in current configuration.
- TY_CAPI TYEnqueueBuffer (TY_DEV_HANDLE hDevice, void ∗buffer, uint32_t bufferSize)

  Enqueue a user allocated buffer.
- TY_CAPI TYClearBufferQueue (TY_DEV_HANDLE hDevice)

  Clear the internal buffer queue, so that user can release all the buffer.
- TY_CAPI TYStartCapture (TY_DEV_HANDLE hDevice)

  Start capture.
- TY_CAPI TYStopCapture (TY_DEV_HANDLE hDevice)

  Stop capture.
- TY_CAPI TYSendSoftTrigger (TY_DEV_HANDLE hDevice)

  Send a software trigger when device works in trigger mode.
- TY_CAPI TYRegisterEventCallback (TY_DEV_HANDLE hDevice, TY_EVENT_CALLBACK callback, void ∗userdata)

Register device status callback. Register NULL to clean callback.
- TY_CAPI TYFetchFrame (TY_DEV_HANDLE hDevice, TY_FRAME_DATA ∗frame, int32_t timeout)

  Fetch one frame.
- TY_CAPI TYHasFeature (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATUR↩ E_ID featureID, bool ∗value)

  Get whether has feature.
- TY_CAPI TYGetFeatureInfo (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEAT↩ URE_ID featureID, TY_FEATURE_INFO ∗featureInfo)

  Get feature info.
- TY_CAPI TYGetIntRange (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATU↩ RE_ID featureID, TY_INT_RANGE ∗intRange)

  Get value range of integer feature.
- TY_CAPI TYGetInt (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, int32_t ∗value)

  Get value of integer feature.
- TY_CAPI TYSetInt (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, int32_t value)

  Set value of integer feature.
- TY_CAPI TYGetFloatRange (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEA↩ TURE_ID featureID, TY_FLOAT_RANGE ∗floatRange)

  Get value range of float feature.
- TY_CAPI TYGetFloat (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, float ∗value)

  Get value of float feature.
- TY_CAPI TYSetFloat (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, float value)

  Set value of float feature.
- TY_CAPI TYGetEnumEntryCount (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_↩ FEATURE_ID featureID, uint32_t ∗entryCount)

  Get number of enum entries.
- TY_CAPI TYGetEnumEntryInfo (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_F↩ EATURE_ID featureID, TY_ENUM_ENTRY ∗entries, uint32_t entryCount, uint32_t ∗filledEntryCount)

  Get list of enum entries.
- TY_CAPI TYGetEnum (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, int32_t ∗value)

  Get current value of enum feature.
- TY_CAPI TYSetEnum (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, int32_t value)

  Set value of enum feature.
- TY_CAPI TYGetBool (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, bool ∗value)

  Get value of bool feature.
- TY_CAPI TYSetBool (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, bool value)

  Set value of bool feature.
- TY_CAPI TYGetStringLength (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEA↩ TURE_ID featureID, uint32_t ∗size)

  Get internal buffer size of string feature.
- TY_CAPI TYGetString (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, char ∗buffer, uint32_t bufferSize)

  Get value of string feature.
- TY_CAPI TYSetString (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, const char ∗buffer)

Set value of string feature.
- TY_CAPI TYGetStruct (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, void ∗pStruct, uint32_t structSize)

  Get value of struct.
- TY_CAPI TYSetStruct (TY_DEV_HANDLE hDevice, TY_COMPONENT_ID componentID, TY_FEATURE_ID featureID, void ∗pStruct, uint32_t structSize)

  Set value of struct.
- TY_CAPI **_TYInitLib** (void)

4.1.1 Detailed Description

TYApi.h includes camera control and data receiving interface, which supports configuration for image resolution, frame rate, exposure
time, gain, working mode,etc.

CHANGES compare to V2:

1. New Interface Layer Add this layer to specify local network interface to open network camera, solving the problem that someone wants to connect to a network camera with ethernet rather than WIFI. Users have to call interface APIs before openning devices.
2. New Image Processing Library The new library which has header file TYImageProc.h collects all image processing functions we provided.
3. New Coordinate Mapper New TYCoordinateMapper.h handles various convertions, including depth <-> point3D, point3D <-> point3D.
4. Components: Removed Point3D component(TY_COMPONENT_POINT3D). Point3D is a virtual component in V2, and the points are calculated from depth image. We put the calculation outside tycam library to increase flexibility.
5. Features: Removed TY_BOOL_TRIGGER_MODE , covered by TY_STRUCT_TRIGGER_PARAM Added TY_STRUCT_CAM_CALIB_DATA , for easy use in image processing library TY_INT_IMAGE_MODE , covered by new added TY_ENUM_IMAGE_MODE Modified TY_ENUM_IMAGE_MODE , means resolution mode in V2, combind resolution and pixel format in V3 Added some network camera's feature, such as TY_INT_PERSISTENT_IP, TY_INT_PERSISTENT_SUBMASK, TY_INT_PACKET_DELAY, etc.

Copyright(C)2016-2018 Percipio All Rights Reserved

4.1.2 Macro Definition Documentation

4.1.2.1 TY_DECLARE_IMAGE_MODE1

```
#define TY_DECLARE_IMAGE_MODE1(
              pix )
```

Value:

```
TY_DECLARE_IMAGE_MODE0(pix, 160x120), \
            TY_DECLARE_IMAGE_MODE0(pix, 320x240), \
            TY_DECLARE_IMAGE_MODE0(pix, 640x480), \
            TY_DECLARE_IMAGE_MODE0(pix, 1280x960), \
            TY_DECLARE_IMAGE_MODE0(pix, 2592x1944),
```

Definition at line 347 of file TYApi.h.

4.1.3 Typedef Documentation

4.1.3.1 TY_CAMERA_EXTRINSIC

```
typedef struct TY_CAMERA_EXTRINSIC TY_CAMERA_EXTRINSIC
```

[r11, r12, r13, t1, r21, r22, r23, t2, r31, r32, r33, t3, 0, 0, 0, 1]

4.1.3.2 TY_CAMERA_INTRINSIC

```
typedef struct TY_CAMERA_INTRINSIC TY_CAMERA_INTRINSIC
```

[fx, 0, cx, 0, fy, cy, 0, 0, 1]

4.1.4 Enumeration Type Documentation

4.1.4.1 TY_DEVICE_COMPONENT_LIST

```
enum TY_DEVICE_COMPONENT_LIST
```

Enumerator

TY_COMPONENT_DEVICE	Abstract component stands for whole device, always enabled.
TY_COMPONENT_DEPTH_CAM	Depth camera.
TY_COMPONENT_IR_CAM_LEFT	Left IR camera.
TY_COMPONENT_IR_CAM_RIGHT	Right IR camera.
TY_COMPONENT_RGB_CAM_LEFT	Left RGB camera.
TY_COMPONENT_RGB_CAM_RIGHT	Right RGB camera.
TY_COMPONENT_LASER	Laser.
TY_COMPONENT_IMU	Inertial Measurement Unit.
TY_COMPONENT_BRIGHT_HISTO	virtual component for brightness histogram of ir

Definition at line 193 of file TYApi.h.

4.1.4.2 TY_FEATURE_ID_LIST

```
enum TY_FEATURE_ID_LIST
```

Enumerator

TY_STRUCT_CAM_INTRINSIC	see TY_CAMERA_INTRINSIC
TY_STRUCT_EXTRINSIC_TO_LEFT_IR	extrinsic from current component to left IR, see TY_CAMERA_EXTRINSIC
TY_STRUCT_CAM_DISTORTION	see TY_CAMERA_DISTORTION
TY_STRUCT_CAM_CALIB_DATA	see TY_CAMERA_CALIB_INFO
TY_INT_PACKET_DELAY	microseconds
TY_STRUCT_CAM_STATISTICS	statistical information, see TY_CAMERA_STATISTICS
TY_ENUM_IMAGE_MODE	Resolution-PixelFromat mode, see TY_IMAGE_MODE_LIST.
TY_ENUM_TRIGGER_ACTIVATION	Trigger activation, see TY_TRIGGER_ACTIVATION_LIST.
TY_INT_FRAME_PER_TRIGGER	Number of frames captured per trigger.
TY_STRUCT_TRIGGER_PARAM	param of trigger, see TY_TRIGGER_PARAM
TY_BOOL_KEEP_ALIVE_ONOFF	Keep Alive switch.
TY_INT_KEEP_ALIVE_TIMEOUT	Keep Alive timeout.
TY_BOOL_CMOS_SYNC	Cmos sync switch.
TY_INT_TRIGGER_DELAY_US	Trigger delay time, in microseconds.
TY_BOOL_AUTO_EXPOSURE	Auto exposure switch.
TY_INT_EXPOSURE_TIME	Exposure time in percentage.
TY_BOOL_AUTO_GAIN	Auto gain switch.
TY_INT_GAIN	Gain.
TY_BOOL_AUTO_AWB	Auto white balance.
TY_INT_LASER_POWER	Laser power level.
TY_BOOL_LASER_AUTO_CTRL	Laser auto ctrl.
TY_BOOL_UNDISTORTION	Output undistorted image.
TY_BOOL_BRIGHTNESS_HISTOGRAM	Output bright histogram.
TY_BOOL_DEPTH_POSTPROC	Do depth image postproc.
TY_INT_R_GAIN	Gain of R channel.
TY_INT_G_GAIN	Gain of G channel.
TY_INT_B_GAIN	Gain of B channel.
TY_INT_ANALOG_GAIN	Analog gain.

Definition at line 226 of file TYApi.h.

4.1.4.3 TY_PIXEL_FORMAT_LIST

```
enum TY_PIXEL_FORMAT_LIST
```

Enumerator

TY_PIXEL_FORMAT_MONO	0x10000000
TY_PIXEL_FORMAT_BAYER8GB	0x11000000
TY_PIXEL_FORMAT_DEPTH16	0x20000000
TY_PIXEL_FORMAT_YVYU	0x21000000, yvyu422
TY_PIXEL_FORMAT_YUYV	0x22000000, yuyv422
TY_PIXEL_FORMAT_RGB	0x30000000
TY_PIXEL_FORMAT_BGR	0x31000000
TY_PIXEL_FORMAT_JPEG	0x32000000
TY_PIXEL_FORMAT_MJPG	0x33000000

Definition at line 318 of file TYApi.h.

4.1.4.4 TY_RESOLUTION_MODE_LIST

```
enum TY_RESOLUTION_MODE_LIST
```

Enumerator

TY_RESOLUTION_MODE_160x120	0x000a0078
TY_RESOLUTION_MODE_320x240	0x001400f0
TY_RESOLUTION_MODE_640x480	0x002801e0
TY_RESOLUTION_MODE_1280x720	0x005002d0
TY_RESOLUTION_MODE_1280x960	0x005003c0
TY_RESOLUTION_MODE_2592x1944	0x00a20798

Definition at line 333 of file TYApi.h.

4.1.5 Function Documentation

4.1.5.1 TYClearBufferQueue()

```
TY_CAPI TYClearBufferQueue (
            TY_DEV_HANDLE hDevice )
```

Clear the internal buffer queue, so that user can release all the buffer.

Parameters

in	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_BUSY	Device is capturing.

4.1.5.2 TYCloseDevice()

```
TY_CAPI TYCloseDevice (
            TY_DEV_HANDLE hDevice )
```

Close device by device handle.

Parameters

in	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_IDLE	Device has been closed.

4.1.5.3 TYCloseInterface()

```
TY_CAPI TYCloseInterface (
            TY_INTERFACE_HANDLE ifaceHandle )
```

Close interface.

Parameters

in	*ifaceHandle*	Interface to be closed.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Interface not found.

4.1.5.4 TYDeinitLib()

```
TY_CAPI TYDeinitLib (
            void )
```

Deinit this library.

Return values

TY_STATUS_OK	Succeed.

4.1.5.5 TYDisableComponents()

```
TY_CAPI TYDisableComponents (
            TY_DEV_HANDLE hDevice,
            int32_t componentIDs )
```

Disable components.

Parameters

in	*hDevice*	Device handle.
in	*componentIDs*	Components to be disabled.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Some components specified by componentIDs are invalid.
TY_STATUS_BUSY	Device is capturing.

4.1.5.6 TYEnableComponents()

```
TY_CAPI TYEnableComponents (
            TY_DEV_HANDLE hDevice,
            int32_t componentIDs )
```

Enable components.

Parameters

in	*hDevice*	Device handle.
in	*componentIDs*	Components to be enabled.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Some components specified by componentIDs are invalid.
TY_STATUS_BUSY	Device is capturing.

4.1.5.7 TYEnqueueBuffer()

```
TY_CAPI TYEnqueueBuffer (
            TY_DEV_HANDLE hDevice,
            void * buffer,
            uint32_t bufferSize )
```

Enqueue a user allocated buffer.

Parameters

in	*hDevice*	Device handle.
in	*buffer*	Buffer to be enqueued.
in	*bufferSize*	Size of the input buffer.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	buffer is NULL.
TY_STATUS_WRONG_SIZE	The input buffer is not large enough.

4.1.5.8 TYErrorString()

```
TY_EXTC TY_EXPORT const char *TY_STDC TYErrorString (
             TY_STATUS errorID )
```

Get error information.

Parameters

in	*errorID*	Error id.

Returns

Error string.

4.1.5.9 TYFetchFrame()

```
TY_CAPI TYFetchFrame (
             TY_DEV_HANDLE hDevice,
             TY_FRAME_DATA * frame,
             int32_t timeout )
```

Fetch one frame.

Parameters

in	*hDevice*	Device handle.
out	*frame*	Frame data to be filled.
in	*timeout*	Timeout in milliseconds. <0 for infinite.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	frame is NULL.
TY_STATUS_IDLE	Device capturing is not started.
TY_STATUS_WRONG_MODE	Callback has been registered, this function is disabled.
TY_STATUS_TIMEOUT	Timeout.

4.1.5.10 TYForceDeviceIP()

```
TY_CAPI TYForceDeviceIP (
            TY_INTERFACE_HANDLE ifaceHandle,
            const char * MAC,
            const char * newIP,
            const char * newNetMask,
            const char * newGateway )
```

Force device to use new IP address, useful when device use persistent IP and cannot be found.

Parameters

in	*ifaceHandle*	Interface handle.
in	*MAC*	Device MAC, should be "xx:xx:xx:xx:xx:xx".
in	*newIP*	New IP.
in	*newNetMask*	New subnet mask.
in	*newGateway*	New gateway.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Invalid interface handle.
TY_STATUS_WRONG_TYPE	Wrong interface type, should be network.
TY_STATUS_NULL_POINTER	MAC or newIP/newNetMask/newGateway is NULL.
TY_STATUS_INVALID_PARAMETER	MAC is not valid.
TY_STATUS_TIMEOUT	No device found.
TY_STATUS_DEVICE_ERROR	Set new IP failed.

4.1.5.11 TYGetBool()

```
TY_CAPI TYGetBool (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            bool * value )
```

Get value of bool feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*value*	Bool value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_BOOL.
TY_STATUS_NULL_POINTER	value is NULL.

4.1.5.12 TYGetComponentIDs()

```
TY_CAPI TYGetComponentIDs (
            TY_DEV_HANDLE hDevice,
            int32_t * componentIDs )
```

Get all components IDs.

Parameters

in	*hDevice*	Device handle.
out	*componentIDs*	All component IDs this device has.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	componentIDs is NULL.

4.1.5.13 TYGetDeviceInfo()

```
TY_CAPI TYGetDeviceInfo (
            TY_DEV_HANDLE hDevice,
            TY_DEVICE_BASE_INFO * info )
```

Get base info of the open device.

Parameters

in	*hDevice*	Device handle.
out	*info*	Base info out.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	componentIDs is NULL.

4.1.5.14 TYGetDeviceInterface()

```
TY_CAPI TYGetDeviceInterface (
            TY_DEV_HANDLE hDevice,
            TY_INTERFACE_HANDLE * pIface )
```

Get interface handle by device handle.

Parameters

in	*hDevice*	Device handle.
out	*pIface*	Interface handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	pIface is NULL.

4.1.5.15 TYGetDeviceList()

```
TY_CAPI TYGetDeviceList (
            TY_INTERFACE_HANDLE ifaceHandle,
            TY_DEVICE_BASE_INFO * deviceInfos,
            uint32_t bufferCount,
            uint32_t * filledDeviceCount )
```

Get device info list.

Parameters

in	*ifaceHandle*	Interface handle.
out	*deviceInfos*	Device info array to be filled.
in	*bufferCount*	Array size of deviceInfos.
out	*filledDeviceCount*	Number of filled TY_DEVICE_BASE_INFO.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Invalid interface handle.
TY_STATUS_NULL_POINTER	deviceInfos or filledDeviceCount is NULL.

4.1.5.16 TYGetDeviceNumber()

```
TY_CAPI TYGetDeviceNumber (
            TY_INTERFACE_HANDLE ifaceHandle,
            uint32_t * deviceNumber )
```

Get number of current connected devices.

Parameters

in	*ifaceHandle*	Interface handle.
out	*deviceNumber*	Number of connected devices.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Invalid interface handle.
TY_STATUS_NULL_POINTER	deviceNumber is NULL.

4.1.5.17 TYGetEnabledComponents()

```
TY_CAPI TYGetEnabledComponents (
            TY_DEV_HANDLE hDevice,
            int32_t * componentIDs )
```

Get all enabled components IDs.

Parameters

in	*hDevice*	Device handle.
out	*componentIDs*	Enabled component IDs.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.

Return values

TY_STATUS_NULL_POINTER	componentIDs is NULL.

4.1.5.18 TYGetEnum()

```
TY_CAPI TYGetEnum (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            int32_t * value )
```

Get current value of enum feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*value*	Enum value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_ENUM.
TY_STATUS_NULL_POINTER	value is NULL.

4.1.5.19 TYGetEnumEntryCount()

```
TY_CAPI TYGetEnumEntryCount (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            uint32_t * entryCount )
```

Get number of enum entries.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*entryCount*	Entry count.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_ENUM.
TY_STATUS_NULL_POINTER	entryCount is NULL.

4.1.5.20 TYGetEnumEntryInfo()

```
TY_CAPI TYGetEnumEntryInfo (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             TY_ENUM_ENTRY * entries,
             uint32_t entryCount,
             uint32_t * filledEntryCount )
```

Get list of enum entries.

Parameters

`in`	*hDevice*	Device handle.
`in`	*componentID*	Component ID.
`in`	*featureID*	Feature ID.
`out`	*entries*	Output entries.
`in`	*entryCount*	Array size of input parameter "entries".
`out`	*filledEntryCount*	Number of filled entries.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_ENUM.
TY_STATUS_NULL_POINTER	entries or filledEntryCount is NULL.

4.1.5.21 TYGetFeatureInfo()

```
TY_CAPI TYGetFeatureInfo (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             TY_FEATURE_INFO * featureInfo )
```

Get feature info.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*featureInfo*	Feature info.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_NULL_POINTER	featureInfo is NULL.

4.1.5.22 TYGetFloat()

```
TY_CAPI TYGetFloat (
              TY_DEV_HANDLE hDevice,
              TY_COMPONENT_ID componentID,
              TY_FEATURE_ID featureID,
              float * value )
```

Get value of float feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*value*	Float value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_FLOAT.
TY_STATUS_NULL_POINTER	value is NULL.

4.1.5.23 TYGetFloatRange()

```
TY_CAPI TYGetFloatRange (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            TY_FLOAT_RANGE * floatRange )
```

Get value range of float feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*floatRange*	Float range to be filled.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_FLOAT.
TY_STATUS_NULL_POINTER	floatRange is NULL.

4.1.5.24 TYGetFrameBufferSize()

```
TY_CAPI TYGetFrameBufferSize (
            TY_DEV_HANDLE hDevice,
            uint32_t * bufferSize )
```

Get total buffer size of one frame in current configuration.

Parameters

in	*hDevice*	Device handle.
out	*bufferSize*	Buffer size per frame.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_NULL_POINTER	bufferSize is NULL.

4.1.5.25 TYGetInt()

```
TY_CAPI TYGetInt (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             int32_t * value )
```

Get value of integer feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*value*	Integer value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_INT.
TY_STATUS_NULL_POINTER	value is NULL.

4.1.5.26 TYGetInterfaceList()

```
TY_CAPI TYGetInterfaceList (
             TY_INTERFACE_INFO * pIfaceInfos,
             uint32_t bufferCount,
             uint32_t * filledCount )
```

Get interface info list.

Parameters

out	*pIfaceInfos*	Array of interface infos to be filled.
in	*bufferCount*	Array size of interface infos.
out	*filledCount*	Number of filled TY_INTERFACE_INFO.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	pIfaceInfos or filledCount is NULL.

4.1.5.27 TYGetInterfaceNumber()

```
TY_CAPI TYGetInterfaceNumber (
             uint32_t * pNumIfaces )
```

Get number of current interfaces.

Parameters

out	*pNumIfaces*	Number of interfaces.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	deviceNumber is NULL.

4.1.5.28 TYGetIntRange()

```
TY_CAPI TYGetIntRange (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             TY_INT_RANGE * intRange )
```

Get value range of integer feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*intRange*	Integer range to be filled.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_INT.
TY_STATUS_NULL_POINTER	intRange is NULL.

4.1.5.29 TYGetString()

```
TY_CAPI TYGetString (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             char * buffer,
             uint32_t bufferSize )
```

Get value of string feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*buffer*	String buffer.
in	*bufferSize*	Size of buffer.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRING.
TY_STATUS_NULL_POINTER	buffer is NULL.

4.1.5.30 TYGetStringLength()

```
TY_CAPI TYGetStringLength (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             uint32_t * size )
```

Get internal buffer size of string feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*size*	String length including '\0'.

Return values

TY_STATUS_OK	Succeed.

Return values

TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRING.
TY_STATUS_NULL_POINTER	size is NULL.

4.1.5.31 TYGetStruct()

```
TY_CAPI TYGetStruct (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             void * pStruct,
             uint32_t structSize )
```

Get value of struct.

Parameters

`in`	*hDevice*	Device handle.
`in`	*componentID*	Component ID.
`in`	*featureID*	Feature ID.
`out`	*pStruct*	Pointer of struct.
`in`	*structSize*	Size of input buffer pStruct..

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRUCT.
TY_STATUS_NULL_POINTER	pStruct is NULL.
TY_STATUS_WRONG_SIZE	structSize incorrect.

4.1.5.32 TYHasDevice()

```
TY_CAPI TYHasDevice (
             TY_INTERFACE_HANDLE ifaceHandle,
             const char * deviceID,
             bool * value )
```

Check whether the interface has the specified device.

Parameters

in	*ifaceHandle*	Interface handle.
in	*deviceID*	Device ID string, can be get from TY_DEVICE_BASE_INFO.
out	*value*	True if the device exists.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Invalid interface handle.
TY_STATUS_NULL_POINTER	deviceID or value is NULL.

4.1.5.33 TYHasFeature()

```
TY_CAPI TYHasFeature (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            bool * value )
```

Get whether has feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
out	*value*	Whether has feature.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_NULL_POINTER	value is NULL.

4.1.5.34 TYHasInterface()

```
TY_CAPI TYHasInterface (
            const char * ifaceID,
            bool * value )
```

Check if has interface.

Parameters

in	*ifaceID*	Interface ID string, can be get from TY_INTERFACE_INFO.
out	*value*	True if the interface exists.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	ifaceID or outHandle is NULL.

4.1.5.35 TYLibVersion()

```
TY_CAPI TYLibVersion (
            TY_VERSION_INFO * version )
```

Get current library version.

Parameters

out	*version*	Version infomation to be filled.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NULL_POINTER	buffer is NULL.

4.1.5.36 TYOpenDevice()

```
TY_CAPI TYOpenDevice (
            TY_INTERFACE_HANDLE ifaceHandle,
            const char * deviceID,
            TY_DEV_HANDLE * outDeviceHandle )
```

Open device by device ID.

Parameters

in	*ifaceHandle*	Interface handle.
in	*deviceID*	Device ID string, can be get from TY_DEVICE_BASE_INFO.
out	*deviceHandle*	Handle of opened device.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Invalid interface handle.
TY_STATUS_NULL_POINTER	deviceID or deviceHandle is NULL.
TY_STATUS_INVALID_PARAMETER	Device not found.
TY_STATUS_BUSY	Device has been opened.
TY_STATUS_DEVICE_ERROR	Open device failed.

4.1.5.37 TYOpenDeviceWithIP()

```
TY_CAPI TYOpenDeviceWithIP (
            TY_INTERFACE_HANDLE ifaceHandle,
            const char * IP,
            TY_DEV_HANDLE * deviceHandle )
```

Open device by device IP, useful when device not listed.

Parameters

`in`	*ifaceHandle*	Interface handle.
`in`	*IP*	Device IP.
`out`	*deviceHandle*	Handle of opened device.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Invalid interface handle.
TY_STATUS_NULL_POINTER	IP or deviceHandle is NULL.
TY_STATUS_INVALID_PARAMETER	Device not found.
TY_STATUS_BUSY	Device has been opened, may occupied somewhere else.
TY_STATUS_DEVICE_ERROR	Open device failed.

4.1.5.38 TYOpenInterface()

```
TY_CAPI TYOpenInterface (
            const char * ifaceID,
            TY_INTERFACE_HANDLE * outHandle )
```

Open specified interface.

Parameters

in	*ifaceID*	Interface ID string, can be get from TY_INTERFACE_INFO.
out	*outHandle*	Handle of opened interface.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_NULL_POINTER	ifaceID or outHandle is NULL.
TY_STATUS_INVALID_INTERFACE	Interface not found.

4.1.5.39 TYRegisterEventCallback()

```
TY_CAPI TYRegisterEventCallback (
            TY_DEV_HANDLE hDevice,
            TY_EVENT_CALLBACK callback,
            void * userdata )
```

Register device status callback. Register NULL to clean callback.

Parameters

in	*hDevice*	Device handle.
in	*callback*	Callback function.
in	*userdata*	User private data.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_BUSY	Device is capturing.

4.1.5.40 TYSendSoftTrigger()

```
TY_CAPI TYSendSoftTrigger (
            TY_DEV_HANDLE hDevice )
```

Send a software trigger when device works in trigger mode.

Parameters

in	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_FEATURE	Not support soft trigger.
TY_STATUS_IDLE	Device has not started capture.
TY_STATUS_WRONG_MODE	Not in trigger mode.

4.1.5.41 TYSetBool()

```
TY_CAPI TYSetBool (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            bool value )
```

Set value of bool feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
in	*value*	Bool value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_BOOL.
TY_STATUS_BUSY	Device is capturing, the feature is locked.

4.1.5.42 TYSetEnum()

```
TY_CAPI TYSetEnum (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            int32_t value )
```

Set value of enum feature.

Parameters

`in`	*hDevice*	Device handle.
`in`	*componentID*	Component ID.
`in`	*featureID*	Feature ID.
`in`	*value*	Enum value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_ENUM.
TY_STATUS_INVALID_PARAMETER	value is invalid.
TY_STATUS_BUSY	Device is capturing, the feature is locked.

4.1.5.43 TYSetFloat()

```
TY_CAPI TYSetFloat (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             float value )
```

Set value of float feature.

Parameters

`in`	*hDevice*	Device handle.
`in`	*componentID*	Component ID.
`in`	*featureID*	Feature ID.
`in`	*value*	Float value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_FLOAT.
TY_STATUS_OUT_OF_RANGE	value is out of range.
TY_STATUS_BUSY	Device is capturing, the feature is locked.

4.1.5.44 TYSetInt()

```
TY_CAPI TYSetInt (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            int32_t value )
```

Set value of integer feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
in	*value*	Integer value.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_INT.
TY_STATUS_OUT_OF_RANGE	value is out of range.
TY_STATUS_BUSY	Device is capturing, the feature is locked.

4.1.5.45 TYSetString()

```
TY_CAPI TYSetString (
            TY_DEV_HANDLE hDevice,
            TY_COMPONENT_ID componentID,
            TY_FEATURE_ID featureID,
            const char * buffer )
```

Set value of string feature.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
in	*buffer*	String buffer.

Return values

TY_STATUS_OK	Succeed.

Return values

TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRING.
TY_STATUS_NULL_POINTER	buffer is NULL.
TY_STATUS_OUT_OF_RANGE	Input string is too long.
TY_STATUS_BUSY	Device is capturing, the feature is locked.

4.1.5.46 TYSetStruct()

```
TY_CAPI TYSetStruct (
             TY_DEV_HANDLE hDevice,
             TY_COMPONENT_ID componentID,
             TY_FEATURE_ID featureID,
             void * pStruct,
             uint32_t structSize )
```

Set value of struct.

Parameters

in	*hDevice*	Device handle.
in	*componentID*	Component ID.
in	*featureID*	Feature ID.
in	*pStruct*	Pointer of struct.
in	*structSize*	Size of struct.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	Invalid component ID.
TY_STATUS_INVALID_FEATURE	Invalid feature ID.
TY_STATUS_NOT_PERMITTED	The feature is not writable.
TY_STATUS_WRONG_TYPE	The feature's type is not TY_FEATURE_STRUCT.
TY_STATUS_NULL_POINTER	pStruct is NULL.
TY_STATUS_WRONG_SIZE	structSize incorrect.
TY_STATUS_BUSY	Device is capturing, the feature is locked.

4.1.5.47 TYStartCapture()

```
TY_CAPI TYStartCapture (
            TY_DEV_HANDLE hDevice )
```

Start capture.

Parameters

in	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_INVALID_COMPONENT	No components enabled.
TY_STATUS_BUSY	Device has been started.
TY_STATUS_DEVICE_ERROR	Start capture failed.

4.1.5.48 TYStopCapture()

```
TY_CAPI TYStopCapture (
            TY_DEV_HANDLE hDevice )
```

Stop capture.

Parameters

in	*hDevice*	Device handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_INVALID_HANDLE	Invalid device handle.
TY_STATUS_IDLE	Device is not capturing.
TY_STATUS_DEVICE_ERROR	Stop capture failed.

4.1.5.49 TYUpdateDeviceList()

```
TY_CAPI TYUpdateDeviceList (
            TY_INTERFACE_HANDLE ifaceHandle )
```

Update current connected devices.

Parameters

in	*ifaceHandle*	Interface handle.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.
TY_STATUS_INVALID_INTERFACE	Invalid interface handle.

4.1.5.50 TYUpdateInterfaceList()

```
TY_CAPI TYUpdateInterfaceList ( )
```

Update current interfaces.

Return values

TY_STATUS_OK	Succeed.
TY_STATUS_NOT_INITED	TYInitLib not called.

Index