emergent
vision technologies

Camera Programmer's Manual

v1.0.3

July 11, 2019

Emergent Vision Technologies | 3135-580 Nicola Ave | Port Coquitlam, BC | V3B 0P2 | Canada | www.emergentvisiontec.com

Table of Contents

Contact

Emergent Vision Technologies Canada (Headquarters)

3135-580 Nicola Ave
Port Coquitlam, BC
V3B 0P2
Canada
info@emergentvisiontec.com
www.emergentvisiontec.com

Technical Support

info@emergentvisiontec.com

Legal

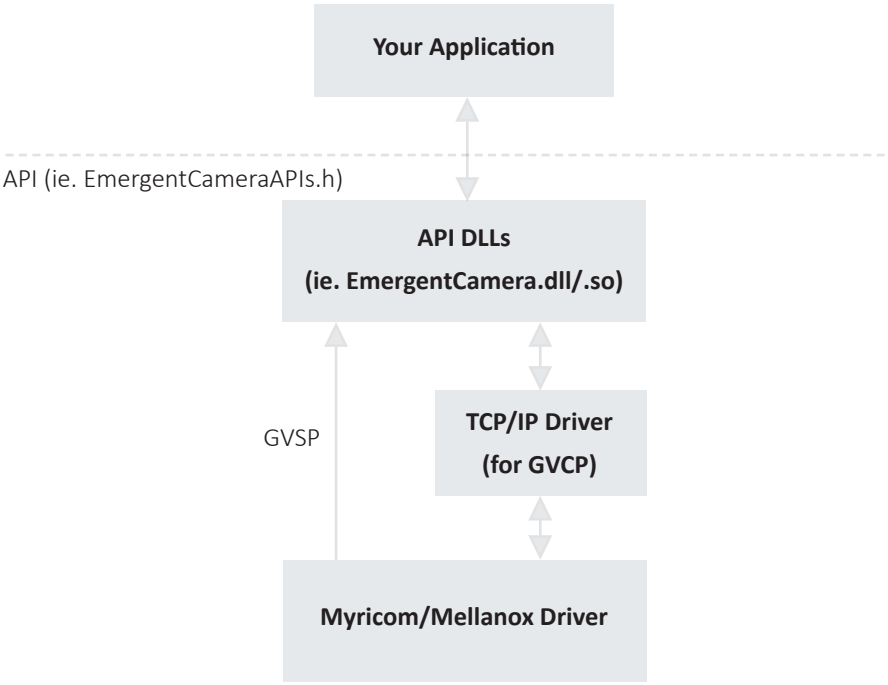
Overview

This document is the programmer's reference manual for Emergent Vision Technologies cameras.

The Emergent Vision Technologies API interface supports all cameras from Emergent.

The API driver interface is a user DLL/SO which communicates with the network drivers for control (GVCP) and receives image data (GVSP) directly from the Myricom or Mellanox drivers.

On Windows the API interface is through .dll and on Linux through .so.



Using the Driver

Platform

The Emergent Vision Technologies driver is supported on the following platforms:

- Windows 7/8/10 (64bit) for Myricom cards and Windows 10 for Mellanox cards.
- Linux (64bit) in distributions from Ubuntu, RedHat, CentOS.

Our driver supports the Myricom 10G PCIe single and dual port NICs with custom Myricom NIC firmware and driver for optimal CPU utilization and latency.

For 25GigE, we support Mellanox NICs using VMA for Linux and Rivermax for Windows for optimal CPU utilization and latency.

Programming Languages

Main Header File:

EmergentCameraAPIs.h

See: <installation directory>\eSDK\include

Main Library File(Windows):

EmergentCamera.lib

See: <installation directory>\eSDK\lib

Main DLL/SO File:

EmergentCamera.dll/.so

See: <installation directory>\eSDK\bin

All files are included in the SDK and the functional examples in the following section illustrate their inclusion and usage.

Using the API

Example Code

C++ example code is included in the …\EVT\eSDK\examples directory.
Examples are provided to demonstrate and exercise all functions in this document and all camera functionality and are thus an excellent resource to software developers.

All example projects can be opened, compiled and run using the Visual Studio Express 2017 software or in Linux using provided makefiles and g++.

The following examples are provided in order of importance:

EVT_BenchmarkHS-
This example is the primary example for starting your application development and for benchmarking your camera at maximum resolution and frame rate.
EVT_ImageFormatControl – illustrates pixel format use and file saving.
EVT_GPIO – illustrates various triggering methods.
EVT_Mcast/EVT_Mcast_Master – illustrates the Multicast functionality.
EVT_Mcast/EVT_Mcast_Slave – illustrates the Multicast functionality.
EVT_PTP – illustrates the PTP functionality.
EVT_DeviceInformation- exercises the Device Information camera features.
EVT_AnalogControl- exercises the Analog Control camera features.
EVT_AcquisitionControl – exercises the Acquisition Control camera features.
EVT_TransportLayerControl – exercises the Transport Layer camera features.
EVT_BenchmarkHS_Sync – illustrates synchronization using Myricom sync NIC.
EVT_BenchmarkHS_Dual – illustrates 2 camera single threaded functionality
EVT_BenchmarkHS_MultiThread – illustrates 2 camera multi-threaded functionality
EVT_BenchmarkHSAVI – illustrates the AVI saving API.

List Available Cameras

Function EVT_ListDevices will enumerate all Emergent Vision Technologies cameras connected to the system.

Opening a Camera

A camera must be opened to control and capture images. Function EVT_CameraOpen is used to open the camera.
The camera must be closed with EVT_CameraClose as the application is finished.

Camera Attributes

Attributes are used to control and monitor various aspects of the driver and camera.
See EVT Camera Attributes Manual for the complete description of camera attributes.

Attribute Type	Get	Set	Range
Enumeration	EVT_CameraGetEnumParam	EVT_CameraSetEnumParam	EVT_ CameraGetEnumParamRange
UInt32	EVT_CameraGetUInt32Param	EVT_CameraSetUInt32Param	EVT_CameraGetUInt32ParamMin EVT_CameraGetUInt32ParamMax EVT_CameraGetUInt32ParamInc
Int32	EVT_CameraGetInt32Param	EVT_CameraSetInt32Param	EVT_CameraGetInt32ParamMin EVT_CameraGetInt32ParamMax EVT_CameraGetInt32ParamMin
String	EVT_CameraGetStringParam	EVT_CameraSetStringParam	N/A
Boolean	EVT_CameraGetBoolParam	EVT_CameraSetBoolParam	N/A
Command	N/A	EVT_CameraExecuteCommand	N/A

Table 1: Functions for Reading and Writing Attributes

The EVT API currently defines the following attribute types:

Enumeration	A set of values. Values are represented as strings.
Uint32	32-bit unsigned value.
Boolean	A simple Boolean value (true, false)
Int32	32-bit signed value
String	A string (null terminated, char[]).
Command	Valueless; a function executes when the attribute is written.

Ie. to change the exposure time, set attribute “Exposure” (see Attributes manual):
EVT_CameraSetUInt32Param (Camera, “Exposure”, 1000); // 1000 µs

Frame Queuing

Frames are class objects containing image data and related info. See CEmergentFrame in emergentframe.h. Users are responsible for managing frame queues.

To create a frame, simply declare as in the examples:

```
CEmergentFrame   evtFrame;
        Or
CEmergentFrame   evtFrames[10];  //Declares an array of frames.
```

To queue a frame, do as follows:

```
CEmergentFrame evtFrame;
EVT_AllocateFrameBuffer(&camera,&evtFrame,EVT_FRAME_BUFFER_ZERO_
COPY);
EVT_CameraQueueFrame(&camera, &evtFrame); //Queue the frame.
(Where camera is of type CEmergentCamera opened with EVT_CameraOpen)
```

To receive a queued-up frame into the application, do as follows:

```
EVT_CameraGetFrame(&camera, &evtFrame, 100);//Timeout in milliseconds
        Or
EVT_CameraGetFrame(&camera, &evtFrame, EVT_INFINITE); //Timeout infinite
```

To release memory for a frame, do as follows (for pre-allocated frame):

```
EVT_ReleaseFrameBuffer(&camera, &evtFrame);
```

For high-speed image capture, it is suggested that multiple frames are queued up at any given time to deal with fluctuations in system calls (particularly on Windows).

A maximum of 200 frame buffers can be queued up at any given time per NIC port or until host runs out of memory.

Image buffers are filled in the order they are queued. Re-queue new frames as the old frames complete.

This and many other elements of programming are illustrated in the provided examples with EVT_BenchmarkHS being the primary example for an efficient grab loop.

Error Codes

Most Emergent API functions return an EVT_ERROR type error code.
Errors are listed with each function in the Function Reference section of this document.
Following are descriptions of error codes that might be returned:

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS	Function successful
EVT_ERROR_SRCH	Parameter not found
EVT_ERROR_INTR	Signal was received
EVT_ERROR_IO	File IO error
EVT_ERROR_AGAIN	Timeout expired
EVT_ERROR_NOMEM	Out of memory
EVT_ERROR_INVAL	Invalid argument
EVT_ERROR_NOT_SUPPORTED	Buffer type is not supported
EVT_ERROR_DEVICE_CONNECTED_ALRD	Device already opened
EVT_ERROR_DEVICE_NOT_CONNECTED	Device not opened
EVT_ERROR_DEVICE_LOST_CONNECTION	Device lost connection.
EVT_ERROR_GENICAM_ERROR	Generic GenICam error
EVT_ERROR_GENICAM_NOT_MATCH	Parameter type does not match.
EVT_ERROR_GENICAM_OUT_OF_RANGE	Parameter value out of range
EVT_ERROR_SOCK	Socket operation failed
EVT_ERROR_GVCP_ACK	GVCP ACK error
EVT_ERROR_GVSP_DATA_CORRUPT	Frame data corrupt
EVT_ERROR_NIC_LIB_INIT	Failed to initialize Myricom API
EVT_ERROR_OS_OBTAIN_ADAPTER	Failed to get host adapter information
EVT_ERROR_SDK	Unexpected SDK error
EVT_GENERAL_ERROR	General error

Structures

CEmergentFrame

```
enum PIXEL_FORMAT pixel_type; //See emergentframebase.h for types
unsigned int size_x; //All from GVSP header
unsigned int size_y;
unsigned int offset_x;
unsigned int offset_y;
unsigned int padding_x;
unsigned int padding_y;
unsigned int trailer_size_y;
unsigned short frame_id; //aka block_id from GVSP header
unsigned char* imagePtr;  //The image data pointer
int convertColor;//See emergentframebase.h for types
int convertBitDepth;  //See emergentframebase.h for types
unsigned long long nsecs; //Timestamp when using Myri sync nic and IRIGB
unsigned long long timestamp; //Timestamp for GevTimestamp values.
```

GigEVisionDeviceInfo //All from GigEVision specification bootstrap registers

```
unsigned short specVersionMajor;
unsigned short specVersionMinor;
unsigned int deviceMode;
char macAddress[18];
unsigned int ipConfigOptions;
unsigned int ipConfigCurrent;
char currentIp[16];
char currentSubnetMask[16];
char defaultGateway[16];
char manufacturerName[32];
char modelName[32];
char deviceVersion[32];
char manufacturerSpecifiedInfo[48];
char serialNumber[16];
char userDefinedName[16];
char hostInterfaceIp[16];
```

```
CEmergentAVIFile
    char fileName[256],
    int codec, //FourCC codec code.
    double fps, //Frames per second.
    int width,    //Width and height (resolution) of AVI file.
    int height,
    bool isColor //TRUE for color, FALSE for mono sensors.
```

Function Reference

EVT_CameraOpen

Open the EVT Camera

Prototype

```
EVT_ERROR EVT_CameraOpen
   (
   CEmergentCamera* camera,
   GigEVisionDeviceInfo* deviceInfo,
   const char* xmlFileName = NULL
   );
```

Parameters

camera:	Camera handle
deviceInfo:	Handle to a GigEVisionDeviceInfo structure
xmlFileName:	Local camera XML file. Omit or NULL to use in-camera XML file.

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_CONNECTED_ALRD
- EVT_ERROR_INVAL
- EVT_ERROR_NOMEM
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_GENERAL_ERROR

Notes

Call EVT_CameraClose() to close camera when done.

See eSDK examples for detailed functional sample usage.

EVT_CameraClose

Closes the EVT Camera

Prototype

```
EVT_ERROR EVT_CameraClose
(
CEmergentCamera* camera,
);
```

Parameters

camera: Camera handle

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_GENERAL_ERROR

Notes

Call EVT_CameraOpen() before closing camera.
See eSDK examples for detailed functional sample usage.

EVT_CameraGetParamAttr

Get the information such as data type for a particular attribute.

Prototype

```
EVT_ERROR EVT_CameraGetParamAttr
(
CEmergentCamera* camera,
const char* name,
struct EvtParamAttribute* attr
);
```

Parameters

camera:	Camera handle
name:	Attribute name
attr:	Attribute information returned in this structure

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetUInt32Param

Get the value of a UInt32 attribute.

Prototype

```
EVT_ERROR EVT_CameraGetUInt32Param
(
CEmergentCamera* camera,
const char* name,
unsigned int* val
);
```

Parameters

camera:	Camera handle
name:	Attribute name
val:	Attribute value returned

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraSetUInt32Param

Set the value of a UInt32 attribute.

Prototype

```
EVT_ERROR EVT_CameraSetUInt32Param
(
CEmergentCamera* camera,
const char* name,
unsigned int val
);
```

Parameters

camera:	Camera handle
name:	Attribute name
val:	Attribute value

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetUInt32ParamMax

Get the maximum value of UInt32attribute.

Prototype

```
EVT_ERROR EVT_CameraGetUInt32ParamMax
(
CEmergentCamera* camera,
const char* name,
unsigned int* max
);
```

Parameters

camera:	Camera handle
name:	Attribute name
max:	Attribute maximum value returned

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetUInt32ParamMin

Get the minimum value of a UInt32 attribute.

Prototype

```
EVT_ERROR EVT_CameraGetUInt32ParamMin
(
CEmergentCamera* camera,
const char* name,
unsigned int* min
);
```

Parameters

camera:	Camera handle
name:	Attribute name
min:	Attribute minimum value returned

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetUInt32ParamInc

Get the increment step of a UInt32 attribute.

Prototype

```
EVT_ERROR EVT_CameraGetUInt32ParamInc
(
CEmergentCamera* camera,
const char* name,
unsigned int* inc
);
```

Parameters

camera:	Camera handle
name:	Attribute name
inc:	Attribute increment step returned

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetInt32Param

Get the value of an Int32 attribute.

Prototype

```
EVT_ERROR EVT_CameraGetInt32Param
(
CEmergentCamera* camera,
const char* name,
int* val
);
```

Parameters

camera:	Camera handle
name:	Attribute name
val:	Attribute value returned

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraSetInt32Param

Set the value of an Int32attribute.

Prototype

```
EVT_ERROR EVT_CameraSetInt32Param
(
CEmergentCamera* camera,
const char* name,
int val
);
```

Parameters

camera:	Camera handle
name:	Attribute name
val:	Attribute value

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetInt32ParamMax

Get the maximum value of an Int32 attribute.

Prototype

```
EVT_ERROR EVT_CameraGetInt32ParamMax
(
CEmergentCamera* camera,
const char* name,
int* max
);
```

Parameters

camera:	Camera handle
name:	Attribute name
max:	Attribute maximum value returned

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetInt32ParamMin

Get the minimum value of an Int32 attribute.

Prototype

```
EVT_ERROR EVT_CameraGetInt32ParamMin
(
CEmergentCamera* camera,
const char* name,
int* min
);
```

Parameters

camera:	Camera handle
name:	Attribute name
min:	Attribute minimum value returned

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetInt32ParamInc

Get the increment step of an Int32 attribute.

Prototype

```
EVT_ERROR EVT_CameraGetInt32ParamInc
(
CEmergentCamera* camera,
const char* name,
unsigned int* inc
);
```

Parameters

camera:	Camera handle
name:	Attribute name
inc:	Attribute increment step returned

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetBoolParam

Get the value of a Boolean attribute.

Prototype

```
EVT_ERROR EVT_CameraGetBoolParam
(
CEmergentCamera* camera,
const char* name,
bool* val
);
```

Parameters

camera:	Camera handle
name:	Attribute name
val:	Attribute value returned

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraSetBoolParam

Set the value of a Boolean attribute.

Prototype

```
EVT_ERROR EVT_CameraSetBoolParam
(
CEmergentCamera* camera,
const char* name,
bool val
);
```

Parameters

camera:	Camera handle
name:	Attribute name
val:	Attribute value

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraExecuteCommand

Execute a command attribute.

Prototype

```
EVT_ERROR EVT_CameraExecuteCommand
(
CEmergentCamera* camera,
const char* name
);
```

Parameters

camera:	Camera handle
name:	Command attribute to be executed

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetStringParam

Get the value of a string attribute.

Prototype

```
EVT_ERROR EVT_CameraGetStringParam
(
CEmergentCamera* camera,
const char* name,
char* buffer,
unsigned long bufferSize,
unsigned long* valueSize
);
```

Parameters

camera:	Camera handle
name:	Attribute name
buffer:	Buffer for returned string attribute
bufferSize:	Size of buffer provided
valueSize:	Actual size of returned string attribute

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraSetStringParam

Set the value of a string attribute.

Prototype

```
EVT_ERROR EVT_CameraSetStringParam
(
CEmergentCamera* camera,
const char* name,
const char* buffer
);
```

Parameters

camera:	Camera handle
name:	Attribute name
buffer:	Buffer containing the string to write.

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_SRCH
- EVT_ERROR_GENICAM_ERROR
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_ERROR_GENICAM_NOT_MATCH
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetStringParamMaxLength

Set the value of a string attribute.

Prototype

```
EVT_ERROR EVT_CameraSetStringParam
(
CEmergentCamera* camera,
const char* name,
int* max
);
```

Parameters

camera:	Camera handle
name:	Attribute name
max:	Maximum size in bytes allowed for EVT_CameraSetStringParam

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetEnumParam

Get the value of an Enumeration attribute.

Prototype

```
EVT_ERROR EVT_CameraGetEnumParam
(
CEmergentCamera* camera,
const char* name,
char* buffer,
unsigned long bufferSize,
unsigned long* valueSize
);
```

Parameters

camera:	Camera handle
name:	Attribute name
buffer:	Buffer for returned enum attribute
bufferSize:	Size of buffer provided
valueSize:	Actual size of returned string attribute

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraSetEnumParam

Set the value of an Enumeration attribute.

Prototype

```
EVT_ERROR EVT_CameraSetEnumParam
(
CEmergentCamera* camera,
const char* name,
const char* buffer
);
```

Parameters

camera:	Camera handle
name:	Attribute name
buffer:	Enum attribute value

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetEnumParamRange

Get the range of an Enumeration attribute.

Prototype

```
EVT_ERROR EVT_CameraGetEnumParamRange
(
CEmergentCamera* camera,
const char* name,
char* buffer,
unsigned long bufferSize,
unsigned long* valueSize
);
```

Parameters

camera:	Camera handle
name:	Attribute name
buffer:	Buffer for returned enum attributes. Comma separated list.
bufferSize:	Size of buffer provided
valueSize:	Actual size of returned enum attribute comma separated list.

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SRCH
EVT_ERROR_GENICAM_ERROR
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_GENICAM_NOT_MATCH
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraOpenStream

Open stream channel on host side.

Prototype

```
EVT_ERROR EVT_CameraOpenStream
(
CEmergentCamera* camera
);
```

Parameters

camera: Camera handle

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_DEVICE_NOT_CONNECTED
EVT_ERROR_SOCK
EVT_ERROR_GVCP_ACK
EVT_ERROR_MYRICOM_INIT
EVT_ERROR_INVAL
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraCloseStream

Closes stream channel on host side.

Prototype

```
EVT_ERROR EVT_CameraCloseStream
(
CEmergentCamera* camera
);
```

Parameters

camera: Camera handle

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS

EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_AllocateFrameBuffer

Allocate memory for the image.

Prototype

```
EVT_ERROR EVT_AllocateFrameBuffer
(
CEmergentCamera* camera,
CEmergentFrame* frame,
int buffer_type
);
```

Parameters

camera:	Camera handle
frame:	Image frame object
buffer_type:	EVT_FRAME_BUFFER_DEFAULT
	EVT_FRAME_BUFFER_ZERO_COPY

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_NOT_SUPPORTED
- EVT_ERROR_INVAL
- EVT_ERROR_NOMEM
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

Use EVT_FRAME_BUFFER_ZERO_COPY for frame for EVT_CameraGetFrame and EVT_CameraQueueFrame as this buffer type is needed for transfer of image data from the NIC with highest efficiency. For all other frames such as the destination frame of EVT_FrameConvert when converting received frames use EVT_FRAME_BUFFER_DEFAULT as this performs standard memory allocation.

EVT_ReleaseFrameBuffer

Release memory previously allocated with EVT_AllocateFrameBuffer()

Prototype

```
EVT_ERROR EVT_ReleaseFrameBuffer
(
CEmergentCamera* camera,
CEmergentFrame* frame
);
```

Parameters

camera:	Camera handle
frame:	Image frame from camera.

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS

EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraQueueFrame

Place an image buffer into the frame queue.

Prototype

```
EVT_ERROR EVT_CameraQueueFrame
(
CEmergentCamera* camera,
CEmergentFrame* frame
);
```

Parameters

camera:	Camera handle
frame:	Camera image frame to queue.

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_INVAL
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_CameraGetFrame

Get frame from the camera. Block calling thread until frame is received.

Prototype

```
EVT_ERROR EVT_CameraGetFrame
(
CEmergentCamera* camera,
CEmergentFrame* frame,
int milliseconds
);
```

Parameters

camera:	Camera handle
frame:	Image frame from camera.
milliseconds:	Time to wait in milliseconds. Use EVT_INFINITE to block indefinitely.

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_INVAL
- EVT_ERROR_AGAIN
- EVT_ERROR_INTR
- EVT_ERROR_NOMEM
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_FrameConvert

Convert a frame using the provided parameters.

Prototype

```
EVT_ERROR EVT_FrameConvert
(
CEmergentFrame* frameSrc,
CEmergentFrame* frameDst,
int convertBitDepth,
int convertColor
);
```

Parameters

frameSrc:	Image frame to convert
frameDst:	Image frame result.
convertBitDepth:	Parameter to define conversion of number of bits per pixel.
convertColor:	Parameter to define color conversion.

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS

EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

convertBitDepth can be one of the following:

EVT_CONVERT_NONE, EVT_CONVERT_8BIT

convertColor can be one of the following:

EVT_COLOR_CONVERT_NONE

EVT_COLOR_CONVERT_NEARESTNEIGHBOR_RGB

EVT_COLOR_CONVERT_NEARESTNEIGHBOR_BGR

EVT_COLOR_CONVERT_BILINEAR_RGB

EVT_COLOR_CONVERT_BILINEAR_BGR

EVT_COLOR_CONVERT_TO_RGB

EVT_COLOR_CONVERT_TO_BGR

EVT_FrameConvert can perform three tasks and all are wrapped up into a single API function call for efficiency purposes. Those three tasks are:

1. Unpacking raw packed frame data from the camera into PC usable format such as taking a 10 bit packed format (which has two 10 bit pixels over 3 bytes) and unpacking this into 16 bits per pixel. This operation is always performed for the applicable formats. The mono8 format ie. is one which no unpacking operation is performed.
2. Bit converting the data such as converting 10 bits per pixel to 8 bits per pixel. Specify the operation through the “convertBitDepth” parameter.
3. Color converting the data. For YUV, BGR, and RGB formats, convert to RGB/BGR options are available. For raw bayer formats such as BayerGB8 and BayerGB10, bayer interpolation options are available. Specify the operation through the “convertColor” parameter.

Omitting the smaller details for clarity (see examples for complete usage). Here are some usage examples.

```
//Using the frame data in src provides the raw data from the camera in whichever
//format is selected. For mono8, mono10, RGB8Packed this is all most applications need.
EVT_CameraGetFrame(camera, src, INFINITE);

//This provides the unpacked data
//ie. 10 bit packed format mono10Packed will be converted to 16 bits per pixel.
EVT_FrameConvert(src, dst, EVT_CONVERT_NONE, EVT_COLOR_CONVERT_NONE);
//This does the same but downgrades 10 bits to 8 bits per pixel after unpacking.
EVT_FrameConvert(src, dst, EVT_CONVERT_8BIT, EVT_COLOR_CONVERT_NONE);

//This unpacks a YUV camera format image and converts to RGB
EVT_FrameConvert(src, dst, EVT_CONVERT_NONE, EVT_COLOR_CONVERT_TO_RGB);

//This takes raw bayer from the camera (ie. Format BayerGB8 or BayerGB10)
//and bayer interpolates the frame with bi-linear method to RGB format.
EVT_FrameConvert(src,dst,EVT_CONVERT_NONE, EVT_COLOR_CONVERT_BILINEAR_RGB);
//Same but downgrades 10bpp to 8bpp (applicable to BayerGB10)
EVT_FrameConvert(src, dst ,EVT_CONVERT_8BIT, EVT_COLOR_CONVERT_BILINEAR_RGB);
```

EVT_ListDevices

Lists all devices connected to Ethernet ports and fills in deviceInfo for each.

Prototype

```
EVT_ERROR EVT_ListDevices
(
GigEVisionDeviceInfo* deviceInfo,
unsigned int* bufSize,
unsigned int* actualNum,
const struct ListDevicesSettings* settings
);
```

Parameters

deviceInfo:	Pointer to array of GigEVisionDeviceInfo objects (one for as many devices in system or buflen devices)
bufSize:	Number of devices in system. Match with number of GigEVisionDeviceInfo objects for first parameter.
actualNum:	How many devices were actually found.
ListDevicesSettings:	Includes “timeout” struct variable for discovery timeout.

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_OS_OBTAIN_ADAPTER
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_FrameSave

Save image to file in selected format.

Prototype

```
EVT_ERROR EVT_FrameSave
(
CEmergentFrame* frame,
char* filename,
int filetype,
int align
);
```

Parameters

frame: Image frame to save to file

filename: File name to use for saved file – a char array.

filetype: Format to save image in.

File type format can be one of:

EVT_FILETYPE_RAW
EVT_FILETYPE_BMP
EVT_FILETYPE_TIF

align: For bit depths less that are not 8 or 16 bits, this parameter determines whether to left align the pixel data before shifting.

EVT_ALIGN_LEFT
EVT_ALIGN_NONE

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS
EVT_ERROR_INVAL
EVT_ERROR_IO
EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_AVIOpen

Open new AVI file for which to append images.

Prototype

```
EVT_ERROR EVT_AVIOpen
(
CEmergentAVIFile* aviFile
);
```

Parameters

aviFile: CEmergentAVIFile to open

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS

EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage but to note that prior to calling the open function that certain parameters must be setup as follows:

CEmergentAVIFile aviFile;
aviFile.isColor = TRUE; //or FALSE if mono.
aviFile.fps = 200; //The frame rate for the AVI.
aviFile.codec = EVT_CODEC_NONE; //The fourcc codec code. EVT_CODEC_NONE = 0.
aviFile.width = 2048; //The width and height(ie. resolution)
aviFile.height = 2048;
strcpy_s(aviFile.fileName, “C:\\myavifile.avi”); //Set file name string. 256 char max.
EVT_AVIOpen(&aviFile); //Now can open the file.

EVT_AVIClose

Close and complete AVI file.

Prototype

```
EVT_ERROR EVT_AVIClose
(
CEmergentAVIFile* aviFile
);
```

Parameters

aviFile: CEmergentAVIFile to close

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS

EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_AVIAppend

Append image to AVI file opened with EVT_AVIAppend.

Prototype

```
EVT_ERROR EVT_AVIAppend
(
CEmergentAVIFile* aviFile,
CEmergentFrame* frame
);
```

Parameters

aviFile:	CEmergentAVIFile for which to append frame.
frame:	Image frame to append to AVI file

Return Value

EVT_ERROR. Return codes for this function are:

EVT_SUCCESS

EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage but to note that the frames that will be appended must match the parameters setup for EVT_AVIOpen.

EVT_ForceIP

Set the IP address on a camera which is already open.

Prototype

```
EVT_ERROR EVT_ForceIP
(
CEmergentCamera* camera,
const char* ipAddress,
const char* subnetMask,
const char* defaultGateway
);
```

Parameters

camera:	Camera handle
ipAddress:	IP address to set. Ie. 192.168.1.70
subnetMask:	Subnet mask to set. Ie. 255.255.255.0
defaultGateway:	Default gateway to set.

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_INVAL
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

EVT_ForceIPEx

Set the IP address on a camera which is not open.

Prototype

```
EVT_ERROR EVT_ForceIPEx
(
const char* macAddress,
const char* ipAddress,
const char* subnetMask,
const char* defaultGateway
);
```

Parameters

macAddress:	MAC address of camera to set IP for. Ie. E0:55:97:00:12:34
ipAddress:	IP address to set. Ie. 192.168.1.70
subnetMask:	Subnet mask to set. Ie. 255.255.255.0
defaultGateway:	Default gateway to set.

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_INVAL
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

Worth noting is that the Emergent MAC is printed on the camera label and is also defined by the camera serial number as follows.
Emergent OUI: E0:55:97 (always for Emergent cameras)
Serial number example: 4660 which converts to 00:12:34 hexi-decimal.
So, we simply append these to get the MAC address:
E0:55:97 + 00:12:34 = E0:55:97:00:12:34

EVT_IPConfig

Set the IP address on a camera which is already open and have this persistent through boot.

Prototype

```
EVT_ERROR EVT_IPConfig
(
CEmergentCamera* camera,
bool usePersistentIP,
const char* ipAddress,
const char* subnetMask,
const char* defaultGateway
);
```

Parameters

camera:	Camera handle
usePersistentIP:	True to use persistent IP or False for DHCP.
ipAddress:	Persistent IP address to set. Ie. 192.168.1.70
subnetMask:	Persistent Subnet mask to set. Ie. 255.255.255.0
defaultGateway:	Persistent Default gateway to set.

Return Value

EVT_ERROR. Return codes for this function are:

- EVT_SUCCESS
- EVT_ERROR_DEVICE_NOT_CONNECTED
- EVT_ERROR_INVAL
- EVT_ERROR_SOCK
- EVT_ERROR_GVCP_ACK
- EVT_GENERAL_ERROR

Notes

See eSDK examples for detailed functional sample usage.

Document History

Version	Date	Description
1.01	29 May 2012	Initial Version
1.02	3 March 2013	Adding Linux Details
1.03	11 July 2019	Re-Write