# SphinxLib

---

## *Library functions*

**Version 2.7.0 (December 2021)**

## Table of Content

# 1. Introduction

Sphinx for GigE Vision is an image acquisition library, which provides access to all GigE Vision compliant devices.

Sensor to Image provides a sample implementation in c source code and also a viewer application (SphinxGEVViewer), which could be used as reference for customized applications.

All GigE Vision devices have to provide an xml-file, describing the system features. So the recommended way to write an application for GigE Vision devices is to download and evaluate this xml-file and then activate features, which are supported by the system.

The following document describes the library functions, which are available for Windows or Linux based systems.

You'll find the SphinxLib.dll/sphinxlib.a in the software directory of the provided archive.

# 2. SphinxLib functions for Windows and Linux

This chapter describes the single functions of the C/C++ Library for Windows/Linux. The following example describes and explains the function:

## Sample function

| | |
|---|---|
| Function: | A short description of the function. |
| Syntax: | The function’s syntax description in C/C++. |
| Description: | This section contains the purpose and the usage of the function. Also the parameters are explained. |
| Return: | Here you find the type and range of the return values.<br>Many functions return predefined status codes (GEV_STATUS_...).<br>Please check SphinxLib.h for reference. |
| Reference: | List of other routines, which have a relationship to the current function.<br>In addition refer to the GigE Vision specification (GEVS).<br>For detailed error codes of GEV_STATUS_LOCAL_PROBLEM, please use function get_local error. |

In the directory `Windows/sphinxlib` you will find the SphinxLib library SphinxLib.dll, sphinxlib.lib and the header-file SphinxLib.h
In the directory `Linux/sphinxlib/` you will find the GigE Vision Device library sphinxlib.a and the header-file SphinxLib.h

The latest version can be obtained from Sensor to Image.

The functions marked as intern, are usually not used in application development. But they are used by other functions and special custom commands.

Following data types are equivalent:

| | | | |
|---|---|---|---|
| Windows: | BYTE | WORD | DWORD |
| Linux: | unsigned char | unsigned short | unsigned long |

Principle of work:

Almost every function corresponds with the feature, referenced by the xml-file. So if a function is used, it uses the registers mentioned in the xml-file.

**Important:**

The files in the directory Windows\SphinxLib\Extras must be in the same directory as the SphinxLib.dll. These files are used by the SphinxLib.dll to parse the xml file of the device.

DLL Versions: MathParser-1.1.2, libxml2-2.9.7, libxslt-1.1.32

libxml2 and libxslt -> http://www.xmlsoft.org , MIT Licence

MathParser -> for calculation of mathematical formulas, http://kirya.narod.ru/mathparser.html , LGPL license

The files in the directory Linux/sphinxlib/extras must be in the same directory as the application. These files are used by the sphinxlib.a library to parse the xml file of the device.

The *.xsd (shema files) and *.xsl (stylesheet files) files are necessary to parse the xml file of the device.

Network byte order for all functions who use ip addresses (GEVInit, GEVDiscovery, GEVDiscoveryAdapter, GEVForceIP, GEVSetNetConfig, GevGetNetConfig).

## 2.1. General functions

### GEVCheckDeviceStatus

| | |
|---|---|
| Function: | Checks the status of a GigE Vision device |
| Syntax: | `WORD GEVCheckDeviceStatus(DWORD ip_adapter, DWORD mask_adapter, DWORD ip_device, BYTE *device_status, DWORD ack_timeout, WORD port);` |
| Description: | This function checks the status of a GigE Vision device.<br>The *ip_adapter* parameter specifies the IP address of the network adapter where the device is connected.<br>The *mask_adapter* parameter specifies the net mask of the network adapter.<br>Parameter *ip_device* specifies the ip address of the GigE Vision device to check.<br>Parameter *status* returns the status of the device. See status entry of DEVICE_PARAM struct.<br>Parameter *ack_timeout* specifies the acknowledge timeout of the status register read.<br>Parameter *port* specifies the destination port address of the control channel. Value of 0 set the port automatically. |
| Return: | see Error Codes |
| Reference: | GEVDiscovery, GEVDiscoveryAdapter |

### GEVClose

| | |
|---|---|
| Function: | Close GigE Vision device |
| Syntax: | `WORD GEVClose(BYTE cam_nr);` |
| Description: | This function closes the communication session (GEV Control Channel) to a GigE Vision device.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50]. |
| Return: | see Error Codes |
| Reference: | GEVInit |

### GEVCloseFilterDriver

| | |
|---|---|
| Function: | Close GigE Vision device filter driver |
| Syntax: | `WORD GEVCloseFilterDriver(BYTE cam_nr);` |
| Description: | This function closes the S2I filter driver for the stream channel of the specified device<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50]. |
| Return: | see Error Codes |
| Reference: | GEVInitFilterDriver, GEVGetFilterDriverVersion |

### GEVCloseStreamChannel

| | |
|---|---|
| Function: | Close GigE Vision device stream channel |
| Syntax: | `WORD GEVCloseStreamChannel(BYTE cam_nr);` |
| Description: | This function closes the stream channel of the specified device.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50]. |

Return: see Error Codes

Reference: GEVOpenStreamChannel

## GEVDiscovery

Function: Find all GigE Vision devices of all network interface adapters

Syntax:

```
WORD GEVDiscovery(DISCOVERY *dis,
DISCOVERY_CALLBACK_FUNC c_func, DWORD d_timeout, BOOL
ignore_subnet, WORD port);
```

Description: This function can discover up to 50 GigE Vision compliant devices in the network.
Paramter *dis* returns number and parameters of the found devices (see 2.6 Structures and SphinxLib.h for the definition of struct DISCOVERY and DEVICE_PARAM).
Parameter c_*func* specifies the discovery callback function.
Parameter *d_timeout* specifies a discovery timeout in ms.
Parameter *ignore_subnet* specifies if subnet ignore flag of discovery message should be set.
Parameter *port* specifies the destination port address of the control channel. Value of 0 set the port automatically.
Network byte order for ip addresses.

There are two methods of obtaining GigE Vision devices.

1) GEVDiscovery(&dis,NULL,200,0);

This call returns the founded devices in the dis parameter. (maximum 255 see DISCOVERY struct see 2.6 Structures)

2) GEVDiscovery(NULL, discovery_callback_func, 200, 0);

This call returns the founded devices in the callback function

**Important:**
Devices returned from the callback did not have their status checked.
This is can you do with the GEVCheckDeviceStatus function.

```
BYTE   WINAPI   discovery_callback_func(int   s_cnt,   DEVICE_PARAM
*dparam)

{

  if(dparam)

  {

   // found device
    if(dparam->status == DISCOVERY_STATUS_NOT_CHECKED)
       GEVCheckDeviceStatus(dparam->AdapterIP,
       dparam->AdapterMask, dparam->IP, &dparam ->status);

    ...

  }

}
```

More you can see in the ConsoleDemo.cpp file.

Return: see Error Codes

Reference: GEVInit, GEVCheckDeviceStatus

## GEVDiscoveryAdapter

Function: Find all GigE Vision devices of one network interface adapters

Syntax:

```
WORD GEVDiscoveryAdapter(DISCOVERY* dis,
ADAPTER_PARAM* adapter, DISCOVERY_CALLBACK_FUNC
c_func, DWORD d_timeout, BOOL ignore_subnet, WORD
port);
```

Description: This function can discover up to 50 GigE Vision compliant devices in the network.
Paramter *dis* returns number and parameters of the found devices (see 2.6 Structures and SphinxLib.h for the definition of struct DISCOVERY and DEVICE_PARAM).
Paramter *adapter* specifies the network interface adapter (see 2.6 Structures and SphinxLib.h for the definition of struct ADAPTER_PARAM).
Parameter c_*func* specifies the discovery callback function.
Parameter *d_timeout* specifies a discovery timeout in ms.
Parameter *ignore_subnet* specifies if subnet ignore flag of discovery message should be set.
Parameter *port* specifies the destination port address of the control channel. Value of 0 set the port automatically.
Network byte order for ip addresses.

There are two methods of obtaining GigE Vision devices.
See GEVDiscovery.

Return: see Error Codes

Reference: GEVEnumerateAdapter, GEVCheckDeviceStatus

## GEVEnableFirewallException

Function: Enable firewall exception

Syntax:

```
GEVEnableFirewallException(char *app_name_with_path,
char *rule_name, BYTE *status);
```

Description: This function can enable a firewall exception for an application.
Parameter *app_name_with_path* specifies the application with path who wants to enable in the firewall.
Parameter *rule_name* specifies the rule name of the application.
Parameter *status* returns the status.

Possible values of the parameter status

```
#define FIREWALL_IS_DISABLED      1
#define FIREWALL_IS_ENABLED       2
#define FIREWALL_IS_ADDED         4
```

Example:

```
// get current application name with path
GetModuleFileName(NULL, AppPath, _MAX_PATH);

error = GEVEnableFirewallException(AppPath, "Test App", &status);
if (error == GEV_STATUS_SUCCESS)
{
 if ((status & FIREWALL_IS_DISABLED) == FIREWALL_IS_DISABLED)
  printf("[INFO] - Application is disabled in the firewall.");
  if ((status & FIREWALL_IS_ENABLED) == FIREWALL_IS_ENABLED)
```

```
   printf("[INFO] - Application is enabled in the firewall.");
  if ((status & FIREWALL_IS_ADDED) == FIREWALL_IS_ADDED)
   printf("[INFO] - Application is addedin to the firewall.");
}
else
{
  if (error == GEV_STATUS_ACCESS_DENIED)
   printf("Access denied when enable the application in the
           firewall.\n Run application as administrator");
  else
   printf("Can't enable the application in the firewall.
           Please check if firewall is enabled.");
}
```

| | |
|---|---|
| Return: | see Error Codes |
| Reference: | --- |

## GEVEnumerateAdapters

| | |
|---|---|
| Function: | Enumerate network interface adapter |
| Syntax: | `WORD GEVEnumerateAdapters(ENUMERATE_ADAPTER* adapter);` |
| Description: | This function can enumerate up to 50 network interface adapter.<br>Paramter *adapter* returns number and parameters of the found network interface adapters (see 2.6 Structures and SphinxLib.h for the definition of struct ENUMERATE_ADAPTER).<br>Network byte order for ip addresses. |
| Return: | see Error Codes |
| Reference: | GEVDiscoveryAdapter |

## GEVForceIp

| | |
|---|---|
| Function: | Force an IP address |
| Syntax: | `WORD GEVForceIp(DWORD new_ip, DWORD subnet, DWORD gateway, BYTE *mac, DWORD adapter_ip);` |
| Description: | This function temporary modifies the network configuration of a device.<br>Parameter *new_ip* sets IP-address, *subnet* sets subnet mask and *gateway* sets the gateway address.<br>Parameter *mac* is the MAC address of the device.<br>Parameter *adapter_ip* specifies the ip address of the network adapter where is connected the device.<br>Network byte order for new_ip, subnet and gateway parameter. |
| Return: | see Error Codes |
| Reference: | GEVSetNetConfig, GEVGetNetConfig |

## GEVGetChannelParameter

| | |
|---|---|
| Function: | Returns channel parameter |
| Syntax: | `WORD GEVGetChannelParameter(BYTE cam_nr, CHANNEL_PARAMETER *cparam);` |
| Description: | This function returns the control and stream channel parameter.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *cparam* specifies the channel parameter (see 2.6 Structures and SphinxLib.h for definition of CHANNEL_PARAMETER). |

Return: see Error Codes

Reference: GEVSetChannelParameter

## GEVGetDetailedLog

Function: Gets detailed log

Syntax: `WORD GEVGetDetailedLog(BYTE cam_nr, BYTE *on_off);`

Description: This function returns the detailed log in the parameter on_off.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

Possible values of the parameter on_off

```
#define DETAILED_LOG_OFF       0
Disable all log outputs.

#define DETAILED_LOG_INFO      1
Enable information outputs

#define DETAILED_LOG_WARNING   2
Enable warning outputs

#define DETAILED_LOG_ERROR     4
Enable error outputs

#define DETAILED_LOG_REGISTER  8
Enable register read/write outputs

#define DETAILED_LOG_DEBUG     16
Enable debug outputs

#define DETAILED_LOG_VERBOSE   32
Enable verbose outputs
```

Example:

```
GEVGetDetailedLog(device, &detailed_log);

if((detailed_log & DETAILED_LOG_INFO) == DETAILED_LOG_INFO)
  printf("Info log  output is on\n");

if((detailed_log & DETAILED_LOG_ERROR) == DETAILED_LOG_ERROR)
  printf("Error log  output is on\n");
```

Return: see Error Codes

Reference: GEVSetDetailedLog

## GEVGetConnectionStatus

Function: Returns connection status

Syntax: `WORD GEVGetConnectionStatus(BYTE cam_nr, BYTE *status, BYTE *lib_status);`

Description: This function returns the connection status.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *status* is a pointer to a variable containing the connection status. 0 = OK, 1 = timeout, 2 = access denied.
Parameter lib_status is a pointer to a variable containing the evaluation status. 0 = OK, 1 = evaluation expired (deprecated), 2 = library is running in evaluation mode (non S2I devices display horizontal stripes).

Return: see Error Codes

Reference: ---

## GEVGetFilterDriverVersion

Function: Get S2I filter driver version

Syntax: `WORD GEVGetFilterDriverVersion(BYTE cam_nr, BYTE *version_major, BYTE *version_minor);`

Description: This function returns the S2I filter driver version of the specified device (cam_nr [1..50]). As the filter driver is bound to each device instance, the parameter *cam_nr* is needed, although no communication to the device is done.
Parameter *version_major* is a pointer to a variable containing the major version.
Parameter *version_minor* is a pointer to a variable containing the minor version.
If the filter driver is not activated the return value in *version_major* and *version_minor* is 0.

Return: see Error Codes

Reference: GEVInitFilterDriver, GEVCloseFilterDriver

## GEVGetHeartbeatRate

Function: Returns heartbeat rate

Syntax: `WORD GEVGetHeartbeatRate(BYTE cam_nr, DWORD *heartbeat_rate);`

Description: This function returns heartbeat rate in ms.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *heartbeat_rate* is a pointer to a variable containing the heartbeat rate.

Return: see Error Codes

## GEVGetMemorySize

Function: Get image memory size

Syntax: `WORD GEVGetMemorySize(BYTE cam_nr, DWORD *mem_size);`

Description: This function returns the image acquisition memory-size.
Parameter *mem_size* is a pointer to a variable containing the memory size.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

Return: see Error Codes

Reference: GEVSetMemorySize

## GEVGetNetConfig

Function: Get network configuration

Syntax: `WORD GEVGetNetConfig(BYTE cam_nr, BYTE *dhcp, DWORD *ip, DWORD *subnet, DWORD *gateway);`

Description: This function reads the network configuration of the device.
Parameter *cam_nr* specifies the GigE Vision device instance.
Parameter *dhcp* returns dhcp enable status, *ip* returns IP-address,

*subnet* returns subnet mask and *gateway* returns the gateway address. Network byte order for ip, subnet and gateway parameter.

Return: see Error Codes

Reference: GEVSetNetConfig

## GEVGetPixelPtr (Deprecated)

Function: Returns the logical address of the image memory

Syntax: `WORD GEVGetPixelPtr(BYTE cam_nr, DWORD offset, UINT64 *ptr_adr);`

Description: This function returns the logical memory address of the local image buffer. This function is deprecated, it is recommended to use *GEVGetImageBuffer*.
Parameter *offset* shifts the returned start of the image buffer. It can be used when working with memory divided in several pages for different images.
Pointer *ptr_adr* gives access to the pixeldata.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Example:

GEVGetPixelPtr(1,0, &ptradr);
GPixel = (BYTE *)ptradr;

Return: see Error Codes

Reference: GEVGetImage, GEVGetImageBuffer

## GEVGetReadWriteParameter

Function: Returns read/write parameter

Syntax: `WORD GEVGetReadWriteParameter(BYTE cam_nr, DWORD *ack_timeout, BYTE *retry_count);`

Description: This function returns read/write parameter for the *GEVReadRegister*, *GEVWriteRegister*, *GEVReadMemory* and *GEVWriteMemory* functions.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *ack_timeout* is a pointer to a variable containing the acknowledge timeout in ms.
Parameter *retry_count* is a pointer to a variable containing the retry count for the read/write commands.

Return: see Error Codes

Reference: GEVSetReadWriteParameter

## GEVInit

Function: Initialize a GigE Vision device

Syntax: `WORD GEVInit(BYTE cam_nr, CONNECTION *con, ERROR_CALLBACK_FUNC error_func, BYTE save_xml, BYTE open_mode);`

Description: This function opens a control channel to a GigE Vision device. In addition the heartbeat thread is started.
With *cam_nr* an instance number for the camera is assigned [1..50].
Parameter *con* specifies the connection data (see 2.6 Structures and SphinxLib.h for definition of CONNECTION).

Parameter *error_func* describes the error callback function. The error function will be executed, when an error occured.
Parameter *save_xml* specifies the flag to save the xml file to disk. Directory is fixed to the location of the library.
Parameter *open_mode* specifies the control privileges of the device. Network byte order for ip addresses.

```
#define OPEN_ACCESS       0
#define EXCLUSIVE_ACCESS  1
#define CONTROL_ACCESS    2
```

Return: see Error Codes

Reference: GEVClose, GEVGetLocalError

## GEVInitFilterDriver

Function: Init S2I filter driver

Syntax:

```
WORD GEVInitFilterDriver(BYTE cam_nr);
```

Description: This function activates the S2I stream channel filter driver for the specified device (*cam_nr* [1..50]).

Return: see Error Codes

Reference: GEVCloseFilterDriver, GEVGetFilterDriverVersion

## GEVOpenStreamChannel

Function: Open GigE Vision device stream channel

Syntax:

```
WORD GEVOpenStreamChannel(BYTE cam_nr, DWORD ip, WORD
port, DWORD multicast);
```

Description: This function opens the stream channel of the specified GigE Vision device (*cam_nr* [1..50]).
Parameter *ip* specifies the destination ip address of the stream channel.
Parameter *port* specifies the destination port address of the stream channel.
Parameter *multicast* specifies the multicast ip address of the stream channel, address 0.0.0.0 disables multicast.

Return: see Error Codes

Reference: GEVCloseStreamChannel

## GEVReadRegister

Function: Receive data from a GigE Vision device

Syntax:

```
WORD GEVReadRegister(BYTE cam_nr, DWORD cmd, BYTE
cnt, DWORD *pptr);
```

Description: This function reads 32bit data from the specified GigE Vision device (*cam_nr* [1..50]). This function implements the GEV Read_Reg command.

The register address is set in *cmd*.
The number of 32bit reads is set in *cnt*.
The parameters itself are accessible through the pointer *pptr*

Return: see Error Codes

Reference: GEVSetReadWriteParameter ,GEVWriteRegister

## GEVReadMemory

Function: Receive memory data from a GigE Vision device

Syntax:

```
WORD GEVReadMemory(BYTE cam_nr, DWORD maddr, DWORD
cnt, DWORD *pptr);
```

Description: This function reads a memory block from the specified GigE Vision device (*cam_nr* [1..50]). This function implements the GEV Read_Mem command.
The memory address is given in *maddr*.
The number of bytes to read is set in *cnt*.
The memory data itself is accessible through the pointer *pptr*
To get the write status in percent use the *GEVSetReadWriteMemoryCallback* function to set a callback function.

Return: see Error Codes

Reference: GEVWriteMemory, GEVSetReadWriteMemoryCallback

## GEVSetActionCommand

Function: Set action command

Syntax:

```
WORD GEVSetActionCommand(BYTE cam_nr, DWORD
device_key, DWORD group_key, DWORD group_mask, DWORD
action_time);
```

Description: This function sets the action command.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *device_key* specifies the device key to authorize the action on this device.
Parameter *group_key* specifies the group key to define a group of devices on which the actions have to be executed.
Parameter *group_mask* specifies the group mask to be used to filter out some of these devices from the group.

Return: see Error Codes

Reference: GEVS

## GEVSetChannelParameter

Function: Set channel parameter

Syntax:

```
WORD GEVSetChannelParameter(BYTE cam_nr,
CHANNEL_PARAMETER cparam);
```

Description: This function sets the control and stream channel parameter.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *cparam* specifies the channel parameter (see 2.6 Structures and SphinxLib.h for definition of CHANNEL_PARAMETER).

Return: see Error Codes

Reference: GEVGetChannelParameter

## GEVSetDetailedLog

Function: Sets detailed log

Syntax:

```
WORD GEVSetDetailedLog(BYTE cam_nr, BYTE on_off);
```

Description: This function sets the detailed log with the parameter on_off.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

Possible values of the parameter on_off

```
#define DETAILED_LOG_OFF       0
Disable all log outputs.

#define DETAILED_LOG_INFO      1
Enable information outputs

#define DETAILED_LOG_WARNING   2
Enable warning outputs

#define DETAILED_LOG_ERROR     4
Enable error outputs

#define DETAILED_LOG_REGISTER  8
Enable register read/write outputs

#define DETAILED_LOG_DEBUG     16
Enable debug outputs

#define DETAILED_LOG_VERBOSE   32
Enable verbose outputs
```

Example: Enable info and error log outputs

```
GEVSetDetailedLog(device, DETAILED_LOG_INFO | DETAILED_LOG_ERROR);
```

Return: see Error Codes

Reference: GEVGetDetailedLog

## GEVSetHeartbeatRate

Function: Set heartbeat rate

Syntax:

```
WORD GEVSetHeartbeatRate(BYTE cam_nr, DWORD
heartbeat_rate);
```

Description: This function sets the heartbeat timeout register and the timeout value for connection checking.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *heartbeat_rate* is the heartbeat timeout value in ms. The heartbeat timeout register of the device is set to this value. The interval value to check the connection state is set to half of the heartbeat timeout value. With the function *GEVGetChannelParameter* you can read the interval value for the connection check. (CHANNEL_PARAMETER cc_heartbeat_timeout.)

Return: see Error Codes

## GEVSetMemorySize (Deprecated)

Function: Set image memory size

Syntax:

```
WORD GEVSetMemorySize(BYTE cam_nr, DWORD mem_size);
```

Description: This function reallocates memory as framebuffer. The reference to the pixel data must be updated with get_pixel_ptr!
Parameter *mem_size* is the memory size in bytes.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

This function is deprecated and is no longer needed if image buffer is handled in user application and provided to the library via *GEVGetImageBuffer*.

Return: see Error Codes

Reference: GEVGetMemorySize, GEVGetPixelPtr, GEVGetImageBuffer

### GEVSetMessageChannel

Function: Bind function to message channel event

Syntax: `WORD GEVSetMessageChannel(BYTE cam_nr, unsigned short port, MESSAGECHANNEL_CALLBACK_FUNC c_func);`

Description: This function binds a custom function to the message channel, which will be executed, when an event occurred.
Parameter *cam_nr* specifies the GigE Vision device instance.
Parameter *port* specifies the message channel port to open.
Parameter *c_func* describes the callback function.
`BYTE (WINAPI *MESSAGECHANNEL_CALLBACK_FUNC )(BYTE cam_nr, int event_id, int data_length, BYTE *data);`

Return: see Error Codes

Reference: -

### GEVSetNetConfig

Function: Set network configuration

Syntax: `WORD GEVSetNetConfig(BYTE cam_nr, BYTE dhcp, DWORD ip, DWORD subnet, DWORD gateway);`

Description: This function modifies the network configuration of a GigE Vision device.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *dhcp* sets the dhcp enable status, *ip* sets IP-address, *subnet* sets the subnet mask and *gateway* sets the gateway address.
Network byte order for ip, subnet and gateway parameter.

Return: see Error Codes

Reference: GEVGetNetConfig

### GEVSetReadWriteMemoryCallback

Function: Associate function with read/write memory functions

Syntax: `WORD GEVSetReadWriteMemoryCallback(BYTE cam_nr, READ_WRITE_MEM_CALLBACK_FUNC c_func);`

Description: This function associates a custom function to get the read/write status in percent.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *c_func* describes the callback function.
`BYTE (WINAPI *READ_WRITE_MEM_CALLBACK_FUNC )(BYTE cam_nr, int s_cnt);`

Return: see Error Codes

Reference: GEVReadMemory, GEVWriteMemory

## GEVSetReadWriteParameter

Function: Sets read/write parameter

Syntax: `WORD GEVSetReadWriteParameter(BYTE cam_nr, DWORD ack_timeout, BYTE retry_count);`

Description: This function sets the read/write parameter for the parameter for the *GEVReadRegister*, *GEVWriteRegister*, *GEVReadMemory* and *GEVWriteMemory* funtions
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *ack_timeout* is the acknowledge timeout in ms.
Parameter *retry_count* is the number of retries if read/write command wasn't acknowledged.

Return: see Error CodesIT

Reference: GEVGetReadWriteParameter

## GEVTestPacket

Function: Trigger GigE Vision test packet

Syntax: `WORD GEVTestPacket(BYTE cam_nr, DWORD *packet_size);`

Description: This function triggers the specified GigE Vision device (*cam_nr* [1..50]) to send a test packet. This feature can be used to find maximum packet size in a network environment.
Parameter *packet_size* returns the packet size of the test packet. Size of the test packet is equal to the stream channel packet size setting.

Return: see Error Codes

Reference: GEVTestFindMaxPacketSize

## GEVSetTraversingFirewallsInterval

Function: Set the interval in seconds for traversing firewalls

Syntax: `WORD GEVSetTraversingFirewallsInterval(BYTE cam_nr, DWORD interval);`

Description: This function sets the interval in seconds for traversing firewalls to the specified GigE Vision device (*cam_nr* [1..50]).
The parameter *interval* is the time in seconds between the packets to be sent to the stream channel. Traversing firewalls is only available when the stream channel source port register is available in the device.

Return: see Error Codes

Reference: GEVOpenStreamChannel

## GEVWriteMemory

Function: Send memory data to a GigE Vision Device device

Syntax: `WORD GEVWriteMemory(BYTE cam_nr, DWORD maddr, DWORD cnt, DWORD *pptr);`

Description: This function writes a memory block to the specified GigE Vision device device (*cam_nr* [1..50]). This function implements the GEV Write_Mem command.

The memory address is given in *maddr*.
The number of bytes to write is set in *cnt*.
The data itself is accessible through the pointer *pptr*
To get the write status in percent use the *GEVSetReadWriteMemoryCallback* function to set a callback function.

Return: see Error Codes

Reference: GEVReadMemory, GEVSetReadWriteMemoryCallback

## GEVWriteRegister

Function: Send register data to GigE Vision Device device

Syntax:

```
WORD GEVWriteRegister(BYTE cam_nr, DWORD cmd, BYTE
cnt, DWORD *pptr);
```

Description: This function writes 32bit data to the specified GigE Vision device (*cam_nr* [1..50]). This function implements the GEV Write_Reg command.

The register address is set in *cmd*.
The number of 32bit words to be written is set in *cnt*.
The parameters itself are accessible through the pointer *pptr*.

Example:

```
error = GEVWriteRegister(CAM_0,
                         0xA100,
                         1,
                         &reg_data);
```

Return: see Error Codes

Reference: GEVReadRegister

## 2.2. XML Handling

### 2.2.1. Introduction

Part of the GigE Vision standard is that each device has to provide a xml-file, which describes the functions of the device. This xml-file contains a set of nodes, where each node has a type and a unique name. Nodes can be linked to others and each connection plays a certain role. The syntax and elements of this xml-file is described in the GenICam GenAPI standard document available at www.genicam.org.

The S2I-software, which evaluates the xml file has some restrictions, so not all features of the xml are supported right now.

**Supported interface and node types:**

- Integer (IntReg, MaskIntReg, Integer)
- String (StringReg)
- Command
- Boolean
- Category
- Port
- Enumeration
- StructReg
- SwissKnife
- IntSwissKnife
- Float (FloatReg)
- EnumEntry
- pVariable
- Formula
- FormulaTo
- FormulaFrom
- Converter
- IntConverter
- Register

**Supported node elements:**

- Address
- pAddress
- Value
- pValue
- pVariable
- Min
- pMin
- Max
- pMax
- CommandValue
- AccessMode
- OnValue
- OffValue
- LSB
- MSB
- Bit
- Inc

- pInc
- Representation
- Endianess
- Length
- Visibility
- pIsImplemented
- pIsAvailable
- DisplayName
- ToolTip
- Unit
- pIsLocked
- Sign
- pIndex
- pValueIndexed
- ValueIndexed
- ValueDefault
- pValueDefault
- pValueCopy
- Slope
- pOffset
- pLength
- DisplayNotation
- DisplayPrecision
- pSelected

That means only nodes and elements listed above are accessible through following functions.

From application side it is also possible to use the GenICam reference implementation for xml handling. Please refer to sample application for.

### *2.2.2. XML-Functions*

#### GEVGetFeatureBoolean

| | |
|---|---|
| Function: | Returns the Boolean value of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureBoolean(BYTE cam_nr, char *feature_name, DWORD *bool_value);` |
| Description: | This functions returns the Boolean value of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains the feature name (e.g. LUTEnable)<br>Parameter *bool_value* returns the Boolean value. |
| Return: | see Error Codes |
| Reference: | GEVSetFeatureBoolean, GEVGetFeatureParameter |

#### GEVGetFeatureCommand

| | |
|---|---|
| Function: | Returns the command value of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureCommand(BYTE cam_nr, char *feature_name, DWORD *cmd_value);` |
| Description: | This function returns the command value of a feature descripted in xml-file. |

| | |
|---|---|
| | Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature node name (e.g. AcquisitionStart).<br>Parameter *cmd_value* returns the command value. |
| Return: | see Error Codes |
| Reference: | GEVSetFeatureCommand, GEVGetFeatureParameter |

## GEVGetFeatureDisplayName

| | |
|---|---|
| Function: | Returns the display name of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureDisplayName(BYTE cam_nr, char *feature_name, char *display_name, int display_name_length);` |
| Description: | This function returns the display name of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Width)<br>Parameter *display_name* returns the display name.<br>Parameter *display_name_length* contains length of the display name pointer. |
| Return: | see Error Codes |
| Reference: | GEVGetFeatureTooltip |

## GEVGetFeatureEnableStatus

| | |
|---|---|
| Function: | Returns the enable status of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureEnableStatus(BYTE cam_nr, char *feature_name, BYTE *enable);` |
| Description: | This function returns the access status of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Width)<br>Parameter *enable* returns the status. |
| Return: | see Error Codes |
| Reference: | -- |

## GEVGetFeatureEnumeration

| | |
|---|---|
| Function: | Returns the enumeration name of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureEnumeration(BYTE cam_nr, char *feature_name, char *enum_name, int str_len);` |
| Description: | This function returns the current enumeration value of an enumeration feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Pixelformat)<br>Parameter *enum_name* returns feature enumeration value.<br>Parameter *str_len* contains length of feature name pointer. |
| Return: | see Error Codes |
| Reference: | GEVSetFeatureEnumeration , GEVGetFeatureEnumerationName, GEVGetFeatureParameter |

## GEVGetFeatureEnumerationName

Function: Returns the name of an enumeration value

Syntax:

```
WORD GEVGetFeatureEnumerationName(BYTE cam_nr, char
*feature_name, BYTE enum_index, char *enum_name, int
str_len);
```

Description: This function returns an enumeration value indexed by enum_index of a feature descripted in xml-file.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *feature_name* contains feature name (e.g. Pixelformat)
Parameter *enum_index* contains enumeration index.
Parameter *enum_name* returns feature name of enumeration name. (e.g. Mono8)
If returned string length of *enum_name* is equal 0, then enumeration name is not available.
Parameter *str_len* contains length of feature name pointer.
For a complete list of possible enum values call *GEVGetFeatureParameter*.

Return: see Error Codes

Reference: GEVSetFeatureEnumeration, GEVGetFeatureEnumeration, GEVGetFeatureParameter

## GEVGetFeatureFloat

Function: Returns the float value of a dedicated feature, descripted in xml file

Syntax:

```
WORD GEVGetFeatureFloat(BYTE cam_nr, char
*feature_name, float *float_value);
```

Description: This function returns the float value of a feature descripted in xml-file.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *feature_name* contains the feature name (e.g. DeviceTemperature)
Parameter *float_value* returns the float value.

Return: see Error Codes

Reference: GEVSetFeatureFloat

## GEVGetFeatureInteger

Function: Returns the integer value of a dedicated feature, descripted in xml file

Syntax:

```
WORD GEVGetFeatureInteger(BYTE cam_nr, char
*feature_name, DWORD *int_value);
```

Description: This function returns the integer value of a feature descripted in xml-file.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *feature_name* contains the feature name (e.g. Width)
Parameter *int_value* returns feature integer value.

Return: see Error Codes

Reference: GEVSetFeatureInteger, GEVGetFeatureParameter

## GEVGetFeatureInvalidator

| | |
|---|---|
| Function: | Returns the invalidator of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureInvalidator(BYTE cam_nr, char *feature_name, BYTE index, char *invalidator_name, int str_len);` |
| Description: | This function returns the invalidator of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Width)<br>Parameter *index* contains the invalidator index. To obtain the total number of invalidators, use the function *GEVGetFeatureParameter*.<br>Parameter *invalidator_name* returns invalidator name.<br>Parameter *str_len* contains length of invalidator_name pointer. |
| Return: | see Error Codes |
| Reference: | GEVGetFeatureParameter |

## GEVGetFeatureList

Function: Returns the pointer of a dedicated feature list, descripted in xml file

Syntax: `WORD GEVGetFeatureList(BYTE cam_nr, FeatureListPtr *featureListPtr, BYTE *maxLevel);`

Description: This function returns a pointer to the feature list descripted in xml-file.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *featureListPtr* contains the feature list.
XMLs have a hierarchical structure.
Parameter *maxLevel* returns the maximal level to list.

```
error = GEVGetFeatureList(camera,&featureListPtr,
&maxLevel);
if(featureListPtr == NULL)
   return;

while(featureListPtr != NULL)
{
 for(i = 0; i < featureListPtr->Level;i++)
   printf("\t");
 printf("%s\n",featureListPtr->Name);
 featureListPtr = featureListPtr->Next;
}
```

Return: see Error Codes

Reference: ---

## GEVGetFeatureParameter

| | |
|---|---|
| Function: | Returns properties of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureParameter(BYTE cam_nr, char *feature_name, FEATURE_PARAMETER *f_param);` |
| Description: | This function returns properties of a device feature descripted in xml-file.<br>Parameter *cam*_nr specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Width)<br>Paramter *f_param* returns the parameters of the given feature (see 2.6 Structures and Sphinxlib.h for definition of struct FEATURE_PARAMETER). |

Example:

```
error = GEVGetFeatureParameter(camera,
                        feature_str,
                        &f_param);
printf(„Feature-Value-Min: %d\n“,f_param.Min);
printf(„Feature-Value-Max: %d\n“,f_param.Max);
…
```

| | |
|---|---|
| Return: | see Error Codes |
| Reference: | --- |

## GEVGetFeaturePort

| | |
|---|---|
| Function: | Returns the port of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeaturePort(BYTE cam_nr, char *feature_name, char *port_name, int port_name_length);` |
| Description: | This function returns the port of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Width)<br>Parameter port_*name* returns port name.<br>Parameter port_*name_length* contains length of port_name pointer. |
| Return: | see Error Codes |
| Reference: | --- |

## GEVGetFeatureRegister

| | |
|---|---|
| Function: | Returns the values of a dedicated register feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureRegister(BYTE cam_nr, char *feature_name, DWORD len, BYTE *pbuffer);` |
| Description: | This function returns values of a register feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. LUTValueAll)<br>Parameter *len* contains length of buffer/register.<br>Parameter *pbuffer* returns buffer values. |
| Return: | see Error Codes |
| Reference: | GEVSetFeatureRegister, GEVGetFeatureParameter |

## GEVGetFeatureString

| | |
|---|---|
| Function: | Returns the string value of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureString(BYTE cam_nr, char *feature_name, char *str_value);` |
| Description: | This function returns the string value of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. DeviceVersion)<br>Parameter *str_value* returns feature string value. |
| Return: | see Error Codes |
| Reference: | GEVSetFeatureString, GEVGetFeatureParameter |

## GEVGetFeatureTooltip

| | |
|---|---|
| Function: | Returns the tooltip string of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureTooltip(BYTE cam_nr, char *feature_name, char *tooltip_name, int tooltip_name_length);` |
| Description: | This function returns the tooltip string of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Width)<br>Parameter *tooltip_name* returns tooltip string.<br>Parameter *tooltip_name_length* contains length of tooltip_name pointer. |
| Return: | see Error Codes |
| Reference: | GEVGetFeatureDisplayName |

## GEVGetFeatureUnit

| | |
|---|---|
| Function: | Returns unit string of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVGetFeatureUnit(BYTE cam_nr, char *feature_name, char *unit_name, int unit_name_length);` |
| Description: | This function returns a unit string of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. DeviceTemperature)<br>Parameter *unit_name* returns unit string (e.g. C).<br>Parameter *unit_name_length* contains length of unit_name pointer. |
| Return: | see Error Codes |
| Reference: | --- |

## GEVGetXmlFile

| | |
|---|---|
| Function: | Get the xml file |
| Syntax: | `WORD GEVGetXmlFile(BYTE cam_nr, BYTE **xmlfile);` |
| Description: | This function returns the xml-file of the device. Before using this function, call GEVGetXmlSize to get the size of the xml file and allocate memory (*xmlfile*).<br>Parameter *cam_*nr specifies the GigE Vision device instance [1..50]. |
| Return: | see Error Codes |
| Reference: | GEVGetXmlSize |

## GEVGetXmlSize

| | |
|---|---|
| Function: | Get the size of the deivice xml file |
| Syntax: | `WORD GEVGetXmlSize(BYTE cam_nr, DWORD *size);` |
| Description: | This function returns the size of the xml-file of the device. The xml file size is stored in the parameter *size*. Call this function before using GEVGetXmlFile.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50]. |
| Return: | see Error Codes |

| | |
|---|---|
| Reference: | GEVGetXmlFile |

## GEVInitXml

| | |
|---|---|
| Function: | Initialize xml |
| Syntax: | `WORD GEVInitXml(BYTE cam_nr);` |
| Description: | This function gets the xml-file from the device and builds the feature list in the library.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>This function also needs the schema files in the library/program path.<br>The schema files can you find in the SphinxLib\Extra directory. |
| Return: | see Error Codes |
| Reference: | GEVSetXmlFile, GEVSetSchemaPath |

## GEVSetFeatureBoolean

| | |
|---|---|
| Function: | Set the Boolean value of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVSetFeatureBoolean(BYTE cam_nr, char *feature_name, DWORD bool_value);` |
| Description: | This function writes the Boolean value of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. LUTEnable)<br>Parameter *bool_value* contains new Boolean value. |
| Return: | see Error Codes |
| Reference: | GEVGetFeatureBoolean, GEVGetFeatureParameter |

## GEVSetFeatureCommand

| | |
|---|---|
| Function: | Set command value of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVSetFeatureCommand(BYTE cam_nr, char *feature_name, DWORD cmd_value);` |
| Description: | This function writes the command value of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. AcquisitionStart)<br>Parameter *cmd_value* contains command value. |
| Return: | see Error Codes |
| Reference: | GEVGetFeatureCommand, GEVGetFeatureParameter |

## GEVSetFeatureEnumeration

| | |
|---|---|
| Function: | Set enumeration name of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVSetFeatureEnumeration(BYTE cam_nr, char *feature_name, char *enum_name, int str_len);` |
| Description: | This function writes an enumeration value to a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. Pixelformat)<br>Parameter *enum_name* contains the feature’s enumeration name (e.g. |

Mono8).
Parameter *str_len* contains length of feature name pointer.

Return: see Error Codes

Reference: GEVGetFeatureEnumeration , GEVGetFeatureEnumerationName, GEVGetFeatureParameter

## GEVSetFeatureFloat

Function: Set float value of a dedicated feature, descripted in xml file

Syntax:

```
WORD GEVSetFeatureFloat(BYTE cam_nr, char
*feature_name, float float_value);
```

Description: This function writes a float value to a feature descripted in xml-file.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *feature_name* contains feature name (e.g. ExposureTime)
Parameter *float_value* contains new float value.

Return: see Error Codes

Reference: GEVGetFeatureFloat

## GEVSetFeatureInteger

Function: Set integer value of a dedicated feature, descripted in xml file

Syntax:

```
WORD GEVSetFeatureInteger(BYTE cam_nr, char
*feature_name, DWORD int_value);
```

Description: This function writes an integer value of a feature descripted in xml-file.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *feature_name* contains feature name (e.g. Width)
Parameter *int_value* contains the new integer value.

Return: see Error Codes

Reference: GEVGetFeatureInteger, GEVGetFeatureParameter

## GEVSetFeatureRegister

Function: Set register values of a dedicated feature, descripted in xml file

Syntax:

```
WORD GEVSetFeatureRegister(BYTE cam_nr, char
*feature_name, DWORD len, BYTE *pbuffer);
```

Description: This function writes buffer values to a register feature descripted in xml-file.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *feature_name* contains feature name (e.g. LUTValueAll)
Parameter *len* contains the length of buffer.
Parameter *pbuffer* contains the new buffer values.

Return: see Error Codes

Reference: GEVGetFeatureRegister, GEVGetFeatureParameter

## GEVSetFeatureString

| | |
|---|---|
| Function: | Set string value of a dedicated feature, descripted in xml file |
| Syntax: | `WORD GEVSetFeatureString(BYTE cam_nr, char *feature_name, char *str_value);` |
| Description: | This function writes the string value of a feature descripted in xml-file.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *feature_name* contains feature name (e.g. DeviceUserID)<br>Parameter *str_value* contains the new string value. |
| Return: | see Error Codes |
| Reference: | GEVGetFeatureInteger, GEVGetFeatureParameter |

## GEVSetSchemaPath

| | |
|---|---|
| Function: | Set the path of the schema files |
| Syntax: | `WORD GEVSetSchemaPath(BYTE cam_nr, char *schema_path);` |
| Description: | This function sets the new path of the schema files.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *schema_path* contains the new path of the schema files.<br>Default path of the schema files are current program/library path.<br>This function must be called after GEVInit function and before the GEVInitXml function. |
| Return: | see Error Codes |

Reference: GEVInitXml

## GEVSetXmlFile

| | |
|---|---|
| Function: | Set xml file in the library |
| Syntax: | `WORD GEVSetXmlFile(BYTE cam_nr, char *xml_name);` |
| Description: | This function sets the xml-file in the library.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *xml_name* contains path to new xml file to read and use in the library. |
| Return: | see Error Codes |

Reference: GEVInitXml

## 2.3. Camera functions

### GEVAcquisitionStart

Function: Start image acquisition

Syntax: `WORD GEVAcquisitionStart(BYTE cam_nr , DWORD number_images_to_acquire);`

Description: This function starts the image acquisition and writes network image data to a PC ringbuffer. Transfer of images to image memory has to be done by *GEVGetImage* (deprecated) or *GEVGetImageBuffer*.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *number_images_to_acquire* contains the number of images to grab. This parameter is used only, if AcquisitionMode is set to MultiFrame and node AcquisitionFrameCount is not available. After that number, GEVAcquisitionStop is called automatically.

| AcquisitionMode | number_images_to_acquire |
|---|---|
| Continuous | 0 (unlimited) |
| SingleFrame | 1 |
| MultiFrame | 1..n<br>Get the value from the AcquisitionFrameCount node |

If GEV_STATUS_FEATURE_NOT_AVAILABLE error is returned, please check if one of these entries is not in the XML file.

- AcquisitionMode (if Multi is set then also Acquisition FrameCount should be present)
- Width
- Height
- PixelFormat
- PayloadSize
- AcquisitionStart

Return: see Error Codes

Reference: GEVAcquisitionStop, GEVGetImage

### GEVAcquisitionStartEx

Function: Start image acquisition without library internal xml handling

Syntax: `WORD GEVAcquisitionStartEx(BYTE cam_nr , DWORD number_images_to_acquire, DWORD image_size);`

Description: This function initializes an image ring buffer and its parameters. It starts the image acquisition thread. Transfer of images to image memory has to be done by *GEVGetImage* (deprecated) or *GEVGetImageBuffer*.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *number_images_to_acquire* contains the number of images to grab. This parameter is used only, if AcquisitionMode is set to MultiFrame and node AcquisitionFrameCount is not available. After that

number GEVAcquisitionStop is called automatically.
Parameter *image_size* contains the size of one payload block (image).

Use this function if xml handling is done outside of sphinxlib. Do not forget to send acquisition start command to the device.

| AcquisitionMode | number_images_to_acquire |
|---|---|
| Continuous | 0 (unlimited) |
| SingleFrame | 1 |
| MultiFrame | 1..n |

| | |
|---|---|
| Return: | see Error Codes |
| Reference: | GEVAcquisitionStart , GEVAcquisitionStop, GEVGetImage |

## GEVAcquisitionStop

| | |
|---|---|
| Function: | Stop image acquisition |
| Syntax: | `WORD GEVAcquisitionStop(BYTE cam_nr);` |
| Description: | This function stops the image acquisition tread and sends acquisition stop command to the device, if xml handling is done by sphinxlib.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50]. |
| Return: | see Error Codes |
| Reference: | GEVAcquisitionStart, GEVGetImage |

## GEVGetBufferCount

| | |
|---|---|
| Function: | Returns number of buffers of library managed ring buffer |
| Syntax: | `WORD GEVGetBufferCount(BYTE cam_nr, WORD *count);` |
| Description: | Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>The parameter *count* returns the number of buffers of the image ring buffer. |
| Return: | see Error Codes |
| Reference: | GEVSetBufferCount |

## GEVGetImage (Deprecated)

| | |
|---|---|
| Function: | Get an image from internal ring buffer and copy it to internal lib memory |
| Syntax: | `WORD GEVGetImage(BYTE cam_nr, IMAGE_HEADER *image_header)` |
| Description: | Get an image form the ring buffer and copy it to the internal image memory.<br>The pointer to the internal image memory you can get it with the function *GEVGetPixelPtr*.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *image_header* returns information of the image (see 2.6 Structures and SphinxLib.h for definition of IMAGE_HEADER).<br>This function is deprecated and is no longer needed if image buffer is |

| | |
|---|---|
| | handled in user application and provided to the library via *GEVGetImageBuffer*. |
| Return: | see Error Codes |
| Reference: | GEVAcquisitionStart, GEVAcquisitionStop, GEVGetPixelPtr |

## GEVGetImageBuffer

| | |
|---|---|
| Function: | Get an image from the library managed ring buffer and copy it to user memory |
| Syntax: | `WORD GEVGetImageBuffer(BYTE cam_nr, IMAGE_HEADER *image_header, BYTE *image_buffer)` |
| Description: | Get an image form the ring buffer and transmit it to the *image_buffer* memory pointer. This has to be allocated by the application.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *image_header* returns information of the image (see 2.6 Structures and SphinxLib.h for definition of IMAGE_HEADER). |
| .Return: | see Error Codes |
| Reference: | GEVAcquisitionStart , GEVAcquisitionStop, |

## GEVGetImageRingBuffer

| | |
|---|---|
| Function: | Get an image from user managed ring buffer and return buffer index |
| Syntax: | `WORD GEVGetImageRingBuffer(BYTE cam_nr, IMAGE_HEADER *image_header, WORD *index)` |
| Description: | Get an image form the user ring buffer and return the *index* of buffer part. The ring buffer has to be allocated by the application.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *image_header* returns information of the image (see 2.6 Structures and SphinxLib.h for definition of IMAGE_HEADER).<br>Note: Not work with linux/mac filter driver, is not implemented in the linux/mac filter driver. |
| Return: | see Error Codes |
| Reference: | GEVAcquisitionStart , GEVAcquisitionStop, GEVSetRingBuffer, GEVQueueRingBuffer, GEVReleaseRingBuffer |

## GEVGetImageFPS

| | |
|---|---|
| Function: | Returns the number of the acquired frames per second |
| Syntax: | `WORD GEVGetImageFPS(BYTE cam_nr, double *fps);` |
| Description: | This function returns the number of acquired frames per second.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *fps* returns the frames per second. |
| Return: | see Error Codes |
| Reference: | --- |

## GEVGetPacketResend

| | |
|---|---|
| Function: | Gets packet resend activation status |
| Syntax: | `WORD GEVGetPacketResend(BYTE cam_nr, BYTE *enable);` |
| Description: | Parameter *cam_nr* specifies the GigE Vision device instance.<br>The parameter *enable* returns the packet resend activation status:<br>0 = disabled, 1 = enabled. |
| Return: | see Error Codes |
| Reference: | GEVSetPacketResend |

## GEVGetPacketsOutOfOrder

| | |
|---|---|
| Function: | Returns the number of handled packets out of order |
| Syntax: | `WORD WINAPI GEVGetPacketsOutOfOrder(BYTE cam_nr, BYTE *packets_out_of_order);` |
| Description: | Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *packets_out_of_order* returns the number of packets taken into account, before sending packet resend requests. |
| Return: | see Error Codes |
| Reference: | GEVSetPacketsOutOfOrder |

## GEVGetSecureTransfer (Depricated)

| | |
|---|---|
| Function: | Returns secure transfer mode enable status |
| Syntax: | `WORD GEVGetSecureTransfer(BYTE cam_nr, BYTE *enable);` |
| Description: | In Secure Transfer Mode packet resend is sent until the current block is complete. Next block shall be sent by the device not before receiving an acknowledge command.<br>This function returns the enable status of secure transfer mode.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *enable* is a pointer to a variable containing the enable status (0 = disable, 1 = enable).<br>Note: this function is no longer functional. |
| Return: | see Error Codes |
| Reference: | GEVSetSecureTransfer |

## GEVPacketResend

| | |
|---|---|
| Function: | Send packet resend command |
| Syntax: | `WORD GEVPacketResend(BYTE cam_nr, WORD stream_channel, WORD block_id, DWORD first_packet_id, DWORD last_packet_id)` |
| Description: | This function sends the packet resend command.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *stream_channel* contains the number of the stream channel. (0 is first stream channel)<br>Parameter *block_id* contains the block id.<br>Parameter *first_packet_id* contains the first packet to resend.<br>Parameter *last_packet_id* contains the last packet to resend. |

Example: packet 100 -105 of block 20 shall be resent:

```
GEVPacketResend(CAM_0, 0, 20, 100, 105);
```

Return: see Error Codes

Reference: GEVSetPacketResend, GEVGetPacketResend

## GEVQueueRingBuffer

Function: Queue an user managed ring buffer part for acquisition of new data

Syntax:

```
WORD GEVQueueRingBuffer(BYTE cam_nr, WORD
image_buffer_index);
```

Description: If data acquisition ring buffer is manged by user, a buffer part is blocked for user processing by *GEVGetImageRingBuffer*.
This function releases this user ring buffer part with index *image_buffer_index* for acquisition of new data.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

Note: Not work with linux/mac filter driver, is not implemented in the linux/mac filter driver.

Return: see Error Codes

Reference: GEVSetRingBuffer, GEVReleaseRingBuffer, GEVImageRingBuffer

## GEVReleaseRingBuffer

Function: Decouple user managed ring buffer from the library

Syntax:

```
WORD GEVReleaseRingBuffer(BYTE cam_nr);
```

Description: This function decouples the user ring buffer from the library.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

Note: Not work with linux/mac filter driver, is not implemented in the linux/mac filter driver.

Return: see Error Codes

Reference: GEVSetRingBuffer, GEVQueueRingBuffer, GEVImageRingBuffer

## GEVSetBufferCount

Function: Sets number of buffers in library managed ring buffer

Syntax:

```
WORD GEVSetBufferCount(BYTE cam_nr, WORD count);
```

Description: This function sets the number of buffers (*count*) in the library managed ring buffer. Default setting is 4 buffers.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

Return: see Error Codes

Reference: GEVGetBufferCount

## GEVSetPacketResend

Function: Enables or disables packet resend feature

Syntax:

```
WORD GEVSetPacketResend(BYTE cam_nr, BYTE enable);
```

Description: This function enables or disables the packet resend feature.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
The parameter *enable* sets the packet resend status:
0 = disable, 1 = enable.

Return: see Error Codes

Reference: GEVGetPacketResend

## GEVSetPacketsOutOfOrder

Function: Sets the number of handled out of order packets

Syntax:

```
WORD WINAPI SEVGetPacketsOutOfOrder(BYTE cam_nr, BYTE
packets_out_of_order);
```

Description: Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *packets_out_of_order* sets the number of packets taken into account, before sending packet resend requests. This function is only effective if packet resend is enabled.

Return: see Error Codes

Reference: GEVGetPacketsOutOfOrder

## GEVSetRingBuffer

Function: Couple user managed ring buffer with the library

Syntax:

```
WORD GEVSetRingBuffer(BYTE cam_nr, WORD index, void
*buffer);
```

Description: This function sets user ring buffer in the library.
Parameter *index* contains the index for the user ring buffer.
Parameter *buffer* contains the user ring buffer to set in the library.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].

Note: Not work with linux/mac filter driver, is not implemented in the linux/mac filter driver.

Return: see Error Codes

Reference: GEVReleaseRingBuffer, GEVQueueRingBuffer, GEVImageRingBuffer

## GEVSetSecureTransfer (Depricated)

Function: Enable secure transfer mode

Syntax:

```
WORD GEVSetSecureTransfer(BYTE cam_nr, BYTE enable,
SECURE_TRANSFER_CALLBACK_FUNC c_func);
```

Description: In Secure Transfer Mode packet resend is sent until the current block is complete. Next block shall be sent by the device not before receiving an acknowledge command.
This function sets the secure transfer mode.
Parameter *cam_nr* specifies the GigE Vision device instance [1..50].
Parameter *enable* enables or disables (0 = disable, 1 = enable) the secure transfer mode.
Parameter *c_func* describes the callback function for secure transfer acknowledge.

Note: this function is no longer functional.

Example:

```
error = GEVGetFeatureEnumeration(pparams->device,
"TransferOperationMode",s1 , sizeof(s1));

if(error == GEV_STATUS_SUCCESS)
{
  if(strcmp(s1,"Secure") == 0)
  {
    // set secure transfer in the SphinxLib
    GEVSetSecureTransfer(pparams->device, 1,
secure_transfer_callback_func);

    secure_transfer = 1;
  }
  else
    secure_transfer = 0;
}
BYTE WINAPI secure_transfer_callback_func(BYTE cam_nr)
{
  WORD error;
  FEATURE_PARAMETER f_param;

  error = GEVGetFeatureParameter(cam_nr, "TransferAcknowledge",
&f_param);

  if(error)
  {
    ...
    return(1);
  }
  error = GEVSetFeatureCommand(cam_nr, "TransferAcknowledge",
f_param.CommandValue);

  if(error)
  {
    ...
    return(1);
  }
  return(0);
}
```

Return: see Error Codes

Reference: GEVGetSecureTransfer

## 2.4. Camera link functions

Following functions are available only for S2I CameraLink to GigEVision converters.

### GetClinkStatus

| | |
|---|---|
| Function: | Returns status of CameraLink serial interface |
| Syntax: | `WORD GetClinkStatus(BYTE cam_nr, CLINK_STATUS *status);` |
| Description: | This function returns the *status* of the CameraLink's serial interface.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>See Sphinxlib.h for the definition of CLINK_STATUS. |
| Return: | see Error Codes |
| Reference: | InitClinkSerial, SendClink, ReceiveClink |

### InitClinkSerial

| | |
|---|---|
| Function: | Initializes CameraLink serial interface |
| Syntax: | `WORD InitClinkSerial(BYTE cam_nr, DWORD baud);` |
| Description: | This function sets baudrate of the CameraLink serial interface.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Valid values for *baud* rate are: 9600, 19200, 38400, 57600, 115200. |
| Return: | see Error Codes |
| Reference: | SendClink, ReceiveClink, GetClinkStatus |

### ReceiveClink

| | |
|---|---|
| Function: | Receive data over CameraLink serial interface |
| Syntax: | `WORD ReceiveClink(BYTE cam_nr, WORD count_buffer, BYTE *recv_buffer, WORD *recv_count);` |
| Description: | This function reads receive buffer of CameraLink's serial interface.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *count_buffer* specifies maximum number of characters to be read. Parameter *recv_buffer* points to the received characters.<br>Parameter *recv_count* returns the number of characters read. |
| Return: | see Error Codes |
| Reference: | InitClinkSerial, SendClink, GetClinkStatus |

### SendClink

| | |
|---|---|
| Function: | Send data over CameraLink serial interface |
| Syntax: | `WORD SendClink(BYTE cam_nr, WORD count, BYTE *send_buffer);` |
| Description: | This function writes *count* characters, given by *send_buffer* to serial sender.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50]. |
| Return: | see Error Codes |
| Reference: | InitClinkSerial, ReceiveClink, GetClinkStatus |

## 2.5. Test functions

### GEVTestPacketResend

| | |
|---|---|
| Function: | Tests the packet resend function |
| Syntax: | `WORD GEVTestPacketResend(BYTE cam_nr, BYTE on_off, WORD packet_number, WORD count);` |
| Description: | This function tests the packet resend by requesting one or more selectable packets.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *on_off* switches the packet resend test on or off.<br>Parameter *packet_number* specifies the start of packets to resend.<br>Parameter count specifies the number of packets to resend. Packet Resend must be enabled with function *GEVSetPacketResend*. Also enable the DETAILED _LOG_INFO flag in the *GEVSetDetailedLog* function to see the results. |
| Return: | see Error Codes |
| Reference: | --- |

### GEVTestFindMaxPacketSize

| | |
|---|---|
| Function: | Find the maximum of packet size |
| Syntax: | `WORD GEVTestFindMaxPacketSize(BYTE cam_nr, WORD *packet_size, WORD ps_min, WORD ps_max, WORD ps_inc);` |
| Description: | This function finds the maximum of packet size.<br>Parameter *cam_nr* specifies the GigE Vision device instance [1..50].<br>Parameter *packet_size* returns the maximum found packet size.<br>Parameter *ps_min* (minimum packetsize), *ps_max* (maximum packetsize) and *ps_inc* (increment) shall be read from the xml and passed to this function. |
| Return: | see Error Codes |
| Reference: | GEVTestPacket |

## 2.6. Structures

### ENUMERATE_ADAPTER

| | |
|---|---|
| Count: | Number of network interface adapter found. |
| Param: | Struct of found network interface adapter parameter (ADAPTER_PARAM) |

### ADAPTER_PARAM

| | |
|---|---|
| AdapterIP: | IP address of network interface adapter. |
| AdapterMask: | Subnet mask of network interface adapter. |
| AdapterName: | Name of network interface adapter. |

### DISCOVERY

| | |
|---|---|
| Count: | Number of devices found. |
| Param: | Struct of found device parameter (DEVICE_PARAM) |

### DEVICE_PARAM

| | |
|---|---|
| IP: | ip address of the device |
| manuf: | manufacturer name of the device |
| mode: | model name of the device |
| version: | version of the device |
| AdapterIP: | ip address of the network adapter |
| AdapterMask: | mask of the network adapter |
| Mac: | mac address of the device |
| subnet: | subnet mask of the device |
| gateway: | gateway of the device adapter_name:network adapter name |
| serial: | serial number of the device |
| userdef_name: | user defined name |
| status: | connection status of the device |

```
#define DISCOVERY_STATUS_OK                 0
#define DISCOVERY_STATUS_ALLREADY_OPEN      1
#define DISCOVERY_STATUS_NOT_SAME_SUBNET    2
#define DISCOVERY_STATUS_CONTROL_OPEN       3
```

### CONNECTION

| | |
|---|---|
| IP_CANCam: | ip address of the device |
| PortData: | socket port of the stream channel (value of 0 set port automatic) |
| PortCtrl: | socket port of the control channel (value of 0 set port automatic) |
| AdapterIP: | ip address of the network adapter |
| AdapterMask: | subnet mask of the network adapter |
| adapter_name: | name of the network adapter |
| PortMessage: | socket port of the message channel (value of 0 set port automatic) |

## CHANNEL_PARAMETER

cc_heartbeat_timeout: control channel heartbeat timeout counter in milliseconds

cc_timeout: control channel timeout in milliseconds

cc_retry: control channel retry count

sc_timeout: stream channel timeout in milliseconds

sc_packet_resend: stream channel packet resend count

sc_image_wait_timeout: stream channel wait of an image timeout in milliseconds

## FeatureList

Next: pointer to next feature

Index: index of feature (1,2,3,…)

Name: name of the feature

Type: type of the feature

```
#define TYPE_CATEGORY     0
#define TYPE_FEATURE      1
#define TYPE_INTEGER      2
#define TYPE_FLOAT        3
#define TYPE_STRING       4
#define TYPE_ENUMERATION  5
#define TYPE_COMMAND      6
#define TYPE_BOOLEAN      7
#define TYPE_REGISTER     8
#define TYPE_PORT         9
```

Level: level of the feature

- feature 1 -> level 0
  - feature 1.1 -> level 1
  - feature 1.2 -> level 1
    - feature 1.2.1 -> level 2
    - feature 1.2.2 -> level 2
- feature 2 -> level 0
  - feature 2.1 -> level 1
  - feature 2.2 -> level 1
    - feature 2.2.1 -> level 2
    - feature 2.2.2 -> level 2

## FEATURE_PARAMETER

Type: type of the feature

```
#define TYPE_CATEGORY     0
#define TYPE_FEATURE      1
#define TYPE_INTEGER      2
#define TYPE_FLOAT        3
#define TYPE_STRING       4
#define TYPE_ENUMERATION  5
```

```
#define TYPE_COMMAND      6
#define TYPE_BOOLEAN      7
#define TYPE_REGISTER     8
#define TYPE_PORT         9
```

Min: minimum value of a integer feature

Max: maximum value of a integer feature

OnValue: OnValue of a Boolean node (see GenICam Standard)

OffValue: OffValue of a Boolean node (see GenICam Standard)

AccessMode: access mode of the feature

```
#define ACCESS_MODE_RO    0x524F  // read only
#define ACCESS_MODE_RW    0x5257  // read/write
#define ACCESS_MODE_WO    0x574F  // write only
```

Representation: The Representation element gives a hint about how to display the integer.

```
// Slider with linear behaviour
#define REPRESENTATION_LINEAR       0
// Slider with logarithmic behaviour
#define REPRESENTATION_LOGARITHMIC  1
// Checkbox
#define REPRESENTATION_BOOLEAN      2
// Decimal number in an edit control
#define REPRESENTATION_PURE_NUMBER  3
// Hex number in an edit control
#define REPRESENTATION_HEX_NUMBER   4
// Undefinded Representation
#define REPRESENTATION_UNDEFINDED   5
```

Inc: increment of an integer feature

CommandValue: command value of a command node

Length: length of the feature in bytes

EnumerationCount: count of enumeration node

Visibility: The Visibility element defines the user level that should get access to the feature

```
#define VISIBILITY_INVISIBLE 0
#define VISIBILITY_BEGINNER  1
#define VISIBILITY_EXPERT    2
#define VISIBILITY_GURU      3
```

FloatMin: minimum value of a float feature

FloatMax: maximum value of a float feature

IsImplemented:

IsAvailable:

IsLocked: The IsImplemented, IsAvaliable, and IsLocked elements contains the names of nodes implementing an IInteger interface.

Sign: The Sign element can have the value *Singed* or *Unsigned*.

```
#define SIGN_UNSIGNED   0
```

```
#define SIGN_SIGNED      1
```

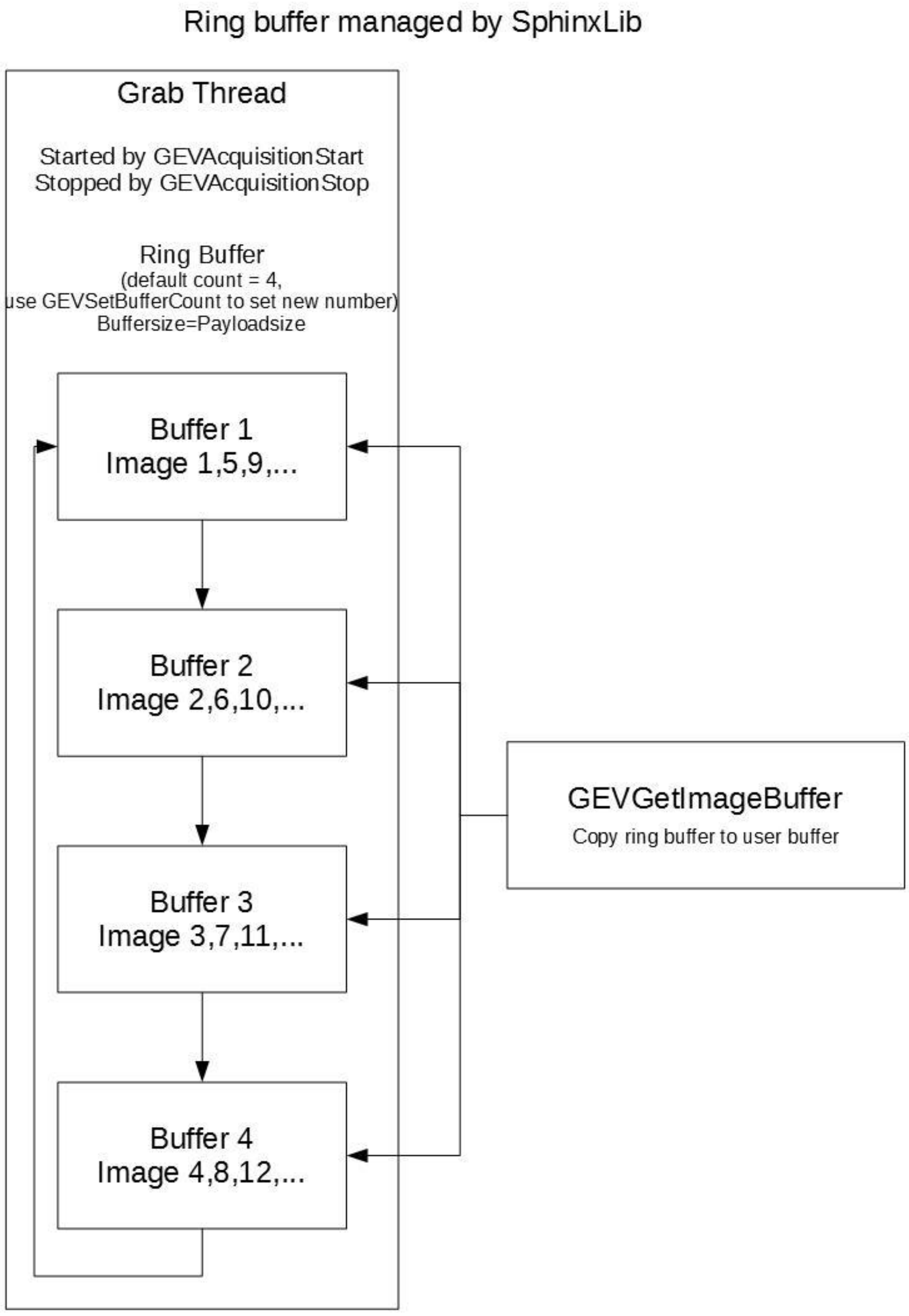
| | |
|---|---|
| Address: | Address of the feature |
| DisplayNotation: | Notation of the display<br>`#define DISPLAY_NOTATION_AUTOMATIC   0`<br>`#define DISPLAY_NOTATION_FIXED       1`<br>`#define DISPLAY_NOTATION_SCIENTIFIC  2` |
| DisplayPrecision: | Precision of the display |
| InvalidatorCount: | Number of invalidators of the feature |
| PollingTime: | Polling time of the feature |

## IMAGE_HEADER

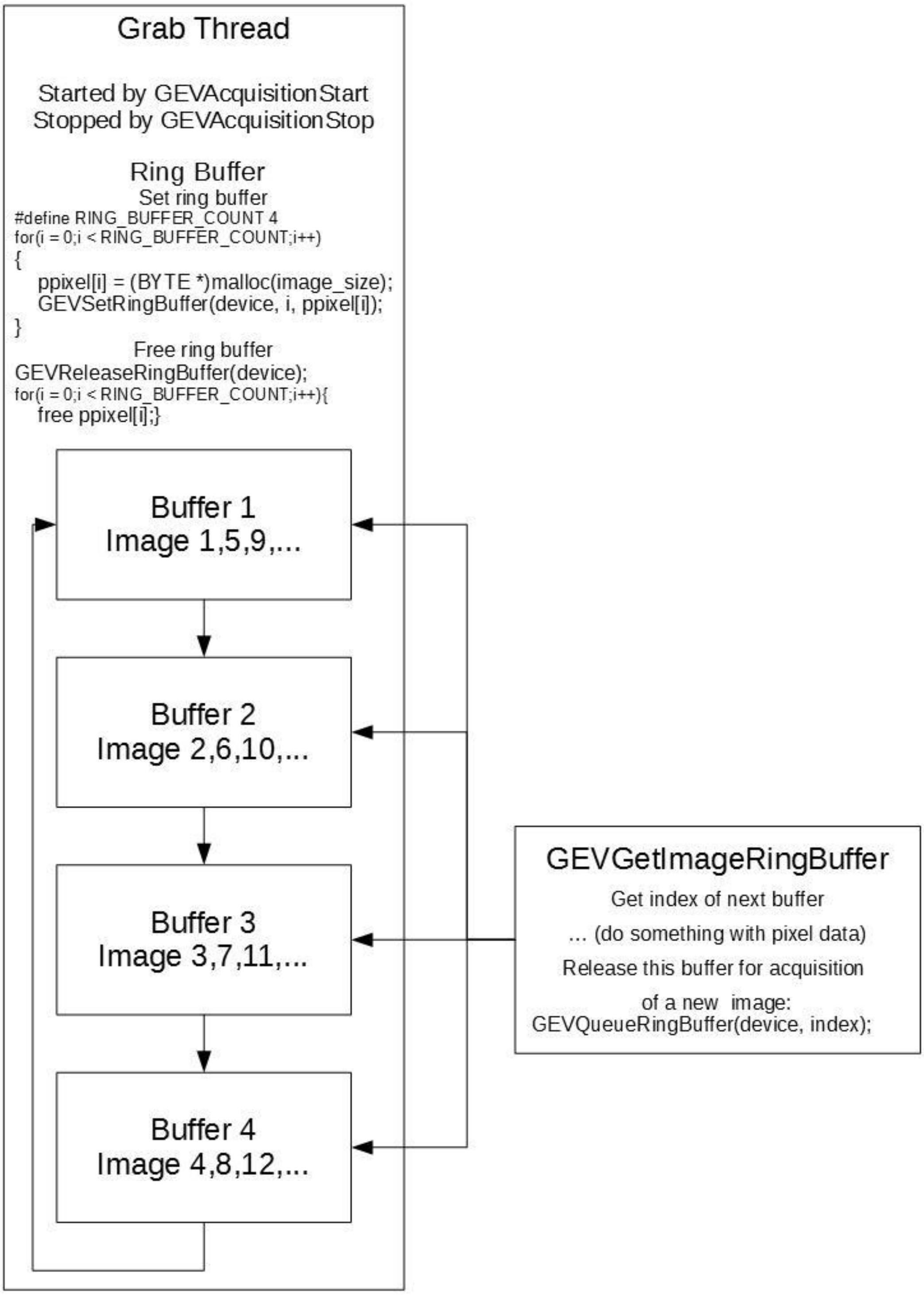
| | |
|---|---|
| FrameCounter: | Current frame counter |
| TimeStamp: | Timestamp of the image |
| PixelType | Pixel type of the image |
| SizeX: | Width in pixels of the image |
| SizeY: | Height in lines of the image |
| OffsetX: | Offset in pixels from the image origin |
| OffsetY: | Offset in lines from the image origin |
| PaddingX: | Horizontal padding expressed in bytes |
| PaddingY: | Vertical padding expressed in bytes |
| MissingPacket: | Number of missing packets |
| PayloadType | Payload Type |
| ChunkDataPayloadLength | Length of all the chunks data payload in bytes. |
| ChunkLayoutId | Chunk layout ID |

# 3. Appendix

## 3.1. Buffer Structure



Ring buffer managed by user



```
#define RING_BUFFER_COUNT 4
for(i = 0;i < RING_BUFFER_COUNT;i++)
{
    ppixel[i] = (BYTE *)malloc(image_size);
    GEVSetRingBuffer(device, i, ppixel[i]);
}
```

```
GEVReleaseRingBuffer(device);
for(i = 0;i < RING_BUFFER_COUNT;i++){
    free ppixel[i];}
```

```
GEVQueueRingBuffer(device, index);
```

## 3.2. Error Codes

| Status Value | Value | Description |
|---|---|---|
| GEV_STATUS_SUCCESS | 0x0000 | Operation executed successfully |
| GigE Vision Status Codes | 0x8001 | |
| | … | |
| | 0x8FFF | See GigE Vision Specification |
| | | |
| GEV_STATUS_CAMERA_NOT_INIT | 0xC001 | Device was not opened with the function GEVInit |
| GEV_STATUS_CAMERA_ALWAYS_INIT | 0xC002 | Device was not closed with the function GEVClose |
| GEV_STATUS_CANNOT_CREATE_SOCKET | 0xC003 | Can't create Socket port |
| GEV_STATUS_SEND_ERROR | 0xC004 | Error sending a GigE Vision command |
| GEV_STATUS_RECEIVE_ERROR | 0xC005 | Error receiving a GigE Vision acknowledge |
| GEV_STATUS_CAMERA_NOT_FOUND | 0xC006 | Device not found (no longer used) |
| GEV_STATUS_CANNOT_ALLOC_MEMORY | 0xC007 | Error to allocate memory |
| GEV_STATUS_TIMEOUT | 0xC008 | Timeout to get data from the socket port |
| GEV_STATUS_SOCKET_ERROR | 0xC009 | Socket port error |
| GEV_STATUS_INVALID_ACK | 0xC00A | Command acknowledge invalid |
| GEV_STATUS_CANNOT_START_THREAD | 0xC00B | Thread could not be started |
| GEV_STATUS_CANNOT_SET_SOCKET_OPT | 0xC00C | Failed to set a socket option |
| GEV_STATUS_CANNOT_OPEN_DRIVER | 0xC00D | Driver could not be opened. Check if driver is installed or you don't have administrator privileges |
| GEV_STATUS_HEARTBEAT_READ_ERROR | 0xC00E | Heartbeat read error (no longer used) |
| GEV_STATUS_EVALUATION_EXPIRED | 0xC00F | Evaluation expired |
| GEV_STATUS_GRAB_ERROR | 0xC010 | Image could not be completely transferred. See IMAGE_HEADER struct MissingPacket parameter |
| GEV_STATUS_DRIVER_READ_ERROR | 0xC011 | Error communicate with the driver |
| GEV_STATUS_XML_READ_ERROR | 0xC012 | Error read xml file |
| GEV_STATUS_XML_OPEN_ERROR | 0xC013 | Error open xml file |
| GEV_STATUS_XML_FEATURE_ERROR | 0xC014 | Feature error (no longer used) |
| GEV_STATUS_XML_COMMAND_ERROR | 0xC015 | Feature command error (no longer used) |
| GEV_STATUS_GAIN_NOT_SUPPORTED | 0xC016 | Gain feature not supported (no longer used) |
| GEV_STATUS_EXPOSURE_NOT_SUPPORTED | 0xC017 | Exposure feature not supported (no longer used) |
| GEV_STATUS_CANNOT_GET_ADAPTER_INFO | 0xC018 | Can't get information from network adapter |
| GEV_STATUS_ERROR_INVALID_HANDLE | 0xC019 | Invalide handle |
| GEV_STATUS_CLINK_SET_BAUD | 0xC01A | Error to set Clink baud rate |
| GEV_STATUS_CLINK_SEND_BUFFER_FULL | 0xC01B | Clink send buffer is full |
| GEV_STATUS_CLINK_RECEIVE_BUFFER_NO_DATA | 0xC01C | No data in receive buffer available |
| GEV_STATUS_FEATURE_NOT_AVAILABLE | 0xC01D | Feature not available |
| GEV_STATUS_MATH_PARSER_ERROR | 0xC01E | Math parser error |
| GEV_STATUS_FEATURE_ITEM_NOT_AVAILABLE | 0xC01F | Feature item not available (enum entry). |
| GEV_STATUS_NOT_SUPPORTED | 0xC020 | Not supported. |
| GEV_STATUS_GET_URL_ERROR | 0xC021 | Error reading the url string of the device |
| GEV_STATUS_READ_XML_MEM_ERROR | 0xC022 | Error read xml file of the device |
| GEV_STATUS_XML_SIZE_ERROR | 0xC023 | Size of xml file is wrong. |
| GEV_STATUS_XML_ZIP_ERROR | 0xC024 | Error unzip xml file |
| GEV_STATUS_XML_ROOT_ERROR | 0xC025 | Unable to get xml root element of the xml file |
| GEV_STATUS_XML_FILE_ERROR | 0xC026 | Xml file is corrupt. |
| GEV_STATUS_DIFFERENT_IMAGE_HEADER | 0xC027 | Stream image header is wrong. |
| GEV_STATUS_XML_SCHEMA_ERROR | 0xC028 | Error while parsing the XML with the schema file |
| GEV_STATUS_XML_STYLESHEET_ERROR | 0xC029 | Error while parsing the XML with the stylesheet file |
| GEV_STATUS_FEATURE_LIST_ERROR | 0xC02A | Failed to create the feature list |
| GEV_STATUS_ALREADY_OPEN | 0xC02B | Device is already open. |
| GEV_STATUS_TEST_PACKET_DATA_ERROR | 0xC02C | Data from test packet is corrupt |
| GEV_STATUS_FEATURE_NOT_FLOAT | 0xC02D | The feature is not a float feature |
| GEV_STATUS_FEATURE_NOT_INTEGER | 0xC02E | The feature is not a integer feature |

| | | |
|---|---|---|
| GEV_STATUS_XML_DLL_NOT_FOUND | 0xC02F | Xml library libxml2.dll not found. Copy libxml2.dll to same directory as SphinxLib or set dll path with SetDllDirectory function. |
| GEV_STATUS_XML_NOT_INIT | 0xC030 | XML was not initialized with GEVInitXml |
| GEV_STATUS_NOT_SAME_SUBNET | 0xC031 | device is not in the same subnet as network card |

## 3.3. Supported Payload Types

| Payload Type | Value | Description |
|---|---|---|
| Image | 0x0001 | Uncompressed image data |
|  | 0x4001 | Support for Image extended chunk data is allowed. In this case, Image must be the first chunk |
|  |  |  |
| Raw data | 0x0002 | Raw binary data |
|  |  |  |
| Chunk data | 0x0004 | Tagged blocks of data, as per the GenICam specification |
|  |  |  |
| JPEG | 0x0006 | JPEG compressed image, as per ITU-T Rec. T.81 |
|  | 0x4006 | Support for JPEG extended chunk data is allowed. In this case, JPEG compressed image must be the first chunk. |
|  |  |  |
| JPEG 2000 | 0x0007 | JPEG 2000 codestream, as per ITU-T Rec. T.800 |
|  | 0x4007 | Support for JPEG 2000 extended chunk data is allowed. In this case, JPEG 2000 codestream must be the first chunk. |
|  |  |  |
| Multi-part | 0x000A | Multiple elements of an image or other data, called parts, coming from the same acquisition time. |
|  |  |  |
| GenDC | 0x000B | A GenICam GenDC container. GenDC with extended chunk data is not allowed as chunk should be part the GenDC container. |

## 3.4. Compiling the library

### *3.4.1. Windows*

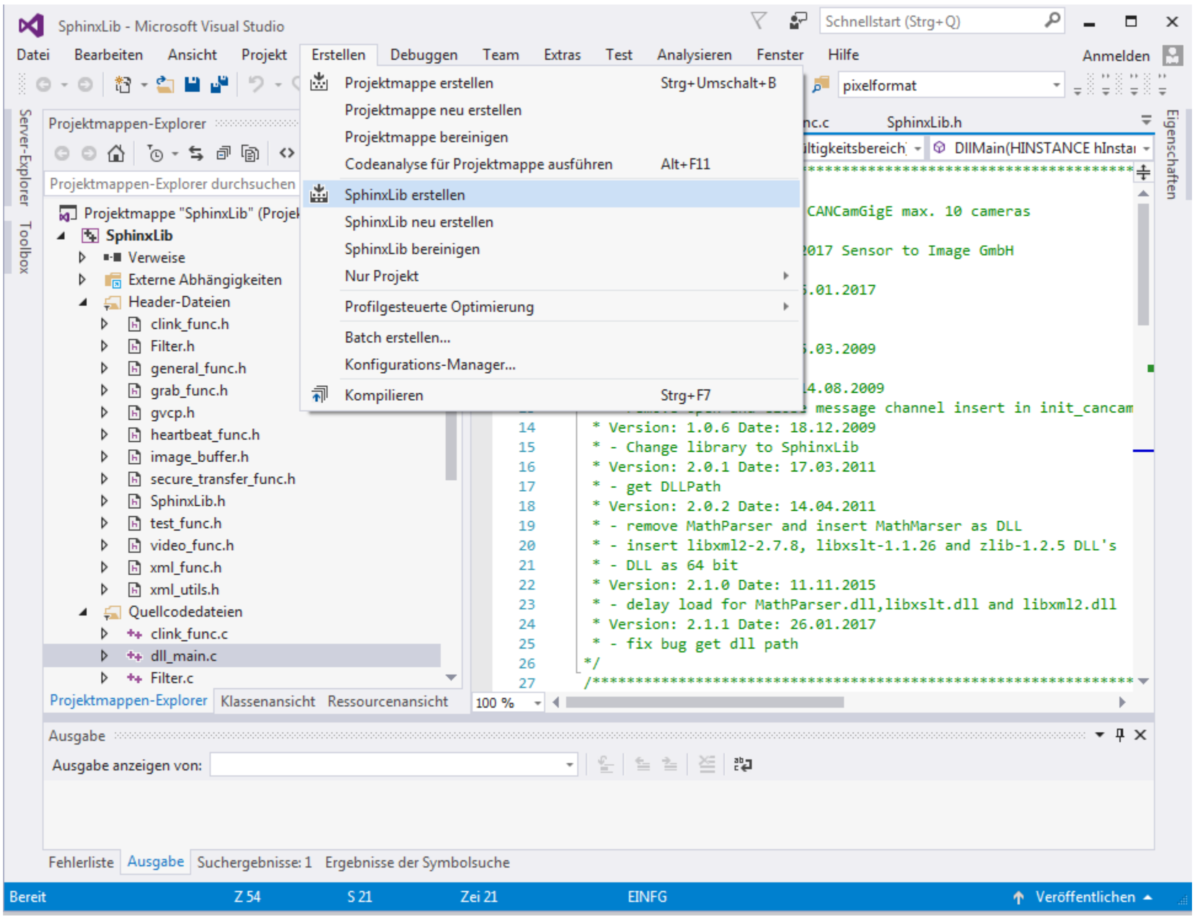
#### *3.4.1.1. Library*

The library is provided as a complete project for Visual Studio 2015. Other compilers may be able to import that project or a new project has to be created.

There is available a free version of Visual Studio, called Visual Sudio Express.
(see https://www.visualstudio.com/de/vs/visual-studio-express/

or

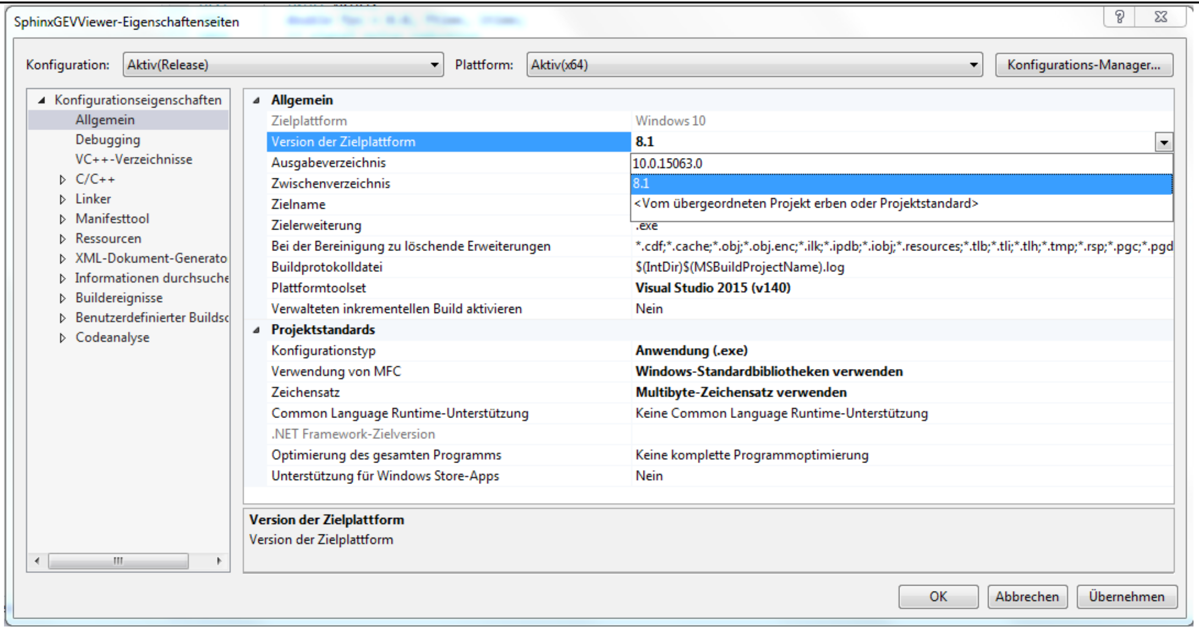
https://developer.microsoft.com/en-us/windows/hardware/windows-driver-kit)



```
****************************************
CANCamGigE max. 10 cameras
017 Sensor to Image GmbH
.01.2017
.03.2009
4.08.2009
 message channel insert in init_cancam
 * Version: 1.0.6 Date: 18.12.2009
 * - Change library to SphinxLib
 * Version: 2.0.1 Date: 17.03.2011
 * - get DLLPath
 * Version: 2.0.2 Date: 14.04.2011
 * - remove MathParser and insert MathMarser as DLL
 * - insert libxml2-2.7.8, libxslt-1.1.26 and zlib-1.2.5 DLL's
 * - DLL as 64 bit
 * Version: 2.1.0 Date: 11.11.2015
 * - delay load for MathParser.dll,libxslt.dll and libxml2.dll
 * Version: 2.1.1 Date: 26.01.2017
 * - fix bug get dll path
 */
/*******************************************************************
```

*Build Library in Visual Studio 2015*

**Note:**
If you have the windows sdk 8.1 installed you have to change in the settings to 8.1.



Konfiguration: Aktiv(Release)
Plattform: Aktiv(x64)

10.0.15063.0
8.1
<Vom übergeordneten Projekt erben oder Projektstandard>

| | |
|---|---|
| **Allgemein** | |
| Zielplattform | Windows 10 |
| Version der Zielplattform | **8.1** |
| Ausgabeverzeichnis | |
| Zwischenverzeichnis | |
| Zielname | |
| Zielerweiterung | .exe |
| Bei der Bereinigung zu löschende Erweiterungen | *.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.tlb;*.tli;*.tlh;*.tmp;*.rsp;*.pgc;*.pgd |
| Buildprotokolldatei | $(IntDir)$(MSBuildProjectName).log |
| Plattformtoolset | **Visual Studio 2015 (v140)** |
| Verwalteten inkrementellen Build aktivieren | Nein |
| **Projektstandards** | |
| Konfigurationstyp | **Anwendung (.exe)** |
| Verwendung von MFC | **Windows-Standardbibliotheken verwenden** |
| Zeichensatz | **Multibyte-Zeichensatz verwenden** |
| Common Language Runtime-Unterstützung | Keine Common Language Runtime-Unterstützung |
| .NET Framework-Zielversion | |
| Optimierung des gesamten Programms | Keine komplette Programmoptimierung |
| Unterstützung für Windows Store-Apps | Nein |

**Version der Zielplattform**
Version der Zielplattform

### 3.4.1.2. Filterdriver

Compilation of filter driver needs Microsoft Windows Driver Kit (WDK 10)
(see https://developer.microsoft.com/en-us/windows/hardware/windows-driver-kit)

Use the visual studio project in the FilterDriverSourceCode6 directory to compile filter driver.

## 3.4.2. Linux

### 3.4.2.1. Directories

GEV/ConsoleDemo ⇨ console demo application
GEV/ConsoleDemo-GenICam3 ⇨ console GenICam demo application
GEV/FilterDriverLinux ⇨ s2i GigE-Vision linux filter driver (if available)
GenTL/GenTLConsumer ⇨ GenTL consumer demo
GEV/GenTLProducer ⇨ GenTL producer (if available)
GEV/GenTLProducerSource ⇨ GenTL producer source code (if available)
Xml/MathParser ⇨ math parser library
Xml/MathParserSource ⇨ math parser library source code
GEV/SphinxGEVViewerQt ⇨ Qt sphinxGEVviewer application
GEV/SphinxLib ⇨ sphinxlib library
GEV/SphinxLibSource ⇨ sphinxlib library source code
Xml/SphinxXmlLib ⇨ sphinxlib library (if available)
Xml/SphinxXmlLibSource ⇨ sphinxlib library source code (if available)
GEV/UpdateGigEConsole ⇨ update GigE console (if available)
GEV/UpdateGigEConsoleSource ⇨ update GigE console source code (if available)

### 3.4.2.2. Install required libraries (as root or using sudo)

- SDL2
- Qt5

- libxml2
- libxslt
- C++ Compiler

Install on Ubuntu

```
apt-get install libsdl2-2.0-0 libsdl2-dev qt5-default qtbase5-dev libqt5x11extras5 libqt5x11extras5-dev libxml2 libxml2-dev libxslt1.1 libxslt1-dev g++
```

Notes: We changed the Qt sphinxGEVviewer application to Qt5 and SDL2.
If you want to use Qt4 and SDL 1.2, you must replace SDL2 with SDL in the sphinxGEVviewer.pro file.

#### 3.4.2.3. Build the MathParser linux library

Only possible if the MathParser source code (Xml/MathParserSource directory) are available.

Command: *cd Xml/MathParserSource/Linux64*
or
*cd Xml/MathParserSource/Linux32*
*make*
or for debug version
*make DEBUG=1*

The MathParser library is copied to the ../../MathParser/Linux64 or ../../MathParser/Linux32 directory

Notes: It is necessary to have the gnu c++ compiler.

#### 3.4.2.4. Build the Sphinx linux library

Only possible if the SphinxLib source code (/GEV/SphinxlibSource directory) are available.

Command: *cd GEV/SphinxLibSource/Linux64*
or
*cd GEV/SphinxLibSource/Linux32*
*make*
or for debug version
*make DEBUG=1*
or for library without xml parser support
*make NO_XML=1*

The SphinxLib library is copied to the ../../SphinxLib/Linux64 or ../../SphinxLib/Linux32 directory.

Notes: It is necessary to have libxml2, libxslt and zlib development libraries. Also the gnu c++ compiler.

#### 3.4.2.5. Build the console demo application

Command: *GEV/ConsoleDemo/Linux64*

```
GEV/ConsoleDemo/Linux64
or
GEV/ConsoleDemo/Linux32
make
cd Release
./ConsoleDemo
```

### 3.4.2.6. Build the console GenICam demo application

Command:

```
cd GEV/ConsoleDemo-GenICam3/Linux64
or
cd GEV/ConsoleDemo-GenICam3/Linux32
make
cd Release
source ./SetGenICamRoot.sh
./ConsoleDemo
```

### 3.4.2.7. Build the QT sphinxGEVviewer application

Command:

```
cd GEV/SphinxGEVViewerQt/Linux64
or
cd GEV/SphinxGEVViewerQt/Linux32
```

To create a makefile:

Command:

```
/usr/bin/qmake -unix -o Makefile sphinxGEVviewer.pro
```

Command:

```
make
make install
cd Release
./sphinxGEVviewer
or when use filter driver
sudo ./sphinxGEVviewer
```

Notes: It is necessary to have SDL2 and QT5 development libraries. Also the gnu c++ compiler.
We changed the Qt sphinxGEVviewer application to Qt5 and SDL2.
If you want to use Qt4 and SDL 1.2, you must replace SDL2 with SDL in the sphinxGEVviewer.pro file.

### 3.4.2.8. Build the s2igevfilter filter driver

a) Build the module

Command: *make*

Notes: It is necessary to have the /usr/src/linux symbolic link set to point to sources of your current kernel.

b) Install the module

Command: *make install* (must be executed as root)
Notes: It installs the module into current kernel's module directory (/lib/modules/<kernel_version>/extra) and runs depmod to regenerate dependencies.
Create device nodes s2igevfilter0.. s2igevfilter9 in /lib/udev/devices.
Insert alias char-major-120 s2igevfilter to modprobe.conf file.

These two steps load the module and create the device node during boot process.
After this steps it is possible to use s2igevfilter module.

c) Uninstall the module
Command: *make uninstall* (must be executed as root)
Notes: This uninstalls the module from current kernel's module directory (in my system it is /lib/modules/<kernel_version>/extra) and runs depmod to regenerate dependencies.
Delete device nodes s2igevfilter0.. s2igevfilter9 in /lib/udev/devices.

#### 3.4.2.9. Build the GenTL producer

Only possible if the GenTL producer source code (GEV/GenTLProducerSource directory) are available.

Command:

```
cd GEV/ GenTLProducerSource/Linux64
or
cd GEV/ GenTLProducerSource/Linux32
chmod +x ../generate_header_files.sh
mkdir Release
cd Release
cmake ../
make
make install
```

The producer is copied to the ../../GenTLProducer/Linux64 or ../../GenTLProducer/Linux32 directory.

Notes: It is necessary to have cmake installed.

#### 3.4.2.10. Build the SphinxXmlLib library

Needed for GenTL consumer demo application.

Only possible if the SphinxXmlLib source code (Xml/SphinxXmlLibSourceCode directory) are available.

Command:

```
cd Xml/SphinxXmlLibSource/Linux64
or
cd Xml/SphinxXmlLibSource/Linux32
make
or for debug version
make DEBUG=1
```

The SphinxXmlLib is copied to the ../../SphinxXmlLib/Linux64 or ../../SphinxXmlLib/Linux32 directory

Notes: It is necessary to have libxml2 and libxslt development libraries. Also the gnu c compiler.

#### 3.4.2.11. Build the GenTL consumer demo application

Only possible if the GenTL consumer demo source code (GenTL/GenTLConsumer directory) are available.
Command:

```
cd GenTL/GenTLConsumer/Linux64
or
cd GenTL/GenTLConsumer/Linux32
make
cd Release
```

```
cp ../../../GEV/GenTLProducer/Linux64/libGEVTLS2I.cti .
or
cp ../../../GEV/GenTLProducer/Linux32/libGEVTLS2I.cti .
./GenTLDemo
```

### 3.4.2.12. Build the update GigE console application

Only possible if the update GigE console source code (GEV/UpdateGigEConsoleSource directory) are available.

Command:
```
cd GEV/ UpdateGigEConsole/Linux64
or
cd GEV/ UpdateGigEConsole/Linux32
make
cd Release
./UpdateGigEConsole
```

## 3.4.3. MacOS

### 3.4.3.1. Directories

GEV/ConsoleDemo ⇨ console demo application
GEV/ConsoleDemo-GenICam3 ⇨ console GenICam demo application
GEV/FilterDriverMac ⇨ s2i GigE-Vision mac filter driver (if available)
GenTL/GenTLConsumer ⇨ GenTL consumer demo
GEV/GenTLProducer ⇨ GenTL producer (if available)
GEV/GenTLProducerSource ⇨ GenTL producer source code (if available)
Xml/MathParser ⇨ math parser library
Xml/MathParserSource ⇨ math parser library source code
GEV/SphinxGEVViewerQt ⇨ Qt sphinxGEVviewer application
GEV/SphinxLib ⇨ sphinxlib library
GEV/SphinxLibSource ⇨ sphinxlib library source code
Xml/SphinxXmlLib ⇨ sphinxlib library (if available)
Xml/SphinxXmlLibSource ⇨ sphinxlib library source code (if available)
GEV/UpdateGigEConsole ⇨ update GigE console (if available)
GEV/UpdateGigEConsoleSource ⇨ update GigE console source code (if available)

### 3.4.3.2. Install required libraries (as root or using sudo)

Install Xcode from the app store

Download SDL2 from https://www.libsdl.org/download-2.0.php
To Install:
Copy the SDL2.framework to /Library/Frameworks
You may alternatively install it in <Your home directory>/Library/Frameworks if your access privileges are not high enough.

Download Qt 5.10 from https://www.qt.io/download-qt-installer?hsCtaTracking=9f6a2170-a938-42df-a8e2-a9f0b1d6cdce%7C6cb0de4f-9bb5-4778-ab02-bfb62735f3e5
and install it

### 3.4.3.3. Build the MathParser linux library

Only possible if the MathParser source code (Xml/MathParserSource directory) are available.

Command:
```
cd Xml/MathParserSource/Mac64
make
```
or for debug version
```
make DEBUG=1
```

The MathParser library is copied to the ../../MathParser/Mac64 directory.

Notes: It is necessary to have the gnu c++ compiler.

#### 3.4.3.4. Build the Sphinx linux library

Only possible if the SphinxLib source code (/GEV/SphinxlibSource directory) are available.

Command:
```
cd GEV/SphinxLibSource/Mac64
make
```
or for debug version
```
make DEBUG=1
```
or for library without xml parser support
```
make NO_XML=1
```

The SphinxLib library is copied to the ../../SphinxLib/Mac64 directory.

Notes: It is necessary to have libxml2, libxslt and zlib development libraries. Also the gnu c++ compiler.

#### 3.4.3.5. Build the console demo application

Command:
```
GEV/ConsoleDemo/Mac64
make
cd Release
./ConsoleDemo
```

#### 3.4.3.6. Build the console GenICam demo application

Command:
```
cd GEV/ConsoleDemo-GenICam3/Mac64
make
cd Release
source ./SetGenICamRoot.sh
./ConsoleDemo
```

#### 3.4.3.7. Build the QT sphinxGEVviewer application

Command:
```
cd GEV/SphinxGEVViewerQt/Mac64
```

To create a makefile:

```
~/Qt/5.10.1/clang_64/bin/qmake -o Makefile sphinxGEVviewer.pro
```

Command:
```
make
make install
cd Release/sphinxGEVviewer/Contents/MacOS
./sphinxGEVviewer
```
or when use filter driver
```
sudo ./sphinxGEVviewer
```

Notes: It is necessary to have SDL and QT development libraries. Also the gnu c++ compiler.

#### 3.4.3.8. Build the s2igevfilter filter driver

a) Build the driver
Open the GEV/FilterDriverMac/s2igevfilter.xcodeproj with Xcode and build the driver.
or

```
cd GEV/FilterDriverMac
xcodebuild -project s2igevfilter.xcodeproj -alltargets -configuration Release
```

b) Install the driver

Enable SIP and allow installation of unsigned kernel extensions

1. Reboot your Mac into Recovery Mode by restarting your computer and holding down Command+R until the Apple logo appears on your screen.
2. Click Utilities > Terminal.
3. In the Terminal window, type in "csrutil enable --without kext" and press Enter.
4. Restart your Mac.

go to driver

```
cd GEV/FilterDriverMac/Build/Release
```

do this as root

```
sudo -s
```

copy driver to /Library/Extensions

```
cp -r s2igevfilter.kext /Library/Extensions
```

change rigths

```
chmod -R 755 /Library/Extensions/s2igevfilter.kext
chown -R root:wheel /Library/Extensions/s2igevfilter.kext
```

load driver

```
kextload -v /Library/Extensions/s2igevfilter.kext
```

Note: After the PC was restarted, the driver only has to be loaded.
If you want to use the sphinxGEVviewer application with the filter driver, you must start the sphinxGEVviewer application as root

c) Uninstall the driver

do this as root

```
sudo -s
```

unload driver

```
kextunload /Library/Extensions/s2igevfilter.kext
```

delete driver

```
rm -rf /Library/Extensions/s2igevfilter.kext
```

### 3.4.3.9. Build the GenTL producer

Only possible if the GenTL producer source code (GEV/GenTLProducerSource directory) are available.

Command:
```
cd GEV/ GenTLProducerSource/Mac64
chmod +x ../generate_header_files.sh
mkdir Release
cd Release
cmake ../
make
make install
```

The producer is copied to the ../../GenTLProducer/Mac64

Notes: It is necessary to have cmake installed. (brew install cmake)

### 3.4.3.10. Build the SphinxXmlLib library

Needed for GenTL consumer demo application.

Only possible if the SphinxXmlLib source code (Xml/SphinxXmlLibSourceCode directory) are available.

Command:
```
cd Xml/SphinxXmlLibSource/Mac64
make
or for debug version
make DEBUG=1
```

The SphinxXmlLib is copied to the ../../SphinxXmlLib/Mac64

Notes: It is necessary to have libxml2 and libxslt development libraries. Also the gnu c compiler.

### 3.4.3.11. Build the GenTL consumer demo application

Only possible if the GenTL consumer demo source code (GenTL/GenTLConsumer directory) are available.

Command:
```
cd GenTL/GenTLConsumer/Mac64
make
cd Release
cp ../../../GEV/GenTLProducer/Mac64/libGEVTLS2I.cti .
./GenTLDemo
```

### 3.4.3.12. Build the update GigE console application

Only possible if the update GigE console source code (GEV/UpdateGigEConsoleSource directory) are available.

Command:
```
cd GEV/ UpdateGigEConsole/Mac64
make
cd Release
./UpdateGigEConsole
```

# 4. Revision Changes

## 4.1. V2.6.0 to V2.7.0

Refactoring SphinxLib and filter drivers

## 4.2. V2.5.0 to V2.6.0

Added PortMessage parameter to CONNECTION struct.

### GEVCheckDeviceStatus

**New**

```
WORD GEVCheckDeviceStatus(DWORD ip_adapter,DWORD mask_adapter,DWORD
ip_device,BYTE *device_status, DWORD ack_timeout, WORD port);
```

### GEVDiscovery

**Old**

```
WORD GEVDiscovery(DISCOVERY *dis, DISCOVERY_CALLBACK_FUNC c_func, DWORD
d_timeout, BOOL ignore_subnet);
```

**New**

```
WORD GEVDiscovery(DISCOVERY *dis, DISCOVERY_CALLBACK_FUNC c_func, DWORD
d_timeout, BOOL ignore_subnet, WORD port);
```

### GEVDiscoveryAdapter

**Old**

```
WORD GEVDiscoveryAdapter(DISCOVERY* dis, ADAPTER_PARAM* adapter,
DISCOVERY_CALLBACK_FUNC c_func, DWORD d_timeout, BOOL ignore_subnet);
```

**New**

```
WORD GEVDiscoveryAdapter(DISCOVERY* dis, ADAPTER_PARAM* adapter,
DISCOVERY_CALLBACK_FUNC c_func, DWORD d_timeout, BOOL ignore_subnet,
WORD port);
```

### GEVForceIp

**Old**

```
WORD GEVForceIp(DWORD new_ip, DWORD subnet, DWORD gateway, BYTE
*mac);
```

**New**

```
WORD GEVForceIp(DWORD new_ip, DWORD subnet, DWORD gateway, BYTE
*mac, DWORD adapter_ip);
```

## 4.3. V2.4.0 to V2.5.0

Network byte order for all functions who use ip addresses.

### GetClinkStatus

**Old**

```
WORD CANCamGetClinkStatus(BYTE cam_nr, CLINK_STATUS *status);
```

**New**

```
WORD GetClinkStatus(BYTE cam_nr, CLINK_STATUS *status);
```

### InitClinkSerial

#### Old

```
WORD CANCamInitClinkSerial(BYTE cam_nr, DWORD baud);
```

#### New

```
WORD InitClinkSerial(BYTE cam_nr, DWORD baud);
```

### ReceiveClink

#### Old

```
WORD CANCamReceiveClink(BYTE cam_nr, WORD count_buffer, BYTE
*recv_buffer, WORD *recv_count);
```

#### New

```
WORD ReceiveClink(BYTE cam_nr, WORD count_buffer, BYTE
*recv_buffer, WORD *recv_count);
```

### SendClink

#### Old

```
WORD CANCamSendClink(BYTE cam_nr, WORD count, BYTE *send_buffer);
```

#### New

```
WORD SendClink(BYTE cam_nr, WORD count, BYTE *send_buffer);
```

### DISCOVERY_CALLBACK_FUNC

#### Old

```
typedef BYTE ( WINAPI *DISCOVERY_CALLBACK_FUNC )( int s_cnt);
```

#### New

```
typedef BYTE ( WINAPI *DISCOVERY_CALLBACK_FUNC )( int s_cnt,
DEVICE_PARAM *dparam);
```

### ERROR_CALLBACK_FUNC

#### Old

```
typedef BYTE ( WINAPI *ERROR_CALLBACK_FUNC )(BYTE cam_nr, char
*error_str);
```

#### New

```
typedef BYTE ( WINAPI *ERROR_CALLBACK_FUNC )(BYTE cam_nr, char
*error_str, BYTE detailed_log);
```

## 4.4. V2.3.0 to V2.4.0

### GEVEnumerateAdapter

#### New

```
WORD WINAPI GEVEnumerateAdapter(ENUMERATE_ADAPTER* adapter);
```

### GEVDiscoveryAdapter

#### New

```
WORD WINAPI GEVDiscoveryAdapter(DISCOVERY* dis, ADAPTER_PARAM* adapter,
DISCOVERY_CALLBACK_FUNC c_func, DWORD d_timeout, BOOL ignore_subnet);
```

## 4.5. V2.2.0 to V2.3.0

Added and use the pixel format naming convention header file PFNC.h in SphinxLib directory. Replaces the pixel format defines in sphinx lib.h.

## 4.6. V2.1.4 to V2.2.0

### GEVEnableFirewallException

**New**

```
WORD GEVEnableFirewallException(char *app_name_with_path, char
*rule_name, BYTE *status);
```

### GEVQueueRingBuffer

**Old**

```
WORD WINAPI GEVQueueRingBuffer(BYTE cam_nr, BYTE image_buffer_index);
```

**New**

```
WORD WINAPI GEVQueueRingBuffer(BYTE cam_nr, WORD image_buffer_index);
```

### GEVGetImageRingBuffer

**Old**

```
WORD WINAPI GEVGetImageRingBuffer(BYTE cam_nr, IMAGE_HEADER
*image_header, BYTE *image_buffer_index);
```

**New**

```
WORD WINAPI GEVGetImageRingBuffer(BYTE cam_nr, IMAGE_HEADER
*image_header, WORD *image_buffer_index);
```

## 4.7. V2.1.3 to V2.1.4

### GEVSetTraversingFirewallsInterval

**New**

```
WORD GEVSetTraversingFirewallsInterval(BYTE cam_nr, DWORD interval);
```

### GEVGetFeaturePort

**New**

```
WORD GEVGetFeaturePort(BYTE cam_nr, char *feature_name, char
*port_name, int port_name_length);
```

### GEVSetSchemaPath

**New**

```
WORD GEVGetFeaturePort(BYTE cam_nr, char *schema_path);
```

## 4.8. V2.1.2 to V2.1.3

### GEVGetDetailedLog

Supported register read/write log output.

### GEVSetDetailedLog

Supported register read/write log output.

### GEVTestPacketResend

**Old**
```
WORD GEVTestPacketResend(BYTE cam_nr, BYTE on_off, WORD packet_number);
```

**New**
```
WORD GEVTestPacketResend(BYTE cam_nr, BYTE on_off, WORD packet_number,
WORD count);
```

### IMAGE_HEADER

`Added PayloadType, ChunkDataPayloadLength and ChunkLayoutId into struct. Used by GEVGetImage function.`

## 4.9. V2.1.1 to V2.1.2

No new functions or changes in structures were made.

## 4.10. V2.1.0 to V2.1.1

### FEATURE_PARAMETER

Added PollingTime into struct. Used by GEVGetFeatureParameter function.

### GEVTestFindMaxPacketSize

**Old**
```
WORD GEVTestFindMaxPacketSize(BYTE cam_nr, WORD *packet_size, WORD
ps_min, WORD ps_max, WORD ps_inc);
```

**New**
```
WORD WINAPI GEVTestFindMaxPacketSize(BYTE cam_nr, WORD *packet_size,
WORD ps_min, WORD ps_max, WORD ps_inc);
```

## 4.11. V2.0.9 to V2.1.0

### GEVGetXmlSize

**New**
```
WORD WINAPI GEVGetXmlSize(BYTE cam_nr, DWORD *size);
```

### GEVGetXmlFile

**New**
```
WORD WINAPI GEVGetXmlFile(BYTE cam_nr, BYTE **xmlfile);
```

### GEVAcquisitionStartEx

**New**
```
WORD WINAPI GEVAcquisitionStartEx(BYTE cam_nr, DWORD
number_images_to_acquire, DWORD image_size);
```

### GEVGetPacketsOutOfOrder

**New**
```
WORD WINAPI GEVGetPacketsOutOfOrder(BYTE cam_nr, BYTE
*packets_out_of_order);
```

### GEVSetPacketsOutOfOrder

#### New

```
WORD WINAPI GEVSetPacketsOutOfOrder(BYTE cam_nr, BYTE
packets_out_of_order);
```

### GEVTestFindMaxPacketSize

#### Old

```
WORD GEVTestFindMaxPacketSize(BYTE cam_nr, WORD *packet_size);
```

#### New

```
WORD GEVTestFindMaxPacketSize(BYTE cam_nr, WORD *packet_size, WORD
ps_min, WORD ps_max, WORD ps_inc);
```

### GEVGetFeatureInvalidator

#### New

```
WORD WINAPI GEVGetFeatureInvalidator(BYTE cam_nr, char *feature_name,
BYTE index, char *invalidator_name, int str_len);
```

### GEVSetReadWriteMemoryCallback

#### New

```
WORD WINAPI GEVSetReadWriteMemoryCallback(BYTE cam_nr,
READ_WRITE_MEM_CALLBACK_FUNC c_func);
```

### CANCamSendClink

#### Old

```
WORD WINAPI CANCamSendClink(BYTE cam_nr, BYTE count, BYTE
*send_buffer);
```

#### New

```
 WORD WINAPI CANCamSendClink(BYTE cam_nr, WORD count, BYTE
*send_buffer);
```

### CANCamReceiveClink

#### Old

```
WORD WINAPI CANCamReceiveClink(BYTE cam_nr, BYTE count_buffer, BYTE
*recv_buffer, BYTE *recv_count);
```

#### New

```
WORD WINAPI CANCamReceiveClink(BYTE cam_nr, WORD count_buffer, BYTE
*recv_buffer, WORD *recv_count);
```

## 4.12. V2.0.8 to V2.0.9

### MESSAGECHANNEL_CALLBACK_FUNC

#### Old

```
typedef BYTE ( WINAPI *MESSAGECHANNEL_CALLBACK_FUNC )(BYTE cam_nr, int
event_id, int data_length, BYTE *data);
```

**New**

```
typedef BYTE ( WINAPI *MESSAGECHANNEL_CALLBACK_FUNC )(BYTE cam_nr,
MESSAGECHANNEL_PARAMETER mcparam);
```

### GEVGetPixelPtr

**Old**

```
WORD GEVGetPixelPtr(BYTE cam_nr, DWORD offset, DWORD *ptr_adr);
```

**New**

```
WORD GEVGetPixelPtr(BYTE cam_nr, DWORD offset, UINT64 *ptr_adr);
```

### GEVGetImageBuffer

**New**

```
WORD WINAPI GEVGetImageBuffer(BYTE cam_nr, IMAGE_HEADER *image_header,
BYTE *image_buffer);
```

### GEVSetRingBuffer

**New**

```
WORD WINAPI GEVSetRingBuffer(BYTE cam_nr, WORD index, void *buffer);
```

### GEVQueueRingBuffer

**New**

```
WORD WINAPI GEVQueueRingBuffer(BYTE cam_nr, BYTE image_buffer_index);
```

### GEVReleaseRingBuffer

**New**

```
WORD WINAPI GEVReleaseRingBuffer(BYTE cam_nr);
```

### GEVGetImageRingBuffer

**New**

```
WORD WINAPI GEVGetImageRingBuffer(BYTE cam_nr, IMAGE_HEADER
*image_header, BYTE *image_buffer_index);
```

## 4.13. V2.0.7 to V2.0.8

### GEVSetSecureTransfer

**New**

```
WORD WINAPI GEVSetSecureTransfer(BYTE cam_nr, BYTE enable,
SECURE_TRANSFER_CALLBACK_FUNC c_func);
```

### GEVGetSecureTransfer

**New**

```
WORD WINAPI GEVGetSecureTransfer(BYTE cam_nr, BYTE *enable);
```

## 4.14. V2.0.5 to V2.0.6

### GEVSetBufferCount

**New**

```
WORD WINAPI GEVSetBufferCount(BYTE cam_nr, WORD count);
```

### GEVGetBufferCount

**New**

```
WORD WINAPI GEVGetBufferCount(BYTE cam_nr, WORD *count);
```

### GEVSetActionCommand

**New**

```
WORD WINAPI GEVSetActionCommand(BYTE cam_nr, DWORD device_key, DWORD
group_key, DWORD group_mask, DWORD action_time)
```

## 4.15. V2.0.4 to V2.0.5

### MESSAGECHANNEL_CALLBACK_FUNC

**Old**

```
typedef BYTE ( WINAPI *MESSAGECHANNEL_CALLBACK_FUNC )(int event_id, int
data_length, BYTE *data);
```

**New**

```
typedef BYTE ( WINAPI *MESSAGECHANNEL_CALLBACK_FUNC )(BYTE cam_nr, int
event_id, int data_length, BYTE *data);
```

### ERROR_CALLBACK_FUNC

**Old**

```
typedef BYTE ( WINAPI *ERROR_CALLBACK_FUNC )(char *error_str);
```

**New**

```
typedef BYTE ( WINAPI *ERROR_CALLBACK_FUNC )(BYTE cam_nr, char
*error_str);
```

## 4.16. V2.0.2 to V2.0.3

### GEVInit

**Old**

```
WORD WINAPI GEVInit(BYTE cam_nr, CONNECTION *con, ERROR_CALLBACK_FUNC
error_func, BYTE save_xml);

      con.AdapterIP = dis.param[camera-1].AdapterIP;
      con.AdapterMask = dis.param[camera-1].AdapterMask;
      con.IP_CANCam = htonl(dis.param[camera-1].IP);
      con.PortCtrl = 2000;
      con.PortData = 2001;
      strcpy(con.adapter_name,dis.param[camera-1].adapter_name);

      error = GEVInit(camera, &con, NULL, 0);
```

**New**

```
WORD WINAPI GEVInit(BYTE cam_nr, CONNECTION *con, ERROR_CALLBACK_FUNC
error_func, BYTE save_xml,  BYTE open_mode);

      con.AdapterIP = dis.param[camera-1].AdapterIP;
      con.AdapterMask = dis.param[camera-1].AdapterMask;
      con.IP_CANCam = dis.param[camera-1].IP;

      con.PortCtrl = 2000; // set static port
      con.PortData = 2001; // set static port
      or
      con.PortCtrl = 0;    // set port automatically
      con.PortData = 0;    // set port automatically


      strcpy(con.adapter_name,dis.param[camera-1].adapter_name);

      error = GEVInit(camera, &con, NULL, 0,EXCLUSIVE_ACCESS);
```

| Open Mode | Value |
|---|---|
| OPEN_ACCESS | 0 |
| EXCLUSIVE_ACCESS | 1 |
| CONTROL_ACCESS | 2 |

## GEVOpenStreamChannel

**Old**

```
WORD WINAPI GEVOpenStreamChannel(BYTE cam_nr, DWORD ip, WORD port);

      error = GEVOpenStreamChannel(camera, con.AdapterIP, con.PortData);
```

**New**

```
WORD WINAPI GEVOpenStreamChannel(BYTE cam_nr, DWORD ip, WORD port,
DWORD multicast);
      error = GEVOpenStreamChannel(camera, con.AdapterIP, con.PortData,0
);
```

## GEVAcquisitionStart

**Old**

```
WORD WINAPI GEVAcquisitionStart(BYTE cam_nr);
      error = GEVAcquisitionStart(camera);
```

**New**

```
WORD WINAPI GEVAcquisitionStart(BYTE cam_nr, DWORD
number_images_to_acquire);

   acquisition_counter = 0; // start acquisition in the continuous mode
   error = GEVAcquisitionStart(camera,acquisition_counter);
```

| AcquisitionMode | number_images_to_acquire |
|---|---|
| Continuous | 0 (unlimited) |
| SingleFrame | 1 |

| | |
|---|---|
| MultiFrame | 1..n<br>Get the value from the AcquisitionFrameCount node |

## 4.17. V1.x.x to V2.x.x

| V1.x.x | V2.x.x |
|---|---|
| discovery_cancam | GEVDiscovery |
| init_cancam | GEVInit |
| close_cancam | GEVClose |
| init_filter_driver | GEVInitFilterDriver |
| close_filter_driver | GEVCloseFilterDriver |
| send_val | GEVWriteRegister |
| receive | GEVReadRegister |
| get_pixel_ptr | GEVGetPixelPtr |
| debug_info | GEVGetDetailedLog, GEVSetDetailedLog |
| set_memory_size | GEVSetMemorySize |
| get_memory_size | GEVGetMemorySize |
| set_lut | CANCamSetLut |
| get_lut | CANCamGetLut |
| set_message_channel_callback | GEVSetMessageChannelCallback |
| get_local_error | not available |
| receive_memory | GEVWriteMemory |
| send_memory | GEVReadMemory |
| set_net_config | GEVSetNetConfig |
| get_net_config | GEVGetNetConfig |
| force_ip | GEVForceIp |
| set_read_write_parameter | GEVSetReadWriteParameter |
| get_read_write_parameter | GEVGetReadWriteParameter |
| set_heartbeat_rate | GEVSetHeartbeatRate |
| get_heartbeat_rate | GEVGetHeartbeatRate |
| test_packet | GEVTestPacket |
| get_filter_driver_version | GEVGetFilterDriverVersion |
| set_channel_parameter | GEVSetChannelParameter |
| get_channel_parameter | GEVGetChannelParameter |
| --- | GEVOpenStreamChannel |
| --- | GEVCloseStreamChannel |
| --- | GEVInitXml |
| get_root_count | GEVGetFeatureList |
| get_root_name | GEVGetFeatureList |
| get_feature_count | GEVGetFeatureList |
| get_feature_name | GEVGetFeatureList |
| get_sub_feature_name | GEVGetFeatureList |
| get_sub_feature_count | GEVGetFeatureList |
| get_subsub_feature_count | GEVGetFeatureList |
| get_subsub_feature_name | GEVGetFeatureList |
| get_feature_parameter | GEVGetFeatureParameter |
| get_feature_integer | GEVGetFeatureInteger |
| set_feature_integer | GEVSetFeatureInteger |
| get_feature_string | GEVGetFeatureString |
| set_feature_string | GEVSetFeatureString |

| | |
|---|---|
| get_feature_boolean | GEVGetFeatureBoolean |
| set_feature_boolean | GEVSetFeatureBoolean |
| get_feature_command | GEVGetFeatureCommand |
| set_feature_command | GEVSetFeatureCommand |
| set_feature_enumeration | GEVSetFeatureEnumeration |
| get_feature_enumeration | GEVGetFeatureEnumeration |
| get_feature_enumeration_name | GEVGetFeatureEnumerationName |
| get_feature_float | GEVGetFeatureFloat |
| set_feature_float | GEVSetFeatureFloat |
| get_feature_display_name | GEVGetFeatureDisplayName |
| get_feature_tooltip | GEVGetFeatureTooltip |
| set_xml_file | GEVSetXmlFile |
| get_feature_register | GEVGetFeatureRegister |
| set_feature_register | GEVSetFeatureRegister |
| get_feature_unit | GEVGetFeatureUnit |
| get_feature_enable_status | GEVGetFeatureEnableStatus |
| start_grab | GEVAcquisitionStart |
| stop_grab | GEVAcquisitionStop |
| grab_image | GEVGetImage |
| get_image_header | GEVGetImage |
| set_packet_resend | GEVSetPacketResend |
| get_packet_resend | GEVGetPacketResend |
| get_image_fps | GEVGetImageFPS |
| --- | GEVPacketResend |
| --- | GEVGetImageFPS |
| --- | GEVSetBufferCount |
| --- | GEVGetBufferCount |
| --- | GEVTestPacketResend |
| --- | GEVTestFindMaxPacketSize |
| // these functions should not use it | |
| get_pixel_format | CANCamGetPixelFormat |
| get_max_video_window | CANCamGetMaxVideoWindow |
| get_video_window | CANCamGetVideoWindow |
| set_video_window | CANCamSetVideoWindow |
| get_gain | CANCamGetGain |
| set_gain | CANCamSetGain |
| get_gain_min_max | CANCamGetGainMinMax |
| set_gain_auto | CANCamSetGainAuto |
| get_exposure_time | CANCamGetExposureTime |
| set_exposure_time | CANCamSetExposureTime |
| get_exposure_time_min_max | CANCamGetExposureTimeMinMax |
| set_exposure_time_auto | CANCamSetExposureTimeAuto |