Kowa GigE SDK Manual (Library functions)

v1.6.1

Kowa Optronics Co.,Ltd.

1. INTRODUCTION

Kowa GigE SDK is an image acquisition library, which provides access to Kowa Optronics Co., Ltd. cameras. The following document describes the library functions.

2. Kowa GigE SDK

This chapter describes the functions of the Kowa GigE SDK C/C++ library. The following example describes and explains the function:

Item	Contents
Function	A short description of the function.
Syntax	The function's syntax description in C/C++.
Parameter	Description of function parameters.
Description	Description of the purpose and use of the function.
Return	Here you find the type and range of the return values. Many functions return predefined error codes (`GEV_STATUS_...`). Please check `KowaGigE/errors.h` and `KowaGigE/gev_errors.h` for reference.
Reference	List of other routines, which have a relationship to the current function. In addition refer to the GigE Vision specification. For detailed error codes of GEV_STATUS_LOCAL_PROBLEM, please use function get_local error.

Following data types are equivalent:

Type	Byte size
BYTE	1
WORD	2
DWORD	4
BOOL	4

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.

The files in the directory KowaGigEVisionLib/Extras must be in the same directory as the KowaGigEVisionLib.dll. These files are used by the KowaGigEVisionLib.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 KowaGigEVisionLib/extras must be in the same directory as the application. These files are used by the KowaGigEVisionLib.so/.a library to parse the xml file of the device. The *.xsd (shema files) and *.xsl (stylesheet 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

2-1-1. 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);
```

- Parameters
  - `ip_adapter` : It specifies the IP address of the network adapter where the device is connected.
  - `mask_adapter` : It specifies the net mask of the network adapter.
  - `ip_device` : It specifies the ip address of the GigE Vision device to check.
  - `device_status` : It returns the status of the device. See status entry of DEVICE_PARAM struct.
  - `ack_timeout` : It specifies the acknowledge timeout of the status register read.
  - `port` : It specifies the destination port address of the control channel. Value of 0 set the port automatically.

- Description

  This function checks the status of a GigE Vision device.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVDiscovery
  - GEVDiscoveryAdapter

2-1-2. GEVClose

- Function

  Close GigE Vision device.

- Syntax

```
WORD GEVClose(BYTE cam_nr);
```

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].

- Description

  This function closes the communication session (GEV Control Channel) to a GigE Vision device.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVInit

2-1-3. GEVCloseFilterDriver

- Function

Close the Kowa device filter driver of GigE Vision device.

- Syntax

```
WORD GEVCloseFilterDriver(BYTE cam_nr);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
- Description

  This function closes the kowa filter driver for the stream channel of the specified device.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVInitFilterDriver
  - GEVGetFilterDriverVersion

2-1-4. GEVCloseStreamChannel

- Function

  Close GigE Vision device stream channel.

- Syntax

```
WORD GEVCloseStreamChannel(BYTE cam_nr);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
- Description

  This function closes the stream channel of the specified device.

- Return

ErrorCode (see "Error Codes")

- Reference
  - GEVOpenStreamChannel

2-1-5. GEVDiscovery

- Function

  Find all GigE Vision device (Kowa Optronics Co., Ltd. cameras) of all network interface adapters.

- Syntax

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

- Parameters
  - `dis` : It returns number and parameters of the found devices (see Structures and KowaGigEVisionLib.h for the definition of struct DISCOVERY and DEVICE_PARAM).
  - `c_func` : It specifies the discovery callback function.
  - `d_timeout` : It specifies a discovery timeout in ms.
  - `ignore_subnet` : It specifies if subnet ignore flag of discovery message should be set.
  - `port` : It specifies the destination port address of the control channel. Value of 0 set the port automatically.

- Description

  This function can discover up to 128 GigE Vision compliant devices in the network.
  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 128 see DISCOVERY struct see Structures)
  2) `GEVDiscovery(NULL, discovery_callback_func, 200, 0);`
  This call returns the founded devices in the callback function.

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 in sample codes.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVInit
  - GEVCheckDeviceStatus

2-1-6. GEVDiscoveryAdapter

- Function

  Find all GigE Vision device (Kowa Optronics Co., Ltd. cameras) 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);
```

- Parameters
  - `dis` : It returns number and parameters of the found devices (see Structures and KowaGigEVisionLib.h for the definition of struct DISCOVERY and DEVICE_PARAM).
  - `adapter` : It specifies the network interface adapter (see Structures and KowaGigEVisionLib.h for the definition of struct ADAPTER_PARAM).
  - `c_func` : It specifies the discovery callback function.
  - `d_timeout` : It specifies a discovery timeout in ms.
  - `ignore_subnet` : It specifies if subnet ignore flag of discovery message should be set.
  - `port` : It specifies the destination port address of the control channel. Value of 0 set the port automatically.
- Description

  This function can discover up to 50 GigE Vision compliant devices in the network.
  Network byte order for IP addresses.
  There are two methods of obtaining GigE Vision devices.
  See `GEVDiscovery`.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVEnumerateAdapter
  - GEVCheckDeviceStatus

2-1-7. GEVEnableFirewallException

- Function

  Enable firewall exception.

- Syntax

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

- Parameters
  - `app_name_with_path` : It specifies the application with path who wants to enable in the firewall.
  - `rule_name` : It specifies the rule name of the application.
  - `status` : It returns the status.

    Possible values of the parameter `status`.

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

- Description

This function can enable a firewall exception for an application.

- 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

  ErrorCode (see "Error Codes")

2-1-8. GEVEnumerateAdapters

- Function

  Enumerate network interface adapter.

- Syntax

```
WORD GEVEnumerateAdapters(ENUMERATE_ADAPTER* adapter);
```

- Parameters

  - `adapter` : It returns number and parameters of the found network interface adapters (see Structures and KowaGigEVisionLib.h for the definition of struct ENUMERATE_ADAPTER).

- Description

This function can enumerate up to 50 network interface adapter.
Network byte order for IP addresses.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVDiscoveryAdapter

2-1-9. GEVForceIp

- Function

  Force an IP address.

- Syntax

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

- Parameters

  - `new_ip` : It sets IP-address.
  - `subnet` : It sets subnet mask.
  - `gateway` : It sets the gateway address.
  - `mac` : It is the MAC address of the device.
  - `adapter_ip` : It specifies the IP address of the network adapter where is connected the device. `new_ip`, `subnet` and `gateway` parameters are requires network byte order.

- Description

  This function temporary modifies the network configuration of a device.

- Return

  ErrorCode (see "Error Codes")

- Reference

- GEVSetNetConfig
- GEVGetNetConfig

2-1-10. GEVGetChannelParameter

- Function

  Returns channel parameter.

- Syntax

```
WORD GEVGetChannelParameter(BYTE cam_nr, CHANNEL_PARAMETER
*cparam);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `cparam` : It specifies the channel parameter (see Structures and KowaGigEVisionLib.h for definition of CHANNEL_PARAMETER).
- Description

  This function returns the control and stream channel parameter.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVSetChannelParameter

2-1-11. GEVGetDetailedLog

- Function

  Gets detailed log.

- Syntax

```
WORD GEVGetDetailedLog(BYTE cam_nr, BYTE *flags);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `flags` : It returns the conditions for log output. Possible values of the parameter `flags`.

```
// Disable all log outputs.
#define DETAILED_LOG_OFF        0
// Enable information outputs
#define DETAILED_LOG_INFO       1
// Enable warning outputs
#define DETAILED_LOG_WARNING    2
// Enable error outputs
#define DETAILED_LOG_ERROR      4
// Enable register read/write outputs
#define DETAILED_LOG_REGISTER   8
// Enable debug outputs
#define DETAILED_LOG_DEBUG      16
// Enable verbose outputs
#define DETAILED_LOG_VERBOSE    32
```

- Description

  This function returns the detailed log in the parameter `flags`.

- 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

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetDetailedLog

2-1-12. GEVGetConnectionStatus

- Function

  Returns connection status.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `status` : It is a pointer to a variable containing the connection status. (0 = OK, 1 = timeout, 2 = access denied)
  - `lib_status` : It is a pointer to a variable containing the evaluation status. (0 = OK, 1 = evaluation expired (deprecated), 2 = library is running in evaluation mode)

- Description

  This function returns the connection status.

- Return

  ErrorCode (see "Error Codes")

2-1-13. GEVGetFilterDriverVersion

- Function

  Get KOWA filter driver version.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `version_major` : It is a pointer to a variable containing the major version.
  - `version_minor` : It is a pointer to a variable containing the minor version.
- Description

  This function returns the filter driver version of the specified device. As the filter driver is bound to each device instance, the parameter `cam_nr` is needed, although no communication to the device is done.
  If the filter driver is not activated the return value in `version_major` and `version_minor` is 0.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVInitFilterDriver
  - GEVCloseFilterDriver

2-1-14. GEVGetHeartbeatRate

- Function

  Returns heartbeat rate.
- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].

- `heartbeat_rate` : It is a pointer to a variable containing the heartbeat rate.

- Description

  This function returns heartbeat rate in millisecond.

- Return

  ErrorCode (see "Error Codes")

2-1-15. GEVGetMemorySize

- Function

  Get the buffer size required to store one image.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `mem_size` : It is a pointer to a variable containing the memory size.

- Description

  This function returns the image acquisition memory-size.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetMemorySize

2-1-16. GEVGetNetConfig

- Function

  Get network configuration.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance.
  - `dhcp` : It returns dhcp enable status.
  - `ip` : It returns IP-address.
  - `subnet` : It returns subnet mask.
  - `gateway` : It returns the gateway address. Network byte order for `ip`, `subnet` and `gateway` parameter.
- Description

  This function reads the network configuration of the device.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVSetNetConfig

2-1-17. GEVGetPixelPtr (Deprecated)

- Function

  Returns the logical address of the image memory.
- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].

- `offset` : It shifts the returned start of the image buffer. It can be used when working with memory divided in several pages for different images.
- `ptr_adr` : It gives access to the pixeldata.

- Description

  This function returns the logical memory address of the local image buffer.
  This function is deprecated, it is recommended to use GEVGetImageBuffer.

- Example

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

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVGetImage
  - GEVGetImageBuffer

2-1-18. GEVGetReadWriteParameter

- Function

  Returns read/write parameter.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `ack_timeout` : It is a pointer to a variable containing the acknowledge timeout in millisecond.

- `retry_count` : It is a pointer to a variable containing the retry count for the read/write commands.

- Description

  This function returns read/write parameter for the `GEVReadRegister`, `GEVWriteRegister`, `GEVReadMemory` and `GEVWriteMemory` functions.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetReadWriteParameter

2-1-19. 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);
```

- Parameters

  - `cam_nr` : A your desired instance number for the GigE Vision device [1..50].
  - `con` : It specifies the connection data (see Structures and KowaGigEVisionLib.h for definition of CONNECTION).
  - `error_func` : It describes the error callback function. The error function will be executed, when an error occured.
  - `save_xml` : It specifies the flag to save the xml file to disk. Directory is fixed to the location of the library.
  - `open_mode` : It specifies the control privileges of the device.

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

- Description

  This function opens a control channel to a GigE Vision device. In addition the heartbeat thread is started.
  Network byte order for IP addresses.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVClose
  - GEVGetLocalError

2-1-20. GEVInitEx

- Function

  Initialize a GigE Vision device.

- Syntax

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

- Parameters

  - `cam_nr` : The instance number for the GigE Vision device is assigned automatically and it is set to parameter[1..50].
  - `con` : It specifies the connection data (see Structures and KowaGigEVisionLib.h for definition of CONNECTION).
  - `error_func` : It describes the error callback function. The error function will be executed, when an error occured.

- `save_xml` : It specifies the flag to save the xml file to disk. Directory is fixed to the location of the library.
- `open_mode` : It specifies the control privileges of the device.

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

- Description

  This function opens a control channel to a GigE Vision device. In addition the heartbeat thread is started.
  Network byte order for IP addresses.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVClose
  - GEVGetLocalError

2-1-21. GEVInitFilterDriver

- Function

  Init the Kowa filter driver of GigE Vision device.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].

- Description

  This function activates the Kowa filter driver for stream channel of the specified device.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVCloseFilterDriver
  - GEVGetFilterDriverVersion

2-1-22. GEVIssueActionCommandAdapter

- Function

  Issue an action command from the adapter.

- Syntax

```
WORD GEVIssueActionCommandAdapter(const DWORD adapterip[],
size_t count, const ACTION_KEYS* keys, UINT64 actiontime, DWORD
timeoutms, ACTION_ACKNOWLEDGE_CALLBACK_FUNC callback, void*
param);
```

- Parameters

  - `adapterip`: Specifies array of adapter IP addresses for issue the action command to.
  - `count`: Specifies length of the device number array.
  - `keys`: Specifies device_key, group_mask, and group_key.
  - `actiontime`: Specifies the time to execute the action command. If 0 is specified, each device will execute the action command immediately.
  - `timeoutms`: Specifies the maximum waiting time for a response after issuing the action command.
  - `callback`: Specifies the callback to be called for each response to the action command. Since the action command is one-to-many communication, the callback will be called multiple times.
  - `param`: Specifies a general-purpose pointer to be given as the last argument of callback.

- Description

  This function issues an action command from the adapter.

  Since the action command is broadcast, it will also be delivered to other devices on the same network. To determine whether the device should execute the action, keys need to be specified separately.

  Keys need to be set on the device in advance. There are three types: device_key, group_key, and group_mask.

  The device_key is set using the feature name "ActionDeviceKey". The group_key and group_mask are set using feature names such as "ActionGroupKey" and "ActionGroupMask".

  Use the following prototype for the callback:

```
BYTE WINAPI callback(KOWAGIGEVISION_CAMNR cam_nr, DWORD fromip,
WORD status, void* param);
```

  - `cam_nr`: Always GEV_INVALID_CAMNR.
  - `fromip`: The IP address of the device that returned the action command response.
  - `status`: The error code of the action command response.
  - `param`: The general-purpose pointer given to this function as param. If the callback returns 0, this function will wait for the next action command response until the time specified by timeoutms. If the callback returns 1, control will be returned without waiting for the next action command response.

- Return

  Error code (see "Error Codes" section)

2-1-23. GEVIssueActionCommandBroadcast

- Function

  Issue an action command to each device.

- Syntax

```
WORD GEVIssueActionCommandBroadcast(const BYTE cam_nrn[],
size_t count, const ACTION_KEYS* keys, UINT64 actiontime, DWORD
timeoutms, ACTION_ACKNOWLEDGE_CALLBACK_FUNC callback, void*
param);
```

- Parameters
  - `cam_nrn`: Specifies array of device numbers for issue the action command.
  - `count`: Specifies length of the device number array.
  - `keys`: Specifies device_key, group_mask, and group_key.
  - `actiontime`: Specifies the time to execute the action command. If 0 is specified, each device will execute the action command immediately.
  - `timeoutms`: Specifies the maximum waiting time for a response after issuing the action command.
  - `callback`: Specifies the callback to be called for each response to the action command. Since the action command is one-to-many communication, the callback will be called multiple times.
  - `param`: Specifies a general-purpose pointer to be given as the last argument of callback.
- Description

  This function issues an action command to each device.

  Since the action command is broadcast, it will also be delivered to other devices on the same network. To determine whether the device should execute the action, keys need to be specified separately.

  Keys need to be set on the device in advance. There are three types: device_key, group_key, and group_mask.

  The device_key is set using the feature name "ActionDeviceKey". The group_key and group_mask are set using feature names such as "ActionGroupKey" and "ActionGroupMask".

  Use the following prototype for the callback:

```
BYTE WINAPI callback(KOWAGIGEVISION_CAMNR cam_nr, DWORD fromip,
WORD status, void* param);
```

- `cam_nr`: The device that returned the action command response if it is in devices, otherwise GEV_INVALID_CAMNR.
- `fromip`: The IP address of the device that returned the action command response.
- `status`: The error code of the action command response.
- `param`: The general-purpose pointer given to this function as param. If the callback returns 0, this function will wait for the next action command response until the time specified by timeout. If the callback returns 1, control will be returned without waiting for the next action command response.

- Return

  Error code (see "Error Codes" section)

2-1-24. GEVIssueActionCommandAdapterDirected

- Function

  Issue an action command from the adapter.

- Syntax

```
WORD GEVIssueActionCommandAdapterDirected(const DWORD
adapterip[], size_t count, const ACTION_KEYS* keys, UINT64
actiontime, DWORD timeoutms, ACTION_ACKNOWLEDGE_CALLBACK_FUNC
callback, void* param);
```

- Parameters
  - `adapterip`: Specifies array of adapter IP addresses for issue the action command.
  - `count`: Specifies length of the device number array.
  - `keys`: Specifies device_key, group_mask, and group_key.

- `actiontime`: Specifies the time to execute the action command. If 0 is specified, each device will execute the action command immediately.
- `timeoutms`: Specifies the maximum waiting time for a response after issuing the action command.
- `callback`: Specifies the callback to be called for each response to the action command. Since the action command is one-to-many communication, the callback will be called multiple times.
- `param`: Specifies a general-purpose pointer to be given as the last argument of callback.

- Description

  This function issues an action command from the adapter. This function sends directed broadcast (e.g. "192.168.1.255"). The `GEVIssueActionCommandAdapter` function sends limited broadcast ("255.255.255.255").

  Since the action command is broadcast, it will also be delivered to other devices on the same network. To determine whether the device should execute the action, keys need to be specified separately.

  Keys need to be set on the device in advance. There are three types: device_key, group_key, and group_mask.

  The device_key is set using the feature name "ActionDeviceKey". The group_key and group_mask are set using feature names such as "ActionGroupKey" and "ActionGroupMask".

  Use the following prototype for the callback:

```
BYTE WINAPI callback(KOWAGIGEVISION_CAMNR cam_nr, DWORD fromip,
WORD status, void* param);
```

  - `cam_nr`: Always GEV_INVALID_CAMNR.
  - `fromip`: The IP address of the device that returned the action command response.
  - `status`: The error code of the action command response.
  - `param`: The general-purpose pointer given to this function as param. If the callback returns 0, this function will wait for the next action command response until the time

specified by timeoutms. If the callback returns 1, control will be returned without waiting for the next action command response.

- Return

  Error code (see "Error Codes" section)

- Reference

  GEVIssueActionCommandAdapterDirected

2-1-25. GEVIssueActionCommandBroadcastDirected

- Function

  Issue an action command to each device.

- Syntax

```
WORD GEVIssueActionCommandBroadcastDirected(const BYTE
cam_nrn[], size_t count, const ACTION_KEYS* keys, UINT64
actiontime, DWORD timeoutms, ACTION_ACKNOWLEDGE_CALLBACK_FUNC
callback, void* param);
```

- Parameters
  - `cam_nrn`: Specifies array of device numbers for issue the action command.
  - `count`: Specifies length of the device number array.
  - `keys`: Specifies device_key, group_mask, and group_key.
  - `actiontime`: Specifies the time to execute the action command. If 0 is specified, each device will execute the action command immediately.
  - `timeoutms`: Specifies the maximum waiting time for a response after issuing the action command.
  - `callback`: Specifies the callback to be called for each response to the action command. Since the action command is one-to-many communication, the callback will be called multiple times.

- `param`: Specifies a general-purpose pointer to be given as the last argument of callback.

- Description

  This function issues an action command from the adapter. This function sends directed broadcast (e.g. "192.168.1.255"). The `GEVIssueActionCommandBroadcast` function sends limited broadcast ("255.255.255.255").

  Since the action command is broadcast, it will also be delivered to other devices on the same network. To determine whether the device should execute the action, keys need to be specified separately.

  Keys need to be set on the device in advance. There are three types: device_key, group_key, and group_mask.

  The device_key is set using the feature name "ActionDeviceKey". The group_key and group_mask are set using feature names such as "ActionGroupKey" and "ActionGroupMask".

  Use the following prototype for the callback:

```
BYTE WINAPI callback(KOWAGIGEVISION_CAMNR cam_nr, DWORD fromip,
WORD status, void* param);
```

  - `cam_nr`: The device that returned the action command response if it is in devices, otherwise GEV_INVALID_CAMNR.
  - `fromip`: The IP address of the device that returned the action command response.
  - `status`: The error code of the action command response.
  - `param`: The general-purpose pointer given to this function as param. If the callback returns 0, this function will wait for the next action command response until the time specified by timeout. If the callback returns 1, control will be returned without waiting for the next action command response.

- Return

  Error code (see "Error Codes" section)

- Reference

  GEVIssueActionCommandBroadcast

2-1-26. GEVIssueActionCommandUnicast

- Function

  Issue an action command to a single device.

- Syntax

```
WORD GEVIssueActionCommandUnicast(BYTE cam_nr, const
ACTION_KEYS* keys, UINT64 actiontime);
```

- Parameters
  - `cam_nr`: Specifies the GigE Vision device instance [1-50].
  - `keys`: Specifies device_key, group_mask, and group_key.
  - `actiontime`: Specifies the time to execute the action command. If 0 is specified, each device will execute the action command immediately.

- Description

  This function issues an action command to a single device.

  Like GEVIssueActionCommandBroadcast, keys need to be specified.

  Keys need to be set on the device in advance. There are three types: device_key, group_key, and group_mask.

  The device_key is set using the feature name "ActionDeviceKey". The group_key and group_mask are set using feature names such as "ActionGroupKey" and "ActionGroupMask".

- Return

  Error code (see "Error Codes" section)

- Reference
  - GEVIssueActionCommandBroadcast

2-1-27. GEVOpenStreamChannel

- Function

Open GigE Vision device stream channel.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `ip` : It specifies the destination IP address of the stream channel.
  - `port` : It specifies the destination port address of the stream channel.
  - `multicast` : It specifies the multicast IP address of the stream channel, address 0.0.0.0 disables multicast.
- Description

  This function opens the stream channel of the specified GigE Vision device.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVCloseStreamChannel

2-1-28. GEVReadRegister

- Function

  Receive data from a GigE Vision device.
- Syntax

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

- Parameters

- `cam_nr` : It specifies the GigE Vision device instance [1..50].
- `cmd` : It specifies the register address.
- `cnt` : It specifies the number of 32bit reads.
- `ptr` : It returns a pointer to the data that has been read.

- Description

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

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVSetReadWriteParameter
  - GEVWriteRegister

2-1-29. GEVReadMemory

- Function

  Receive memory data from a GigE Vision device.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `maddr` : It specifies the memory address.
  - `cnt` : It specifies the number of bytes to read.
  - `pptr` : It returns a pointer to the received memory data.

- Description

This function reads a memory block from the specified GigE Vision device.
This function implements the GEV Read_Mem command.
To get the write status in percent use the `GEVSetReadWriteMemoryCallback` function to set a callback function.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVWriteMemory
  - GEVSetReadWriteMemoryCallback

2-1-30. GEVSetActionCommand

- Function

  Set action command.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `device_key` : It specifies the device key to authorize the action on this device.
  - `group_key` : It specifies the group key to define a group of devices on which the actions have to be executed.
  - `group_mask` : It specifies the group mask to be used to filter out some of these devices from the group.
  - `action_time` : The action command is executed after this set time. Unit : [s].

- Description

  This function sets the action command.

- Return

  ErrorCode (see "Error Codes")

2-1-31. GEVSetChannelParameter

- Function

  Set channel parameter.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `cparam` : It specifies the channel parameter (see Structures and KowaGigEVisionLib.h for definition of CHANNEL_PARAMETER).

- Description

  This function sets the control and stream channel parameter.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetChannelParameter

2-1-32. GEVSetDetailedLog

- Function

  Sets detailed log.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `flags` : It specifies log output options. Possible values of the parameter `flags`.

```
// Disable all log outputs.
#define DETAILED_LOG_OFF            0
// Enable information outputs
#define DETAILED_LOG_INFO           1
// Enable warning outputs
#define DETAILED_LOG_WARNING        2
// Enable error outputs
#define DETAILED_LOG_ERROR          4
// Enable register read/write outputs
#define DETAILED_LOG_REGISTER       8
// Enable debug outputs
#define DETAILED_LOG_DEBUG          16
// Enable verbose outputs
#define DETAILED_LOG_VERBOSE        32
```

- Description

  This function sets the detailed log with the parameter `flags`.

- Example

  Enable info and error log outputs.

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

- Return

ErrorCode (see "Error Codes")

- Reference
    - GEVGetDetailedLog

2-1-33. GEVSetHeartbeatRate

- Function

  Set heartbeat rate.

- Syntax

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

- Parameters
    - `cam_nr` : It specifies the GigE Vision device instance [1..50].
    - `heartbeat_rate` : It specifies the heartbeat timeout value in millisecond.

- Description

  This function sets the heartbeat timeout register and the timeout value for connection checking.
  The heartbeat timeout register of the device is set to `heartbeat_rate` 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

  ErrorCode (see "Error Codes")

2-1-34. GEVSetMemorySize (Deprecated)

- Function

  Set image memory size.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `mem_size` : It specifies the memory size in bytes.
- Description

  This function reallocates memory as framebuffer. The reference to the pixel data must be updated with `GEVGetPixelPtr`.
  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

  ErrorCode (see "Error Codes")
- Reference
  - GEVGetMemorySize
  - GEVGetPixelPtr
  - GEVGetImageBuffer

2-1-35. GEVSetMessageChannel

- Function

  Bind function to message channel event.
- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].

- `port` : It specifies the message channel port to open.
- `c_func` : It describes the callback function.

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

- Description

  This function binds a custom function to the message channel, which will be executed, when an event occurred.

- Return

  ErrorCode (see "Error Codes")

2-1-36. GEVSetNetConfig

- Function

  Set network configuration.

- Syntax

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

- Parameters
  - `cam_nr`: It specifies the GigE Vision device instance [1..50].
  - `dhcp` : It sets the dhcp enable status.
  - `ip` : It sets IP-address.
  - `subnet` : It sets the subnet mask.
  - `gateway` : It sets the gateway address. Network byte order for `ip`, `subnet` and `gateway` parameter.

- Description

This function modifies the network configuration of a GigE Vision device.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVGetNetConfig

2-1-37. GEVSetReadWriteMemoryCallback

- Function

  Associate function with read/write memory functions.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `c_func` : It describes the callback function.

```
typedef BYTE (WINAPI *READ_WRITE_MEM_CALLBACK_FUNC )(BYTE
cam_nr, int s_cnt);
```

- Description

  This function associates a custom function to get the read/write status in percent.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVReadMemory

- GEVWriteMemory

2-1-38. GEVSetReadWriteParameter

- Function

  Sets read/write parameter.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `ack_timeout` : It specifies the acknowledge timeout in ms.
  - `retry_count` : It specifies the number of retries if read/write command wasn't acknowledged.

- Description

  This function sets the read/write parameter for the parameter for the `GEVReadRegister`, `GEVWriteRegister`, `GEVReadMemory` and `GEVWriteMemory` funtions.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetReadWriteParameter

2-1-39. GEVTestPacket

- Function

  Trigger GigE Vision test packet.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `packet_size` : It returns the packet size of the test packet.
- 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. Size of the test packet is equal to the stream channel packet size setting.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVTestFindMaxPacketSize

2-1-40. GEVSetTraversingFirewallsInterval

- Function

  Set the interval in seconds for traversing firewalls.
- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `interval` : It specifies the time in seconds between the packets to be sent to the stream channel.
- Description

This function sets the interval in seconds for traversing firewalls to the specified GigE Vision device.
Traversing firewalls is only available when the stream channel source port register is available in the device.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVOpenStreamChannel

2-1-41. GEVWriteMemory

- Function

  Send memory data to a GigE Vision device.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `maddr` : It specifies the memory address.
  - `cnt` : It specifies the number of bytes to write.
  - `pptr` : It specifies the pointer to the data to be written.

- Description

  This function writes a memory block to the specified GigE Vision device.
  This function implements the Write_Mem command.
  To get the write status in percent use the `GEVSetReadWriteMemoryCallback` function to set a callback function.

- Return

  ErrorCode (see "Error Codes")

- Reference
    - GEVReadMemory
    - GEVSetReadWriteMemoryCallback

2-1-42. GEVWriteRegister

- Function

  Send register data to GigE Vision device.

- Syntax

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

- Parameters
    - `cam_nr` : It specifies the GigE Vision device instance [1..50].
    - `cmd` : It specifies the register address.
    - `cnt` : It specifies the number of 32bit words to be written.
    - `pptr` : It specifies the pointer to the data to be written.

- Description

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

- Example

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

- Return

ErrorCode (see "Error Codes")

- Reference
    - GEVReadRegister

2-2. XML-FUNCTIONS

2-2-1. GEVExecuteFeatureCommand

- Function

  Execute command value of a dedicated feature, descripted in xml file.

- Syntax

```
WORD GEVExecuteFeatureCommand(BYTE cam_nr, const char*
feature_name);
```

- Parameters
    - `cam_nr` : It specifies the GigE Vision device instance [1..50].
    - `feature_name` : It specifies feature name (e.g. AcquisitionStart)
- Description

  This function execute the command of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

2-2-2. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies the feature name (e.g. LUTEnable).
  - `bool_value` : It returns the Boolean value.
- Description

  This functions returns the Boolean value of a feature descripted in xml-file.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVSetFeatureBoolean
  - GEVGetFeatureParameter

2-2-3. GEVGetFeatureCommand (Deprecated)

- 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature node name (e.g. AcquisitionStart).

- `cmd_value` : It returns the command value.

- Description

  This function returns the command value of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetFeatureCommand
  - GEVGetFeatureParameter

2-2-4. 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);
```

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Width).
  - `display_name` : It returns the display name.
  - `display_name_length` : It specifies length of the display name pointer.

- Description

  This function returns the display name of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
    - GEVGetFeatureTooltip

2-2-5. 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);
```

- Parameters
    - `cam_nr` : It specifies the GigE Vision device instance [1..50].
    - `feature_name` : It specifies feature name (e.g. Width).
    - `enable` : It returns the status.
- Description

  This function returns the access status of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

2-2-6. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Pixelformat).
  - `enum_name` : It returns feature enumeration value.
  - `str_len` : It specifies length of feature name pointer.
- Description

  This function returns the current enumeration value of an enumeration feature descripted in xml-file.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVSetFeatureEnumeration
  - GEVGetFeatureEnumerationName
  - GEVGetFeatureParameter

2-2-7. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Pixelformat).
  - `enum_index` : It specifies enumeration index.

- `enum_name` : It returns feature name of enumeration name (e.g. Mono8).
- `str_len` : It specifies length of feature name pointer.

- Description

  This function returns an enumeration value indexed by `enum_index` of a feature descripted in xml-file.
  If returned string length of `enum_name` is equal 0, then enumeration name is not available.
  For a complete list of possible enum values call `GEVGetFeatureParameter`.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetFeatureEnumeration
  - GEVGetFeatureEnumeration
  - GEVGetFeatureParameter

2-2-8. GEVGetFeatureFloat

- Function

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

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies the feature name (e.g. DeviceTemperature).
  - `float_value` : It returns the float value.

- Description

This function returns the float value of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetFeatureFloat

2-2-9. 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);
```

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies the feature name (e.g. Width).
  - `int_value` : It returns feature integer value.

- Description

  This function returns the integer value of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetFeatureInteger
  - GEVGetFeatureParameter

2-2-10. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Width)
  - `index` : It specifies the invalidator index. To obtain the total number of invalidators, use the function `GEVGetFeatureParameter`.
  - `invalidator_name` : It returns invalidator name.
  - `str_len` : It specifies length of `invalidator_name` pointer.

- Description

  This function returns the invalidator of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetFeatureParameter

2-2-11. GEVGetFeatureList

- Function

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

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `featureListPtr` : It specifies the feature list.
  - `maxLevel` : It returns the maximal level to list.
- Description

  This function returns a pointer to the feature list descripted in xml-file.
  XMLs have a hierarchical structure.
- Example

```
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

  ErrorCode (see "Error Codes")

2-2-12. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Width).
  - `f_param` : It returns the parameters of the given feature (see Structures and KowaGigEVisionLib.h for definition of struct FEATURE_PARAMETER).
- Description

  This function returns properties of a device feature descripted in xml-file.
- 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

  ErrorCode (see "Error Codes")

2-2-13. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Width).
  - `port_name` : It returns port name.
  - `port_name_length` : It specifies length of `port_name` pointer.
- Description

  This function returns the port of a feature descripted in xml-file.
- Return

  ErrorCode (see "Error Codes")

2-2-14. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. LUTValueAll).
  - `len` : It specifies length of buffer/register.
  - `pbuffer` : It returns buffer values.

- Description

  This function returns values of a register feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetFeatureRegister
  - GEVGetFeatureParameter

2-2-15. 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);
```

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. DeviceVersion).
  - `str_value` : It returns feature string value.

- Description

  This function returns the string value of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetFeatureString

- GEVGetFeatureParameter

2-2-16. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Width).
  - `tooltip_name` : It returns tooltip string.
  - `tooltip_name_length` : It specifies length of tooltip_name pointer.
- Description

  This function returns the tooltip string of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetFeatureDisplayName

2-2-17. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. DeviceTemperature).
  - `unit_name` : It returns unit string (e.g. C).
  - `unit_name_length` : It specifies length of `unit_name` pointer.
- Description

  This function returns a unit string of a feature descripted in xml-file.
- Return

  ErrorCode (see "Error Codes")

2-2-18. GEVGetXmlFile

- Function

  Get the xml file data.
- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `xmlfile` : It is the pointer to the output destination buffer pointer. This function uses a user-allocated buffer. The memory size to be allocated can be obtained with the `GEVGetXmlSize` function.
    (Caution:The return value of this function is not terminated with \0.)
- Description

This function returns the contents of the device's xml file. The output destination buffer must be provided by the user, as shown in the following code. \

```
DWORD size;
GEVGetXmlSize(cam_nr, &size);
BYTE* buffer = (BYTE*)malloc(size);
GEVGetXmlFile(cam_nr, &buffer);
```

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVGetXmlSize

2-2-19. GEVGetXmlSize

- Function

  Get the size of the device xml file.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `size` : It return the xml file size.

- Description

  This function returns the size of the xml-file of the device.
  Call this function before using `GEVGetXmlFile`.

- Return

  ErrorCode (see "Error Codes")

- Reference

- GEVGetXmlFile

2-2-20. GEVInitXml

- Function

  Initialize xml.

- Syntax

```
WORD GEVInitXml(BYTE cam_nr);
```

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].

- Description

  This function gets the xml-file from the device and builds the feature list in the library.
  This function also needs the schema files in the library/program path.
  The schema files can you find in the KowaGigEVisionLib/Extra directory.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetXmlFile

  - GEVSetSchemaPath

2-2-21. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. LUTEnable)
  - `bool_value` : It specifies new Boolean value.
- Description

  This function writes the Boolean value of a feature descripted in xml-file.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVGetFeatureBoolean
  - GEVGetFeatureParameter

2-2-22. GEVSetFeatureCommand (Deprecated)

- Function

  Set command value of a dedicated feature, descripted in xml file.
- Syntax

```
WORD GEVSetFeatureCommand(BYTE cam_nr, char *feature_name,
DWORD cmd_value);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. AcquisitionStart)
  - `cmd_value` : It specifies command value.
- Description

  This function writes the command value of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetFeatureCommand
  - GEVGetFeatureParameter

2-2-23. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Pixelformat)
  - `enum_name` : It specifies the feature's enumeration name (e.g. Mono8).
  - `str_len` : It specifies length of feature name pointer.

- Description

  This function writes an enumeration value to a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetFeatureEnumeration
  - GEVGetFeatureEnumerationName

- GEVGetFeatureParameter

2-2-24. GEVSetFeatureFloat

- Function

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

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. ExposureTime)
  - `float_value` : It specifies new float value.
- Description

  This function writes a float value to a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetFeatureFloat

2-2-25. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. Width)
  - `int_value` : It specifies the new integer value.
- Description

  This function writes an integer value of a feature descripted in xml-file.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVGetFeatureInteger
  - GEVGetFeatureParameter

2-2-26. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. LUTValueAll)

- `len` : It specifies the length of buffer.
- `pbuffer` : It specifies the new buffer values.

- Description

  This function writes buffer values to a register feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetFeatureRegister
  - GEVGetFeatureParameter

2-2-27. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `feature_name` : It specifies feature name (e.g. DeviceUserID)
  - `str_value` : It specifies the new string value.

- Description

  This function writes the string value of a feature descripted in xml-file.

- Return

  ErrorCode (see "Error Codes")

- Reference
    - GEVGetFeatureInteger
    - GEVGetFeatureParameter

2-2-28. GEVSetSchemaPath

- Function

  Set the path of the schema files.

- Syntax

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

- Parameters
    - `cam_nr` : It specifies the GigE Vision device instance [1..50].
    - `schema_path` : It specifies the new path of the schema files.

- Description

  This function sets the new path of the schema files.
  The schema file is used to check the validity of the device's XML.
  This function is used exclusively with function `GEVSetSchemaFromZip`.
  Default path of the schema files are current program/library path.
  This function must be called after `GEVInit` function and before the `GEVInitXml` function.

- Return

  ErrorCode (see "Error Codes")

- Reference
    - GEVInitXml
    - GEVSetSchemaFromZip

2-2-29. GEVSetSchemaFromZip

- Function

Set the the schema file as a zip file.

- Syntax

```
WORD GEVSetSchemaFromZip(BYTE cam_nr, void* schema_zip, size_t
len);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `schema_zip` : It specifies the pointer of zip binary data.
  - `len` : It specifies the length of zip binary data.
- Description

  This function sets the schema file from a zip file.
  This function is used exclusively with function `GEVSetSchemaPath`.
  Default path of the schema files are current program/library path.
  This function must be called after `GEVInit` function and before the `GEVInitXml` function.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVInitXml
  - GEVSetSchemaPath

2-2-30. GEVSetXmlFile

- Function

  Set xml file in the library.

- Syntax

```
WORD GEVSetXmlFile(BYTE cam_nr, char *xml_name);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `xml_name` : It specifies path to new xml file to read and use in the library.
- Description

  This function sets the xml-file in the library.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVInitXml

2-3. CAMERA FUNCTIONS

2-3-1. GEVAcquisitionStart

- Function

  Start image acquisition.
- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `number_images_to_acquire` : It specifies 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

AcquisitionMode	Number_images_to_acquire
MultiFrame	2...n

- 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`.

  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
    - AcquisitionFrameCount (MultiFrame only)
    - Width
    - Height
    - PixelFormat
    - PayloadSize
    - AcquisitionStart

- Return

  ErrorCode (see "Error Codes")

- Reference

    - GEVAcquisitionStop
    - GEVGetImage

2-3-2. GEVAcquisitionStartEx

- Function

  Start image acquisition without library internal xml handling.

- Syntax

```
WORD GEVAcquisitionStartEx(BYTE cam_nr , DWORD
number_images_to_acquire, DWORD image_size, DWORD image_width,
DWORD image_height, DWORD pixel_format);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `number_images_to_acquire` : It specifies 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.
  - `image_size` : It specifies the size of one payload block (image).
  - `image_width` : It specifies the image width.
  - `image_height` : It specifies the image height.
  - `pixel_format` : It specifies the pixel format.
- 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`.
  Use this function if xml handling is done outside of KowaGigEVisionLib. Do not forget to send acquisition start command to the device.

AcquisitionMode	Number_images_to_acquire
Continuous	0 (unlimited)
SingleFrame	1
MultiFrame	2...n

  Get the value from the AcquisitionFrameCount node

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVAcquisitionStart

- GEVAcquisitionStop
- GEVGetImage

2-3-3. GEVAcquisitionStop

- Function

  Stop image acquisition.

- Syntax

```
WORD GEVAcquisitionStop(BYTE cam_nr);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
- Description

  This function stops the image acquisition tread and sends acquisition stop command to the device, if xml handling is done by KowaGigEVisionLib.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVAcquisitionStart
  - GEVGetImage

2-3-4. GEVGetBufferCount

- Function

  Returns number of buffers of library managed ring buffer.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `count` : It returns the number of buffers of the image ring buffer.
- Description

  This function gets the number of buffers in the library managed ring buffer.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVSetBufferCount

2-3-5. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `image_header` : It returns information of the image (see Structures and KowaGigEVisionLib.h for definition of IMAGE_HEADER).
- Description

  Get an image form the ring buffer and copy it to the internal image memory.
  The pointer to the internal image memory you can get it with the function `GEVGetPixelPtr`.
  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

  ErrorCode (see "Error Codes")

- Reference
    - GEVAcquisitionStart
    - GEVAcquisitionStop
    - GEVGetPixelPtr

2-3-6. 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);
```

- Parameters
    - `cam_nr` : It specifies the GigE Vision device instance [1..50].
    - `image_header` : It returns information of the image (see Structures and KowaGigEVisionLib.h for definition of IMAGE_HEADER).
    - `image_buffer` : It specifies the image buffer. The required buffer size can be obtained with `GEVGetMemorySize`.

- 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.

- Return

  ErrorCode (see "Error Codes")

- Reference
    - GEVAcquisitionStart
    - GEVAcquisitionStop

2-3-7. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `image_header` : It returns information of the image (see Structures and KowaGigEVisionLib.h for definition of IMAGE_HEADER).
  - `index` : It returns the buffer number of the destination buffer.
- 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.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVAcquisitionStart
  - GEVAcquisitionStop
  - GEVSetRingBuffer
  - GEVQueueRingBuffer
  - GEVReleaseRingBuffer

2-3-8. GEVGetImageFPS

- Function

returns the number of the acquired frames per second.

- Syntax

```
WORD GEVGetImageFPS(BYTE cam_nr, double *fps);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `fps` : It returns the frames per second.
- Description

  This function returns the number of acquired frames per second.
- Return

  ErrorCode (see "Error Codes")

2-3-9. GEVGetPacketResend

- Function

  Gets packet resend activation status.
- Syntax

```
WORD GEVGetPacketResend(BYTE cam_nr, BYTE *enable);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance.
  - `enable` : It returns the packet resend activation status (0 = disabled, 1 = enabled).
- Description

  This function gets packet resend activation status.
- Return

  ErrorCode (see "Error Codes")

- Reference
    - GEVSetPacketResend

2-3-10. GEVGetPacketsOutOfOrder

- Function

  Returns the number of handled packets out of order.

- Syntax

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

- Parameters
    - `cam_nr` : It specifies the GigE Vision device instance [1..50].
    - `packets_out_of_order` : It returns the number of packets taken into account, before sending packet resend requests.

- Description

  this functions returns the number of packets taken into account, before sending packet resend requests.

- Return

  ErrorCode (see "Error Codes")

- Reference
    - GEVSetPacketsOutOfOrder

2-3-11. GEVGetSecureTransfer (Not available)

- Function

  Returns secure transfer mode enable status.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `enable` : It is a pointer to a variable containing the enable status (0 = disable, 1 = 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.
  This function returns the enable status of secure transfer mode.
  Note: this function is no longer functional.
- Return

  ErrorCode (see "Error Codes")
- Reference
  - GEVSetSecureTransfer

2-3-12. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `stream_channel` : It specifies the number of the stream channel. (0 is first stream channel)

- `block_id` : It specifies the block ID.
- `first_packet_id` : It specifies the first packet to resend.
- `last_packet_id` : It specifies the last packet to resend.

- Description

  This function sends the packet resend command.

- Example

  packet 100 -105 of block 20 shall be resent:

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

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVSetPacketResend
  - GEVGetPacketResend

2-3-13. GEVQueueRingBuffer

- Function

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

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `image_buffer_index` : It specifies the index of the ring buffer.

- 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.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetRingBuffer
  - GEVReleaseRingBuffer
  - GEVGetImageRingBuffer

2-3-14. GEVReleaseRingBuffer

- Function

  Decouple user managed ring buffer from the library.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].

- Description

  This function decouples the user ring buffer from the library.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVSetRingBuffer
  - GEVQueueRingBuffer

- GEVGetImageRingBuffer

2-3-15. GEVSetBufferCount

- Function

  Sets number of buffers in library managed ring buffer.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `count` : It specifies the buffer count.
- Description

  This function sets the number of buffers in the library managed ring buffer. Default setting is 4 buffers.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVGetBufferCount

2-3-16. GEVSetPacketResend

- Function

  Enables or disables packet resend feature.

- Syntax

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

- Parameters

- `cam_nr` : It specifies the GigE Vision device instance [1..50].
- `enable` : It sets the packet resend status (0 = disable, 1 = enable).

- Description

  This function enables or disables the packet resend feature.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVGetPacketResend

2-3-17. GEVSetPacketsOutOfOrder

- Function

  Sets the number of handled out of order packets.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `packets_out_of_order` : It sets the number of packets taken into account, before sending packet resend requests.

- Description

  This function is only effective if packet resend is enabled.

- Return

  ErrorCode (see "Error Codes")

- Reference

  - GEVGetPacketsOutOfOrder

2-3-18. GEVSetRingBuffer

- Function

  Couple user managed ring buffer with the library.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `index` : It specifies the index for the user ring buffer.
  - `buffer` : It specifies the user ring buffer to set in the library.
- Description

  This function sets user ring buffer in the library.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVReleaseRingBuffer
  - GEVQueueRingBuffer
  - GEVGetImageRingBuffer

2-3-19. GEVSetSecureTransfer (Not available)

- Function

  Enable secure transfer mode.

- Syntax

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

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `enable` : It enables or disables the secure transfer mode (0 = disable, 1 = enable).
  - `c_func` : It describes the callback function for secure transfer acknowledge.
- Description

  This function sets the secure transfer mode.
  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.
  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 KowaGigEVisionLib
        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

  ErrorCode (see "Error Codes")

- Reference

  - GEVGetSecureTransfer

2-4. TEST FUNCTIONS

2-4-1. GEVGetDllAbiVersion

- Function

  Get the binary interface version of running DLL.

- Syntax

```
void GEVGetDllAbiVersion(DWORD* major, DWORD* minor, DWORD*
micro);
```

- Parameters
  - `major`: buffer pointer of major version
  - `minor`: buffer pointer of minor version
  - `micro`: buffer pointer of micro version
- Description

  This function get binary interface version of running DLL. If the binary interface version equals or earlyer then SDK version at compiling time, your program can be running even if update DLL without recompiling.

  Useable version of compiling time are:

  - `GEV_SDK_VERSION_MAJOR`
  - `GEV_SDK_VERSION_MINOR`
  - `GEV_SDK_VERSION_MICRO`

  You can judge version compatible with the this code:

```
    DWORD abi_major, abi_minor, abi_micro;
    GEVGetDllAbiVersion(&abi_major, &abi_minor, &abi_micro);
    int required_abi =
        abi_major <  GEV_SDK_VERSION_MAJOR || (
        abi_major == GEV_SDK_VERSION_MAJOR && (
        abi_minor <  GEV_SDK_VERSION_MINOR || (
        abi_minor == GEV_SDK_VERSION_MINOR && (
        abi_micro <  GEV_SDK_VERSION_MICRO || (
        abi_micro == GEV_SDK_VERSION_MICRO && (
        1))))));
```

- Return

None

2-4-2. GEVGetDllVersion

- Function

  Get the version of running DLL.

- Syntax

```
void GEVGetDllVersion(DWORD* major, DWORD* minor, DWORD*
micro);
```

- Parameters
  - `major`: buffer pointer of major version
  - `minor`: buffer pointer of minor version
  - `micro`: buffer pointer of micro version
- Description

  This function get binary interface version of running DLL. You can judge to exists using functions if downgrade DLL.

  If you want to use version 1.4.2, you can judge version compatible with the this code:

```
    const DWORD requires_major = 1; // can change to
GEV_SDK_VERSION_MAJOR.
    const DWORD requires_minor = 4;
    const DWORD requires_micro = 2;
    DWORD dll_major, dll_minor, dll_micro;
    GEVGetDllVersion(&dll_major, &dll_minor, &dll_micro);
    int required_dll =
        dll_major >  requires_major || (
        dll_major == requires_major && (
        dll_minor >  requires_minor || (
        dll_minor == requires_minor && (
```

```
        dll_micro >  requires_micro || (
        dll_micro == requires_micro && (
        1))))));
```

- Return

  None

- Reference

  - GEVGetDllAbiVersion

2-4-3. GEVTestPacketResend

- Function

  Tests the packet resend function.

- Syntax

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

- Parameters

  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `on_off` : It switches the packet resend test on or off.
  - `packet_number` : It specifies the start of packets to resend.
  - `count` : It specifies the number of packets to resend.

- Description

  This function tests the packet resend by requesting one or more selectable packets. Packet Resend must be enabled with function `GEVSetPacketResend`. Also enable the DETAILED_LOG_INFO flag in the `GEVSetDetailedLog` function to see the results.

- Return

  ErrorCode (see "Error Codes")

2-4-4. 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);
```

- Parameters
  - `cam_nr` : It specifies the GigE Vision device instance [1..50].
  - `packet_size` : It returns the maximum found packet size.
  - `ps_min` : It specifies the minimum packetsize.
  - `ps_max` : It specifies the maximum packetsize.
  - `ps_inc` : It specifies the increment. `ps_min`, `ps_max` and `ps_inc` should be values read from XML.

- Description

  This function finds the maximum of packet size.

- Return

  ErrorCode (see "Error Codes")

- Reference
  - GEVTestPacket

2-5. ERROR CODES

Status Value	Value	Description
GEV_STATUS_SUCCESS	0x0000	Operation executed successfully
GigE Vision Status Codes	0x8001	See GigE Vision Specification

Status Value	Value	Description
	...	
	0x8FFF	
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)

Status Value	Value	Description
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	Invalid 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

Status Value	Value	Description
GEV_STATUS_XML_DLL_NOT_FOUND	0xC02F	Xml library libxml2.dll not found. Copy libxml2.dll to same directory as KowaGigEVisionLib 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
GEV_STATUS_GET_MANIFEST_TABLE_ERROR	0xC032	The acquisition of MANIFEST_TABLE fails
GEV_STATUS_DEPRECATED	0xC033	Deprecated processing
GEV_STATUS_BUFFERS_NOT_INIT	0xC034	The filterdriver isn't used and the image buffer has not been acquired
GEV_STATUS_INVALID_ARGUMENT	0xC036	Any arguments are invalid
GEV_STATUS_INVALID_OPERATION	0xC039	This operation not allow on current state
GEV_STATUS_UNDECIDED_PORT	0xC03a	Cannot decide port number

2-6. STRUCTURES

2-6-1. ENUMERATE_ADAPTER

Count:
Number of network interface adapter found.

Param:
Struct of found network interface adapter parameter (ADAPTER_PARAM)

2-6-2. ADAPTER_PARAM

AdapterIP:
IP address of network interface adapter.

AdapterMask:
Subnet mask of network interface adapter.

AdapterName:
Name of network interface adapter.

2-6-3. DISCOVERY

Count:
Number of devices found.

Param:
Struct of found device parameter (DEVICE_PARAM)

2-6-4. 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
```

2-6-5. 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)

2-6-6. 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

2-6-7. 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
```

2-6-8. 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:
Is this node implemented

IsAvailable:
Is this node available

IsLocked:
Is this node locked

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

```
#define DISPLAY_NOTATION_AUTOMATIC    0
#define DISPLAY_NOTATION_FIXED        1
#define DISPLAY_NOTATION_SCIENTIFIC   2
```

DisplayPrecision:
Precision of the display

InvalidatorCount:
Number of invalidators of the feature

PollingTime:
Polling time of the feature

2-6-9. 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

2-6-10. ACTION_KEYS

DeviceKey:
Action device key

GroupKey:
Action group key

GroupMask:
Action group mask