Kowa GigE SDK Manual (Library functions) Python Wrapper

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 functions by wrapped for python.

2. Kowa GigE SDK

This chapter describes the functions of the Kowa GigE SDK python-wrapped 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 python.
Parameter	Description of function parameters.
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.
Reference	List of other routines, which have a relationship to the current function. In addition refer to the GigE Vision specification.

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.

2-1. GENERAL FUNCTIONS

2-1-1. checkDeviceStatus

- Function

  Checks the status of a GigE Vision device.

- Syntax

```
checkDeviceStatus(ip_adapter: int, mask_adapter: int,
ip_device: int, ack_timeout: int, port: int) -> int
```

- 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.
  - `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

  none
- Reference
  - discovery
  - discoveryAdapter

2-1-2. close

- Function

  Closes GigE Vision device.
- Syntax

```
close(self)
```

- Parameters

  none
- Description

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

- Return

none

- Reference
    - init

2-1-3. closeFilterDriver

- Function

Close the Kowa device filter driver of GigE Vision device.

- Syntax

```
closeFilterDriver(self)
```

- Parameters

none

- Description

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

- Return

none

- Reference
    - initFilterDriver
    - getFilterDriverVersion

2-1-4. closeStreamChannel

- Function

Close GigE Vision device stream channel.

- Syntax

```
closeStreamChannel(self)
```

- Parameters

  none

- Description

  This function closes the stream channel of the specified device.

- Return

  none

- Reference
  - openStreamChannel

2-1-5. discovery

- Function

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

- Syntax

```
discovery(timeout_ms: int, ignore_subnet: bool, port: int) ->
DISCOVERY
```

- Parameters
  - `timeout_ms` : 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.
Devices returned from the callback did not have their status checked. This is can you do with the checkDeviceStatus function.

- Return
    - `DISCOVERY` : It returns number and parameters of the found devices (see Structures and KowaGigEVisionLib.h for the definition of struct DISCOVERY and DEVICE_PARAM).
- Reference
    - init
    - checkDeviceStatus

2-1-6. discoveryAdapter

- Function

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

- Syntax

```
discoveryAdapter(adapter: ADAPTER_PARAM, d_timeout: int,
ignore_subnet: bool, port: int) -> DISCOVERY
```

- Parameters
    - `adapter` : It specifies the network interface adapter (see Structures and KowaGigEVisionLib.h for the definition of struct ADAPTER_PARAM).
    - `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 function `discovery()`.

- Return
  - `DISCOVERY` : It returns number and parameters of the found devices (see Structures and KowaGigEVisionLib.h for the definition of struct DISCOVERY and DEVICE_PARAM).
- Reference
  - enumerateAdapters
  - checkDeviceStatus

2-1-7. enumerateAdapters

- Function

  Enumerate network interface adapter.

- Syntax

```
enumerateAdapters() -> ENUMERATE_ADAPTER
```

- Parameters

  none

- Description

  This function can enumerate up to 50 network interface adapter.

- Return
  - `ENUMERATE_ADAPTER` : It returns an enumeration of network interface adapters.
- Reference
  - discoveryAdapter

2-1-8. forceIp

- Function

  Force an IP address.

- Syntax

```
forceIp(new_ip: int, subnet: int, gateway: int, mac: int,
adapter_ip: int)
```

- Parameters
  - `new_ip` : It specifies IP-address.
  - `subnet` : It specifies subnet mask.
  - `gateway` : It specifies the gateway address.
  - `mac` : It specifies the MAC address of the device.
  - `adapter_ip` : It specifies the IP address of the network adapter where is connected the device. Network byte order for `new_ip`, `subnet` and `gateway` parameter.

- Description

  This function temporary modifies the network configuration of a device.

- Return

  none

- Reference
  - setNetConfig
  - getNetConfig

2-1-9. getChannelParameter

- Function

  Returns channel parameter.

- Syntax

```
getChannelParameter(self) -> CHANNEL_PARAMETER
```

- Parameters

  none

- Description

  This function returns the control and stream channel parameter.

- Return

  - `CHANNEL_PARAMETER` : Returns a structure of channel parameters.(see Structures and KowaGigEVisionLib.h for definition of CHANNEL_PARAMETER)

- Reference

  - setChannelParameter

2-1-10. getDetailedLog

- Function

  Gets detailed log output conditions.

- Syntax

```
getDetailedLog(self) -> int
```

- Parameters

  none

- Description

  This function returns the detailed log output conditions.

- Return

  The following is a combination of flags.

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

- Reference
    - setDetailedLog

2-1-11. getConnectionStatus

- Function

  Returns connection status.

- Syntax

```
getConnectionStatus(self) -> tuple[status: int, eval: int]
```

- Parameters

  none

- Description

  This function returns the connection status.

- Return
    - `status` : It is a containing the connection status. (0 = OK, 1 = timeout, 2 = access denied)
    - `eval` : It is a containing the evaluation status. (0 = OK, 1 = evaluation expired (deprecated), 2 = library is running in evaluation mode)

2-1-12. getFilterDriverVersion

- Function

Get filter driver version.

- Syntax

```
getFilterDriverVersion(self) -> tuple[version_major: int,
version_minor: int]
```

- Parameters

none

- Description

This function returns the filter driver version of the specified device.
If the filter driver is not activated the return value in `version_major` and `version_minor` is 0.

- Return
    - `version_major` : It returns the major version.
    - `version_minor` : It returns the minor version.
- Reference
    - initFilterDriver
    - closeFilterDriver

2-1-13. getHeartbeatRate

- Function

Returns heartbeat rate.

- Syntax

```
getHeartbeatRate(self) -> heartbeat_rate: int
```

- Parameters

none

- Description

  This function returns heartbeat rate in millisecond.

- Return

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

2-1-14. getNetConfig

- Function

  Get network configuration.

- Syntax

```
getNetConfig(self) -> tuple[dhcp: int, ip: int, subnet: int,
gateway: int]
```

- Parameters

  none

- Description

  This function reads the network configuration of the device.

- Return

  - `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.

- Reference

  - setNetConfig

2-1-15. getReadWriteParameter

- Function

  Returns read/write parameter.

- Syntax

```
getReadWriteParameter(self) -> tuple[ack_timeout: int,
retry_count: int]
```

- Parameters

  none

- Description

  This function returns read/write parameter for the `readRegister`, `writeRegister`, `readMemory` and `writeMemory` functions.

- Return
  - `ack_timeout` : It is a variable containing the acknowledge timeout in millisecond.
  - `retry_count` : It is a variable containing the retry count for the read/write commands.

- Reference
  - setReadWriteParameter

2-1-16. init

- Function

  Initialize a GigE Vision device.

- Syntax

```
init(camera_no: int, connection: CONNECTION, save_xml: int,
open_mode: int) -> _camera_info
```

- Parameters
    - `connection` : It specifies the connection data (see Structures and KowaGigEVisionLib.h for definition of CONNECTION) (CONNECTION is also acceptable in DEVICE_PARAM).
    - `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	value
OPEN_ACCESS	0
EXCLUSIVE_ACCESS	1
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
    - `_camera_info` : It specifies the internal class for GigE Vision device connection information.

- Reference
    - close
    - getLocalError

2-1-17. initEx

- Function

  Initialize a GigE Vision device.

- Syntax

```
initEx(connection: CONNECTION, save_xml: int, open_mode: int) -
> _camera_info
```

- Parameters
  - `connection`: It specifies the connection data (see Structures and KowaGigEVisionLib.h for definition of CONNECTION).
  - `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	value
OPEN_ACCESS	0
EXCLUSIVE_ACCESS	1
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
  - `_camera_info` : It specifies the internal class for GigE Vision device connection information.

- Reference
  - close
  - getLocalError

2-1-18. initFilterDriver

- Function

  Init the Kowa filter driver of GigE Vision device.

- Syntax

```
initFilterDriver(self)
```

- Parameters

none

- Description

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

- Return

  none

- Reference
  - closeFilterDriver
  - getFilterDriverVersion

2-1-19. issueActionCommandAdapter

- Function

  Issue an action command from the adapter.

- Syntax

```
issueActionCommandAdapter(adaptersip: typing.Iterable[str],
keys: ACTION_KEYS, actiontime:int, timeoutms:int, callback)
```

- Parameters
  - `adaptersip`: Specifies array of adapter IP addresses to issue the action command to.
  - `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.

- 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:

```
def callback(camera: None, fromip:str, error: GEVError|None) ->
int|None: ...
```

- `camera`: Always None.
- `fromip`: The IP address of the device that returned the action command response.
- `error`: The error of the action command response. If necessary, raise an exception with raise error. 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

  none

2-1-20. issueActionCommandAdapterDirected

- Function

  Issue an action command from the adapter.

- Syntax

```
issueActionCommandAdapterDirected(adaptersip:
typing.Iterable[str], keys: ACTION_KEYS, actiontime:int,
timeoutms:int, callback)
```

- Parameters
  - `adaptersip`: Specifies array of adapter IP addresses for issue the action command.
  - `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.
- Description

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

```
def callback(camera: None, fromip:str, error: GEVError|None) ->
int|None: ...
```

    - `camera`: Always None.
    - `fromip`: The IP address of the device that returned the action command response.
    - `error`: The error of the action command response. If necessary, raise an exception with raise error. 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

  none

2-1-21. issueActionCommandBroadcast

- Function

  Issue an action command to each device.

- Syntax

```
issueActionCommandAdapter(cameras: typing.Iterable[Camera],
keys: ACTION_KEYS, actiontime:int, timeoutms:int, callback)
```

- Parameters
    - `cameras`: Specifies array of devices for issue the action command.
    - `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.

- 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:

```
def callback(camera: Camera|None, fromip:str, error:
GEVError|None) -> int|None: ...
```

- `camera`: The device that returned the action command response if it is in cameras, otherwise None.
- `fromip`: The IP address of the device that returned the action command response.
- `error`: The error of the action command response. If necessary, raise an exception with raise error. 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

none

2-1-22. issueActionCommandBroadcastDirected

- Function

Issue an action command to each device.

- Syntax

```
issueActionCommandAdapterDirected(cameras:
typing.Iterable[Camera], keys: ACTION_KEYS, actiontime:int,
timeoutms:int, callback)
```

- Parameters
    - `cameras`: Specifies array of devices for issue the action command.
    - `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.
- Description

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

  ```
  def callback(camera: Camera|None, fromip:str, error:
  GEVError|None) -> int|None: ...
  ```

    - `camera`: The device that returned the action command response if it is in cameras, otherwise None.
    - `fromip`: The IP address of the device that returned the action command response.

- `error`: The error of the action command response. If necessary, raise an exception with raise error. 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

none

2-1-23. GEVIssueActionCommandUnicast

- Function

Issue an action command to a single device.

- Syntax

```
issueActionCommandUnicast(self, keys: ACTION_KEYS, actiontime:
int|None = None):
```

- Parameters
  - `keys`: Specifies device_key, group_mask, and group_key.
  - `actiontime`: Specifies the time to execute the action command. If 0 or None is specified, each device will execute the action command immediately.

- Description

This function issues an action command to a single device.

Like issueActionCommandBroadcast, 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

none

- Reference

- issueActionCommandBroadcast

2-1-24. openStreamChannel

- Function

Open GigE Vision device stream channel.

- Syntax

```
openStreamChannel(self, multicast: str = None, *, port: int =
0)
```

- Parameters
  - `multicast` : It specifies the multicast IP address of the stream channel. Defaults to disable multicast.
  - `port`: It specifies the port number of the stream channel. Defaults to auto select.

- Description

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

- Return

none

- Reference
  - closeStreamChannel

2-1-25. readRegister

- Function

Receive data from a GigE Vision device.

- Syntax

```
readRegister(cmd: int, cnt: int) -> values: int
```

- Parameters
    - `cmd`: It specifies the register address.
    - `cnt`: It specifies the number of 32bit reads.
- Description

  This function reads 32bit data from the specified GigE Vision device. This function implements the GEV Read_Reg command.
- Return
    - `values`: It returns the data read.
- Reference
    - setReadWriteParameter
    - writeRegister

2-1-26. readMemory

- Function

  Receive memory data from a GigE Vision device.
- Syntax

```
WORD readMemory(self, maddr: int, cnt: int) -> values: int
```

- Parameters
    - `maddr`: It specifies the memory address.
    - `cnt`: It specifies the number of bytes to read.
- Description

  This function reads a memory block from the specified GigE Vision device.
- Return
    - `values`: It returns the memory data read.

- Reference
    - writeMemory

2-1-27. setActionCommand

- Function

  Set action command.

- Syntax

```
setActionCommand(device_key: int, group_key: int, group_mask:
int, action_time: int)
```

- Parameters
    - `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

  none

2-1-28. setChannelParameter

- Function

  Set channel parameter.

- Syntax

```
setChannelParameter(cparam: CHANNEL_PARAMETER)
```

- Parameters
  - `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

  none
- Reference
  - getChannelParameter

2-1-29. setDetailedLog

- Function

  Sets detailed log.
- Syntax

```
setDetailedLog(flags: int)
```

- Parameters
  - `flags`: sets the log output option flag.

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

- Description

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

- Return

  none

- Reference

  - getDetailedLog

2-1-30. setHeartbeatRate

- Function

  Set heartbeat rate.

- Syntax

```
setHeartbeatRate(heartbeat_rate: int)
```

- Parameters

  - `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 this value. The interval value to check the connection state is set to half of the heartbeat timeout value. With the function `getChannelParameter` you can read the interval value for the connection check. (CHANNEL_PARAMETER cc_heartbeat_timeout.)

- Return

  none

2-1-31. setNetConfig

- Function

  Set network configuration.

- Syntax

```
setNetConfig(enable_dhcp: bool, ip: int, subnet: int, gateway: int)
```

- Parameters
    - `dhcp` : It specifies the dhcp enable status.
    - `ip` : It specifies IP-address.
    - `subnet` : It specifies the subnet mask.
    - `gateway` : It specifies 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

  none
- Reference
    - getNetConfig

2-1-32. setReadWriteParameter

- Function

  Sets read/write parameter.
- Syntax

```
setReadWriteParameter(ack_timeout: int, retry_count: int)
```

- Parameters
    - `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 `readRegister`, `writeRegister`, `readMemory` and `writeMemory` funtions.

- Return

  none

- Reference

  - getReadWriteParameter

2-1-33. testPacket

- Function

  Trigger GigE Vision test packet.

- Syntax

```
testPacket(self) -> packet_size: int
```

- Parameters

  none

- Description

  This function triggers the specified GigE Vision device 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

  - `packet_size` : It returns the packet size of the test packet.

- Reference

  - testFindMaxPacketSize

2-1-34. setTraversingFirewallsInterval

- Function

  Set the interval in seconds for traversing firewalls.

- Syntax

```
setTraversingFirewallsInterval(self, interval: int)
```

- Parameters
  - `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

  none

- Reference
  - openStreamChannel

2-1-35. writeMemory

- Function

  Send memory data to a GigE Vision device.

- Syntax

```
writeMemory(self, maddr: int, values: bytes)
```

- Parameters

- `maddr`: It specifies the memory address.
- `values`: It specifies the byte data to write.

- Description

  This function writes a memory block to the specified GigE Vision device.

- Return

  none

- Reference
  - readMemory

2-1-36. writeRegister

- Function

  Send register data to GigE Vision device.

- Syntax

```
writeRegister(self, cmd: int, values: Iterable[int])
```

- Parameters
  - `cmd`: It specifies the register address to be written to.
  - `values`: It specifies the data array to be written.

- Description

  This function writes 32bit data to the specified GigE Vision device.

- Return

  none

- Reference
  - readRegister

2-2. XML-FUNCTIONS

2-2-1. executeFeatureCommand

- Function

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

- Syntax

```
executeFeatureCommand(self, feature_name: str)
```

- Parameters
  - `feature_name`: It specifies feature name (e.g. AcquisitionStart)

- Description

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

- Return

  none

2-2-2. getFeatureBoolean

- Function

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

- Syntax

```
getFeatureBoolean(self, feature_name: str) -> bool_value: bool
```

- Parameters
  - `feature_name` : It specifies the feature name (e.g. LUTEnable)

- Description

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

- Return
    - `bool_value` : It returns the Boolean value.
- Reference
    - setFeatureBoolean
    - getFeatureParameter

2-2-3. getFeatureCommand (Deprecated)

- Function

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

- Syntax

```
getFeatureCommand(self, feature_name: str) -> cmd_value: int
```

- Parameters
    - `feature_name` : It specifies feature node name (e.g. AcquisitionStart).
- Description

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

- Return
    - `cmd_value` : It returns the command value.
- Reference
    - setFeatureCommand
    - getFeatureParameter

2-2-4. getFeatureDisplayName

- Function

  Returns the display name of a dedicated feature, descripted in xml file.

- Syntax

```
getFeatureDisplayName(self, feature_name: str) -> display_name:
str
```

- Parameters
    - `feature_name` : It specifies feature name (e.g. Width)
- Description

  This function returns the display name of a feature descripted in xml-file.
- Return
    - `display_name` : It returns the display name.
- Reference
    - getFeatureTooltip

2-2-5. getFeatureEnableStatus

- Function

  Returns the enable status of a dedicated feature, descripted in xml file.
- Syntax

```
getFeatureEnableStatus(self, feature_name: str) -> enable: bool
```

- Parameters
    - `feature_name` : It specifies feature name (e.g. Width)
- Description

  This function returns the access status of a feature descripted in xml-file.
- Return
    - `enable` : It returns the status.

2-2-6. getFeatureEnumeration

- Function

  Returns the enumeration name of a dedicated feature, descripted in xml file.

- Syntax

```
getFeatureEnumeration(self, feature_name: str) -> enum_name:
str
```

- Parameters
  - `feature_name` : It specifies feature name (e.g. Pixelformat)

- Description

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

- Return
  - `enum_name` : It returns feature enumeration value.

- Reference
  - setFeatureEnumeration
  - getFeatureEnumerationName
  - getFeatureParameter

2-2-7. getFeatureEnumerationName

- Function

  Returns the name of an enumeration value.

- Syntax

```
getFeatureEnumerationName(self, feature_name: str, enum_index:
int) -> enum_name: str
```

- Parameters
    - `feature_name` : It specifies feature name (e.g. Pixelformat)
    - `enum_index` : It specifies enumeration index.
- 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 `getFeatureParameter`.
- Return
    - `enum_name` : It returns feature name of enumeration name. (e.g. Mono8)
- Reference
    - setFeatureEnumeration
    - getFeatureEnumeration
    - getFeatureParameter

2-2-8. getFeatureFloat

- Function

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

```
getFeatureFloat(self, feature_name: str) -> float_value: float
```

- Parameters
    - `feature_name` : It specifies the feature name (e.g. DeviceTemperature)
- Description

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

  - `float_value` : It returns the float value.

- Reference
  - setFeatureFloat

2-2-9. getFeatureInteger

- Function

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

- Syntax

```
getFeatureInteger(self, feature_name: str) -> int_value: int
```

- Parameters
  - `feature_name` : It specifies the feature name (e.g. Width)

- Description

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

- Return
  - `int_value` : It returns feature integer value.

- Reference
  - setFeatureInteger
  - getFeatureParameter

2-2-10. getFeatureInvalidator

- Function

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

- Syntax

```
getFeatureInvalidator(self, feature_name: str, index: int) ->
invalidator_name: str
```

- Parameters
    - `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 `getFeatureParameter`.
- Description

  This function returns the invalidator of a feature descripted in xml-file.
- Return
    - `invalidator_name` : It returns invalidator name.
- Reference
    - getFeatureParameter

2-2-11. getFeatureList

- Function

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

```
getFeatureList(self) -> tuple[featureList: FeatureList,
maxLevel: int]
```

- Parameters
    - `featureList` : It returns the feature list.
- Description

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

- Return
    - `maxLevel` : It returns the maximal level to list.

2-2-12. getFeatureParameter

- Function

  Returns properties of a dedicated feature, descripted in xml file.

- Syntax

```
getFeatureParameter(self, feature_name: str) -> f_param:
FEATURE_PARAMETER
```

- Parameters
    - `feature_name` : It specifies feature name (e.g. Width)
- Description

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

- Return
    - `f_param` : It returns the parameters of the given feature (see Structures and KowaGigEVisionLib.h for definition of struct FEATURE_PARAMETER).

2-2-13. getFeaturePort

- Function

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

- Syntax

```
getFeaturePort(self, feature_name: str) -> port_name: str
```

- Parameters
    - `feature_name` : It specifies feature name (e.g. Width)

- Description

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

- Return

  - `port_name` : It returns port name.

2-2-14. getFeatureRegister

- Function

  Returns the values of a dedicated register feature, descripted in xml file.

- Syntax

```
getFeatureRegister(self, feature_name: str, length: int) ->
buffer: int
```

- Parameters

  - `feature_name` : It specifies feature name (e.g. LUTValueAll)
  - `length` : It specifies length of buffer/register.

- Description

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

- Return

  - `buffer` :It returns buffer values.

- Reference

  - setFeatureRegister
  - getFeatureParameter

2-2-15. getFeatureString

- Function

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

- Syntax

```
getFeatureString(self, feature_name: str) -> str_value: str
```

- Parameters
    - `feature_name` : It specifies feature name (e.g. DeviceVersion)
- Description

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

- Return
    - `str_value` : It returns feature string value.
- Reference
    - setFeatureString
    - getFeatureParameter

2-2-16. getFeatureTooltip

- Function

Returns the tooltip string of a dedicated feature, descripted in xml file.

- Syntax

```
getFeatureTooltip(self, feature_name: str) -> tooltip_name: str
```

- Parameters
    - `feature_name` : It specifies feature name (e.g. Width)
- Description

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

- Return
    - `tooltip_name` : It returns tooltip string.

- Reference
  - getFeatureDisplayName

2-2-17. getFeatureUnit

- Function

  Returns unit string of a dedicated feature, descripted in xml file.

- Syntax

```
getFeatureUnit(self, feature_name: str) -> unit_name: str
```

- Parameters
  - `feature_name` : It specifies feature name (e.g. DeviceTemperature)
- Description

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

- Return
  - `unit_name` : It returns unit string (e.g. C).

2-2-18. getXmlFile

- Function

  Get the xml file data.

- Syntax

```
getXmlFile(self) -> buf: bytes
```

- Parameters

  none

- Description

This function returns the xml-file of the device.

- Return

Contens of device xml file

- Reference
    - getXmlSize

2-2-19. getXmlSize

- Function

Get the size of the device xml file.

- Syntax

```
getXmlSize(self) -> size: int
```

- Parameters

none

- Description

This function returns the size of the xml-file of the device.

- Return
    - `size`: It returns the XML file size.
- Reference
    - getXmlFile

2-2-20. initXml

- Function

Initialize xml.

- Syntax

```
initXml(self)
```

- Parameters

  none

- 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

  none

- Reference

  - setXmlFile
  - setSchemaPath

2-2-21. setFeatureBoolean

- Function

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

- Syntax

```
setFeatureBoolean(self, feature_name: str, bool_value: bool)
```

- Parameters

  - `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

none

- Reference
    - getFeatureBoolean
    - getFeatureParameter

2-2-22. setFeatureCommand (Deprecated)

- Function

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

- Syntax

```
setFeatureCommand(self, feature_name: str, cmd_value: int)
```

- Parameters
    - `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

  none

- Reference
    - getFeatureCommand
    - getFeatureParameter

2-2-23. setFeatureEnumeration

- Function

  Set enumeration name of a dedicated feature, descripted in xml file.

- Syntax

```
setFeatureEnumeration(self, feature_name: str, enum_name: str)
```

- Parameters
  - `feature_name` : It specifies feature name (e.g. Pixelformat)
  - `enum_name` : It specifies the feature's enumeration name (e.g. Mono8).
- Description

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

  none
- Reference
  - getFeatureEnumeration
  - getFeatureEnumerationName
  - getFeatureParameter

2-2-24. setFeatureFloat

- Function

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

```
setFeatureFloat(self, feature_name: str, float_value: float)
```

- Parameters
  - `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

  none

- Reference

  - getFeatureFloat

2-2-25. setFeatureInteger

- Function

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

- Syntax

```
setFeatureInteger(self, feature_name: str, int_value: int)
```

- Parameters

  - `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

  none

- Reference

  - getFeatureInteger
  - getFeatureParameter

2-2-26. setFeatureRegister

- Function

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

- Syntax

```
setFeatureRegister(self, feature_name: str, buffer: bytes)
```

- Parameters
  - `feature_name` : It specifies feature name (e.g. LUTValueAll)
  - `buffer` : It specifies the new buffer values.
- Description

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

  none
- Reference
  - getFeatureRegister
  - getFeatureParameter

2-2-27. setFeatureString

- Function

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

```
setFeatureString(self, feature_name: str, str_value: str)
```

- Parameters
  - `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

none

- Reference
    - getFeatureInteger
    - getFeatureParameter

2-2-28. setSchemaPath

- Function

Set the path of the schema files.

- Syntax

```
setSchemaPath(self, schema_path: str)
```

- Parameters
    - `schema_path` : It specifies the new path of the schema files.

- Description

This function sets the new path of the schema files.
Default path of the schema files are current program/library path.
This function must be called after init function and before the initXml function.
This function is used exclusively with function `setSchemaFromZip`.

- Return

none

- Reference
    - initXml
    - setSchemaFromZip

2-2-29. setSchemaFromZip

- Function

Set the the schema file as a zip file.

- Syntax

```
setSchemaFromZip(self, schema_zip: bytes)
```

- Parameters
  - `schema_zip` : It specifies the pointer of zip binary data.

- Description

  This function sets the schema data from a zip file data.
  This function must be called after init function and before the initXml function.
  This function is used exclusively with function `setSchemaPath`.
  Default schema data are files in current program/library path.

- Return

  none

- Reference
  - initXml
  - setSchemaPath

2-2-30. setXmlFile

- Function

  Set xml file in the library.

- Syntax

```
setXmlFile(self, xml_name: str)
```

- Parameters
  - `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

  none

- Reference

  - initXml

2-3. CAMERA FUNCTIONS

2-3-1. acquisitionStart

- Function

  Start image acquisition.

- Syntax

```
acquisitionStart(self, number_images_to_acquire: int)
```

- Parameters

  - `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, `acquisitionStop` is called automatically.

AcquisitionMode	**Number_images_to_acquire**
Continuous	0 (unlimited)
SingleFrame	1
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 `getImageBuffer`.

  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

none

• Reference

- acquisitionStop
- getImageBuffer

2-3-2. acquisitionStartEx

• Function

Start image acquisition without library internal xml handling.

• Syntax

```
acquisitionStartEx(self, number_images_to_acquire: int,
image_size: int, image_width: int, image_height: int,
pixel_format: int)
```

• Parameters

- `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 `acquisitionStop` 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.

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

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

  Get the value from the AcquisitionFrameCount node

- Return

  none

- Reference
  - acquisitionStart
  - acquisitionStop
  - getImageBuffer

2-3-3. acquisitionStop

- Function

  Stop image acquisition.

- Syntax

```
acquisitionStop(self)
```

- Parameters

  none

- Description

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

- Return

  none

- Reference
  - acquisitionStart
  - getImageBuffer

2-3-4. getBufferCount

- Function

  Returns number of buffers of library managed ring buffer.

- Syntax

```
getBufferCount(self) -> count: int
```

- Parameters

  none

- Description

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

- Return
  - `count` : It returns the number of buffers of the image ring buffer.

- Reference
  - setBufferCount

2-3-5. getImageBuffer

- Function

Get an image from the library managed ring buffer and copy it to user memory.

- Syntax

```
getImageBuffer(self, buffer: ctypes.c_uint8 * N) ->
image_header: IMAGE_HEADER
```

- Parameters
    - `buffer` : Data buffer of acquired images. This buffer is type `ctypes.c_uint8` array and the buffer size can be obtained by `camera.getFeatureInteger("PayloadSize")`.

- Description

```
img_size = camera.getFeatureInteger("PayloadSize")
image_buffer = (ctypes.c_uint8*img_size)()
...
img_header = camera.getImageBuffer(image_buffer)
```

- Return
    - `image_header` : It returns information of the image (see Structures and KowaGigEVisionLib.h for definition of IMAGE_HEADER).

- Reference
    - acquisitionStart
    - acquisitionStop

2-3-6. getImageRingBuffer

- Function

Get an image from user managed ring buffer and return buffer index.

- Syntax

```
getImageRingBuffer(self) -> tuple[image_header: IMAGE_HEADER,
image_buffer_index: int]
```

- Parameters

  none

- 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
  - `image_header` : It returns information of the image (see Structures and KowaGigEVisionLib.h for definition of IMAGE_HEADER).
  - `image_buffer_index` : It returns the destination buffer number.

- Reference
  - acquisitionStart
  - acquisitionStop
  - setRingBuffer
  - queueRingBuffer
  - releaseRingBuffer

2-3-7. getImageFPS

- Function

  Returns the number of the acquired frames per second.

- Syntax

```
getImageFPS(self) -> fps: float
```

- Parameters

none

- Description

This function returns the number of acquired frames per second.

- Return
  - `fps` : It returns the frames per second.

2-3-8. getPacketResend

- Function

Gets packet resend activation status.

- Syntax

```
getPacketResend(self) -> enable: bool
```

- Parameters

none

- Description

This function gets packet resend activation status.

- Return
  - `enable` : It returns the packet resend activation status.
- Reference
  - setPacketResend

2-3-9. getPacketsOutOfOrder

- Function

Returns the number of handled packets out of order.

- Syntax

```
getPacketsOutOfOrder(self) -> packets_out_of_order: int
```

- Parameters

  none

- Description

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

- Return

  - `packets_out_of_order` : It returns the number of packets taken into account, before sending packet resend requests.

- Reference

  - setPacketsOutOfOrder

2-3-10. packetResend

- Function

  Send packet resend command.

- Syntax

```
packetResend(self, stream_channel: int, block_id: int,
first_packet_id: int, last_packet_id:int)
```

- Parameters

  - `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.

- Return

  none

- Reference

  - setPacketResend
  - getPacketResend

2-3-11. queueRingBuffer

- Function

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

- Syntax

```
queueRingBuffer(self, image_buffer_index: int)
```

- Parameters

  - `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 `getImageRingBuffer`.
  This function releases this user ring buffer part with index `image_buffer_index` for acquisition of new data.

- Return

  none

- Reference

  - setRingBuffer
  - releaseRingBuffer

  - getImageRingBuffer

2-3-12. releaseRingBuffer

- Function

  Decouple user managed ring buffer from the library.

- Syntax

```
releaseRingBuffer(self)
```

- Parameters

  none

- Description

  This function decouples the user ring buffer from the library.

- Return

  none

- Reference

  - setRingBuffer
  - queueRingBuffer
  - getImageRingBuffer

2-3-13. setBufferCount

- Function

  Sets number of buffers in library managed ring buffer.

- Syntax

```
setBufferCount(self, count: int)
```

- Parameters

  - `count` : It specifies the number of buffers.
- Description

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

  none
- Reference
  - getBufferCount

2-3-14. setPacketResend

- Function

  Enables or disables packet resend feature.
- Syntax

```
setPacketResend(self, enable: bool)
```

- Parameters
  - `enable` : It specifies the packet resend status.
- Description

  This function enables or disables the packet resend feature.
- Return

  none
- Reference
  - getPacketResend

2-3-15. setPacketsOutOfOrder

- Function

Sets the number of handled out of order packets.

- Syntax

```
setPacketsOutOfOrder(self, packets_out_of_order: int)
```

- Parameters
    - `packets_out_of_order` : It specifies the number of packets taken into account, before sending packet resend requests.
- Description

This function is only effective if packet resend is enabled.

- Return

none

- Reference
    - getPacketsOutOfOrder

2-3-16. setRingBuffer

- Function

Couple user managed ring buffer with the library.

- Syntax

```
setRingBuffer(self, index: int, buffer: int)
```

- Parameters
    - `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

  none

- Reference
  - releaseRingBuffer
  - queueRingBuffer
  - getImageRingBuffer

2-4. TEST FUNCTIONS

2-4-1. getDllAbiVersion

- Function

  Get the binary interface version of running DLL.

- Syntax

```
getDllAbiVersion() -> tuple[major: int, minor: int, micro: int]
```

- Parameters

  none

- Description

  This function get binary interface version of running DLL.

- Return

  version by tuple. e.g. `(1, 4, 2)`

2-4-2. getDllVersion

- Function

  Get the version of running DLL.

- Syntax

```
getDllVersion() -> tuple[major: int, minor: int, micro: int]
```

- Parameters

  none

- 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:

```
assert kge.getDllVersion() >= (1, 4, 2)
```

- Return

  version by tuple. e.g. `(1, 4, 2)`

- Reference
  - getDllAbiVersion

2-4-3. testPacketResend

- Function

  Tests the packet resend function.

- Syntax

```
testPacketResend(self, on_off: bool, packet_number: int, count: int)
```

- Parameters
  - `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 `setPacketResend`. Also enable the DETAILED_LOG_INFO flag in the `setDetailedLog` function to see the results.

- Return

  none

2-4-4. testFindMaxPacketSize

- Function

  Find the maximum of packet size.

- Syntax

```
testFindMaxPacketSize(self, size_min: int, size_max: int,
size_step: int) -> packet_size: int
```

- Parameters

  - `size_min` : It returns the minimum packetsize.
  - `size_max` : It returns the maximum packetsize.
  - `size_step` : It returns the increment. `size_min`, `size_max` and `size_step` shall be read from the xml and passed to this function.

- Description

  This function finds the maximum of packet size.

- Return

  - `packet_size` : It returns the maximum found packet size.

- Reference

  - testPacket

2-5. ERROR CODES

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

Status Value	Value	Description
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	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

Status Value	Value	Description
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 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	value
DISCOVERY_STATUS_OK	0
DISCOVERY_STATUS_ALLREADY_OPEN	1
DISCOVERY_STATUS_NOT_SAME_SUBNET	2
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	value
TYPE_CATEGORY	0
TYPE_FEATURE	1
TYPE_INTEGER	2
TYPE_FLOAT	3
TYPE_STRING	4
TYPE_ENUMERATION	5
TYPE_COMMAND	6
TYPE_BOOLEAN	7
TYPE_REGISTER	8
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	value
TYPE_CATEGORY	0
TYPE_FEATURE	1
TYPE_INTEGER	2
TYPE_FLOAT	3
TYPE_STRING	4
TYPE_ENUMERATION	5
TYPE_COMMAND	6
TYPE_BOOLEAN	7
TYPE_REGISTER	8
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	value	description
ACCESS_MODE_RO	0x524F	read only

define	value	description
ACCESS_MODE_RW	0x5257	read/write
ACCESS_MODE_WO	0x574F	write only

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

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

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	value
VISIBILITY_INVISIBLE	0
VISIBILITY_BEGINNER	1
VISIBILITY_EXPERT	2
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	value
SIGN_UNSIGNED	0
SIGN_SIGNED	1

Address:
Address of the feature

DisplayNotation:
Notation of the display

define	value
DISPLAY_NOTATION_AUTOMATIC	0
DISPLAY_NOTATION_FIXED	1
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