NITLibrary Python

REFERENCE GUIDE

Contents

1. Introduction

This documentation presents all the existing classes included in the version 3.2.0 of the library.

If you know the version 2.x.x of the NITLibrary, we invite you to begin with:

- Breaking changes since the 2.x.x series
- Porting Guide
  This pages lists the new features of the library.

Every one should read:

- The NITLibrary concepts Who explain the NITLibrary main concepts
- Memory model of the library For a description of the memory usage
- Threading strategy For an in-depth explanation of the threading strategy of the library
- Error reporting To have an overview of the error reporting mecanism.
- Supported platforms
- Installation

If you use SWIR cameras, you can read:

- NUC and BPR

You can read the pages specific to your camera to know the available parameters:

- MC-1003 parameters
- WiDySWIR 640V-S series parameters
- WiDySWIR S320V-S parameters
- WiDySWIR 320V-S series parameters
- WiDySenS 640V-S series parameters
- (Hipe) SenS 640V-S series parameters
- SenS 1280V-S series parameters
- WiDySenS 320V-S series parameters
- MC1003-1GB series parameters
- MC1003-1Gx color series parameters
- WiDySWIR 640G-SE series parameters
- WiDySenS 640G-STE series parameters

You should also read the sample codes pages for a hands-on NITLIbrary:

- Steps to build a program (Sample Codes)

Finally the faq page respond to some recurring questions:

- ref faq(not yet implemented)
- Release Notes
- Document Revision

2. NITLibrary: Breaking changes since the 2.x.x series

Version 3 is a complete rewrite of the NITLibrary taking into account the remarks of our customers.

Global changes:

The python library is based on the C++ NITLibrary 64 bit based computers.
It no longer is dependant from external libraries like OpenCV or the microsoft runtimes.
Different versions are available to run with different Python versions.

Supported cameras:

The USB2 cameras have been removed from the library. We only support USB3 cameras.
Actually the following cameras can be processed with an USB firmware version above or equal 2.0.0:

- WiDySWIR 320V-S
- WiDySWIR 320V-SP
- WiDySWIR S320V-S
- WiDySWIR 640V-S
- WiDySWIR 640V-SP
- WiDySWIR 640V-ST
- WiDySWIR 640V-STP
- WiDySenS 640V-ST
- WiDySenS 640V-STP
- WiDySenS 320V-ST
- SenS 640V-STP
- Hipe SenS 640V-STP
- SenS 1280V-ST
- MC1003-1VF
- MC1003-1VC
- MC1003-1VB

We continue the support of the 2.x.x series of NITLibrary for some time for the cameras not supported by the new SDK

Changes relative to parameter setting:

The functions setParamValue to set parameters have been overloaded to support double, unsigned int and string parameter values.
The parameter names can now be set in different forms. Uppercase letters and spaces are no more mandatory.
The parameter name "Exposure Time" can be passed as "Exposure Time", "ExposureTime", "exposure time" "exposuretime" or any combinaison of these letters.

Some parameters are interdependant. If for example, you change the exposure time, the frame rate range is impacted. Also if you change the capture mode, exposure range can be modified.

On the previous NITLibrary the application that used the library was not aware of those changes.

The class NITConfigObserver if instantiated and connected to the NITDevice acts as a callback who is called when a parameter is changed.
New functions have been added to this class to detect a range change.
The functions relative to parameter change are now:

- NITConfigObserver.onParamChanged(paramName, paramValue as string, param value as float) who is called when a parameter is changed (by the user or internally)

- NITConfigObserver.onParamRangeChanged(paramName, array of paramValues as string, array of param value as float, current param value as string, current param value as float) who is called when a parameter rang changes.

- NITConfigObserver.onFpsChanged(frame rate value as float) who is called when the frame rate is changed (by the user or internally)

- NITConfigObserver.onFpsRangeChanged(min frame rate value as float , max frame rate value as float , current frame rate value as float) who is called when the frame rate range is changed.

  As a consequence NITConfigObserver become a master class of the NITLibrary, it is a good idea to derive from this class and overload the functions.

Changes relative to the NITDevice object:

Before passing the captured frames to the filtering pipeline, NITDevice makes some preprocessing:

- For all cameras, NUC and BPR are automatically applied if the directory containing the NUC files are at the appropriate place (that is in the working directory of the application). Furthermore, for exposure times where no NUC file exists, the NUC is interpolated.
- For Color cameras, the demosaicing is done before the frames are passed to the pipeline. This behavior can be disabled if wanted.

  The captured frames, once preprocessed are outputted as float in the range [0.0-1.0] for grayscale output. For color cameras, if the demosaicing is applied, the output is RGBA.

Changes relative to the NITFilter and NITObserver objects:

All the NITFilter classes can now be modified on the fly while connected to a running pipeline.
For example, you can change the text of the NITAddText filter while the streaming is active. The same is true for NITFlip, you can change the flip (vertical, horizontal, both) on the fly.
There is no more need to disconnect the device and instantiate a new class (has it was the case with the old NITLibrary).
More on this in the pages relative to each class.

3. NITLibrary: Porting Guide

NITLibrary 2.x.x	NITLibrary 3.x.x
Initializing NITManager to use Gige cameras - Applies to Gige cameras	
NITManager.useGige = true NITManager.useUsb = false NITManager.gigeDiscoveryTimeInSeconds = 60 NITManager.numberGigeDevicesToDiscover = 1	NITManager.use(GIGE, 1, 60)
Setting parameters	
Send one parameter immediatly: dev.setParamValueOf(<paramName>, <paramValue>, true) Delayed send multiple parameters: dev.setParamValueOf(<paramName1>, <paramValue1>, false) dev.setParamValueOf(<paramName2>, <paramValue2>, false) dev.setParamValueOf(<paramName3>, <paramValue3>, false) ... dev.updateConfig()	Send one parameter immediatly: dev.setParamValueOf(<paramName>, <paramValue>).updateConfig() Delayed send multiple parameters: dev.setParamValueOf(<paramName1>, <paramValue1>) dev.setParamValueOf(<paramName2>, <paramValue2>) dev.setParamValueOf(<paramName3>, <paramValue3>) ... dev.updateConfig()
Setting fps	
Send immediatly: dev.setFps(50, true)	Send immediatly: dev.setFps(50) dev.updateConfig()

NITLibrary 2.x.x	NITLibrary 3.x.x
Delayed send: dev.setFps(50, false) ... dev.updateConfig()	Delayed send: dev.setFps(50) ... dev.updateConfig()
NUC - Applies mostly to Usb cameras	
You must instantiate an object of one of the Nuc classes eg: NITNuc2Points nuc(filename, width, height, colStart, lineStart) or NITInterpolatedNuc nuc(dev, path_to_SNdirectory) and connect this object to NITDevice dev << nuc << ... To change Nuc with NITNuc2Points, you have to: -Stop the streaming -Disconnect nuc -Create a new nuc object -Connect new nuc -Reconnect all filters after nuc Changing Nuc with NITInterpolatedNuc is automatic when a parameter is changed To disable Nuc, you have to: -Stop the streaming -Disconnect nuc -Reconnect all filters after nuc	Nuc processing is embedded in the NITDevice. At construction of NITDevice, if a directory called SNxxx(with xxx serial number of the camera) is found in the executable working directory(Windows), /var/lib/NIT/SN (Linux) the Nucs are loaded and managed automatically. Changing Nuc is automatic when a parameter is changed To disable Nuc you have to: -call dev.activateNuc(false)

NITLibrary 2.x.x	NITLibrary 3.x.x
BPR - Applies mostly to Usb cameras	
You must instantiate an object of class NITBpr eg: NITBpr bpr(filename, width, height, colStart, lineStart) and connect this object to the pipeline dev << nuc << bpr << ... To change Bpr you have to: -Stop the streaming -Disconnect bpr -Create a new bpr object -Connect new bpr -Reconnect all filters after bpr To disable Bpr, you have to: -Stop the streaming -Disconnect bpr -Reconnect all filters after bpr	Bpr processing is embedded in the NITDevice. At construction of NITDevice, if a directory called SNxxx(with xxx serial number of the camera) is found in the executable working directory(Windows),),/var/lib/NIT/SN (Linux) the Bprs are loaded and managed automatically. Changing Bpr is automatic when a parameter is changed To disable Bpr, you have to: -call dev.activateBpr(false)
Debayering - Applies to MC1003 USB cameras only	
You must instantiate an object of classe NITDemosaicing eg: NITDemosaicing demo(width, height, colorSaturation) and connect this object to the pipeline dev << demo << ... To change Debayering parameter, you have to: -Stop the streaming -Disconnect demo	Debayering is embedded in the NITDevice. At construction of NITDevice, if the camera is a color camera debayering is activated Debayering parameters can be changed on the fly without stopping the streaming

NITLibrary 2.x.x	NITLibrary 3.x.x
-Create a new Demosaicing object -Connect new demo -Reconnect all filters after demo To disable Debayering, you have to: -Stop the streaming -Disconnect demo -Reconnect all filters after demo	To disable Debayering, you have to: -call Demosaicing.activate(dev, false)

4. The NITLibrary concepts

NITLibrary provide an interface to use NIT camera devices.

All the classes and functions of the library live in the namespace **NITLibrary**.
The namespace **NITLibrary** contain the following sub namespaces:

- NITToolBox which embeds the filters and observers.
- Demosaicing which deals only with color cameras. Each class function valid for all the cameras are accessible through the main namespace **NITLibrary**. Classes and functions who work only on some cameras are accessible thrue sub namespaces like Demosaicing.

The library main functions are:

- Device discovery and management
- Setting Device parameters
- Frame capture
- Frame processing

4.1. Device discovery and management:

When an USB device is connected to a host, the operating system is alerted and collects informations about this new device.
This informations include the VendorId(VID) and ProductId(PID) of the USB device and if the device is connected to a USB-2 or USB-3 port.
During this phase, the operating system also negociates the speed of the communication with the device.
In **NITLibrary**, the class who is responsible of the device discovery is the class NITManager.
An object of this class must be instantiated once and live for the livetime of the application. The NITManager class is a singleton.
Once instantiated, the following actions are performed by the NITManager:

- NITManager queries the operating system for the currently connected devices.
    - From this list, it removes all the devices that have not the requested VID/PID pairing. From now on the list contains uniquely NIT devices.

- For each element of the list, NITManager will open the device and send a couple of parameters to set the device in discovery mode. In this mode the camera doesnt stream frames but discovery informations like camera model, serial number, firmware version.
- Once all the NIT devices have been discovered, NITManager have a list of valid NIT devices.
- An application can now call NITManager and ask for a particular device.

This process is always the same, so the beginning of each source code dealing with NIT cameras is the following:

```
NITLibrary.NITManager manager = NITLibrary.NITManager.getInstance() #The first call to this
function create an object NITManager who live for the duration of the application.

nbre_devices = manager.deviceCount() #Query the number of NIT devices available

if( nbre_devices == 0 ) then return #If no device is present we cannot go further

descriptor = manager.getDescriptor(0) #Query a string describing the first device of the list.
#This string is the same you can see in the discovery dialog of NITVision.

NITLibrary.NITDevice dev = manager.openDevice( 0 ); #Query NITManager to open the first device
of #the list.

#At this point the NITDevice is usable.
....
```

4.2. Setting Device parameters:

At power up, the device has default parameters.
To change these values we have to call a couple of function of NITDevice.
Each camera has a set of parameters(see the camera pages).
Each parameter have a name (like "Exposure Time").
Parameter values can be enumerations (like "Mode") or discrete numeric values in a range (like "Number of Column").
The class NITDevice has a family of functions to set or get the parameters as string values or as numeric values.

```
Example: dev.setParamValueOf( "Number of Line", 480 ) #Set rows with a numeric value

dev.setParamValueOf( "Number of Line", "480" ) #Set rows with a string value

rows = dev.paramValueOf( "NumberOfLine" ) #Get the rows as numeric value

rows = dev.paramStrValueOf( "NumberOfLine" ) #Get the rows as string value
```

For enumeration values, the numeric forms of these functions require the index of the enumeration values.

For all the parameters, the value passed must be an exact value, apart for "Exposure Time" and frame rate.
These two parameters are considered as continuous values; the param functions round the passed values to the nearest available value.
Example: if you call dev.setParamValueOf("Exposure Time", 90); the parameter is set to 100 microseconds (if 90 microseconds doesn't exist for this camera).
Regarding the frame rate, NITDevice has the dedicated functions setFps() and fps().

To continue our preceding example, once the camera NITDevice is instantiated

```
... preceding code
#We now set the parameters of the camera
#We assume the default resolution of the camera is OK
dev.setParamValueOf( "Mode", "Global Shutter" ).setParamValueOf( "Exposure Time", 20000
).updateConfig()
dev.setFps( 68 ) #frame rate in frame per second
dev.updateConfig() #To take into account the new frame rate

# This preceding lines could also have been written this way
dev.setParamValueOf( "Mode", "Global Shutter" )
dev.setParamValueOf( "Exposure Time", 20000 ) #Expo time in microseconds
dev.updateConfig()

dev.setFps( 68 ) #frame rate in frame per second
dev.updateConfig()

#We are almost ready to capture frames
....
```

In the code snippet above you can notice setParamValueOf return a reference to NITDevice. This allows us to chain the calls.
Also you can see the call to the function updateConfig(); this function **must** be called to have the parameters sent to the camera.
A good usage is to set the parameters and then call updateConfig() to send all the parameters in one go.

A special parameter is the frame rate.
setFps() permit you to set this value. This value must be comprised in an allowed range.
The frame rate range depends on other parameters of the camera.
This range is calculated when updateConfig() is called.
Another good habit is to set the parameters followed by updateConfig() (this calculate the frame rate range) and then, set the frame rate with setFps() followed by updateConfig().

4.3. Parameter change reporting:

Some parameters are interdependant, that is if you change a parameter this can also change another parameter or the selectable range of another parameter.
To have the program alerterted of this changes, the SDK provide the NITConfigObserver class.
This class is a pure virtual class, you have, as user, to derive a class from NITConfigObserver to catch parameter modifications.

See also

NITConfigObserver.

4.4. Frame processing:

A NITDevice deliver frames (exactly NITFrame objects). These frames have to be processed.
For this purpose we have to connect processing objects to the NITDevice.
Two kind of processing objects are provided:

- NITFilter objects
- NITObserver objects

4.4.1.NITFilter:

A NITFilter is an object which modifies the NITFrame he receives on input, and passes this modified frame to the next processing object.
A NITFilter can be in the enabled or disabled state. When disabled, no transformation is done and the NITFrame is directly passed to the next processing object.
A dedicated sub-namespace, NITToolBox, provides number of predefined filter you can use.

4.4.2.NITObserver:

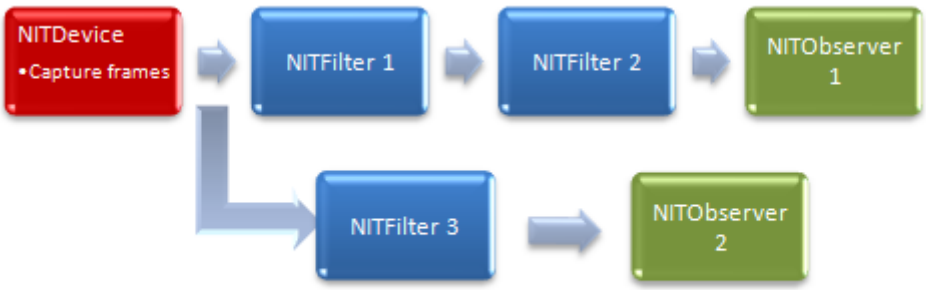
A NITObserver is an object which receives NITFrame objects, use it, but don't modify it.
This kind of object doesn’t pass the NITFrame to another processing object.
This type of object is usually used to display frames or to copy frames. The part of the application not under the control of the SDK then uses the copied frame.
The dedicated sub-namespace, NITToolBox, provide certain predefined observers you can use.

User classes can inherit from NITUserFilter or NITUserObserver to customize the frame processing.

4.4.3.Pipeline:

A NITFilter can be connected to a NITDevice to receive NITFrames. It can also be connected to another NITFilter or NITObserver.
A NITObserver can be connected to a NITDevice to receive frame, or to a NITFilter.
Nothing can be connected to a NITObserver.

We call a chaining of NITFilter and NITObserver a pipeline.



A pipeline

In the above image we have a pipeline with 2 branches. The first branch contain NITFilter1, NITFilter2 and NITObserver1. The second branch contains NITFilter3 and NITObserver2.
In this case, NITDevice makes a copy of the captured frame; send the frame to NITFilter1 and the copy to NITFilter3.
The original frame is modified by NITFilter1 and NITFilter2 and used by NITObserver1(used is for example display the frame).
The copied frame is modified by NITFilter3 and used by NITObserver2(used is for example save the frame).

Our code snippet can now be augmented by the pipeline construction.

```
... preceding code

#We begin by instantiate the filters and observers we need
NITLibrary.NITToolBox.NITAutomaticGainControl agc #A NITFilter who histogram stretch the frame
NITLibrary.NITToolBox.NITColorMap colMap( NITLibrary.NITToolBox.NITColorMap.PINK ) #A
NITFilter #who colorize the raw pixels
NITLibrary.NITToolBox.NITFlip flip( NITLibrary.NITToolBox.NITFlip.HORZ ) #A NITFilter who flip
#the frame horizontally
```

```
NITLibrary.NITToolBox.NITPlayer viewer1( "player 1" )
NITLibrary.NITToolBox.NITPlayer viewer2( "player 2" ) #NITObservers who display the frames

dev << agc << colMap << player1 #first branch of the pipeline
       agc << flip << player2; #second branch of the pipeline

#We are now ready to capture frames
```

4.5. Frame capture:

Once parameters are set, we can begin grabbing images.
NITDevice come with three functions to capture images.

- NITDevice.start() which captures frames until NITDevice.stop() is called.
- NITDevice.captureNFrames() which captures the number of frames passed as parameter.
- NITDevice.captureForDuration() which captures frames for a certain amount of time.

To know when the capture is done when captureNFrames or captureForDuration are called you can use NITDevice.waitEndCapture() which blocks until the capture is done.

The capture doesn't take place in the calling thread.
The function NITConfigObserver.onNewFrame() is called each time a frame is captured with the status of the frame.
For different reasons a frame may be incomplete or invalid; in this case NITConfigObserver.onNewFrame() is called with a status code indicating the frame was dropped. If this happens the frame is not transmitted to the pipeline.

```
... preceding code

dev.start() #start the capture
# do here what you want. I Sleep
time.sleep( 2 )
dev.stop()
```

```
#You can also write
dev.captureNFrames( 1000 )
# do here what you want.
dev.waitEndCapture() #we block waiting the frames are captured
```

```
#Or

dev.captureForDuration( 1000 ) #Capture for 1 second
# do here what you want.
dev.waitEndCapture() #we block waiting the frames are captured
```

4.6. Memory model of the library

At construction of the NITDevice, the SDK allocates space big enough to contain 10 frames of maximum resolution in RGBA format.
This method avoids deallocate/reallocate buffers each time the resolution or the pixel format changes.
Each time a new branch is added to the pipeline, the same process is applied as we have to copy frames flowing in different branches.
When acquisition is started the NITDevice acquire a buffer which is released and given back to the free list of buffers when the buffer arrives at the end of the pipeline branch.

4.7. Threading strategy

At construction of a NITDevice, a pool of 4 threads is created.

When the capture starts, the frames are waited in one of these threads (let's call it Thread_A).

When the totality of the frame is catched, another thread of the pool is fired immediately to wait for the next frame(let's call it Thread_B).

Thread_A call then NITConfigObserver.onNewFrame, and the first filter of the pipeline.

Furthermore, if the pipeline has more than one branch, the remaining branches are called from distinct threads.

This strategy have implications:

- Has NITFilter and NITObserver derived classes can be called from different threads, the critical variables used must be mutex guarded.
- Functions from NITConfigObserver like onNewFrame are called from different threads.
  Particularly if you want to refresh a user interface from these functions, you have to post to the thread where the user interface was created.

In the documentation of the functions, a "\thread" flag is set when caution must be taken.

4.8. Error reporting

In the library errors are reported in two different ways.

- Errors arriving in the thread where a function was called are reported by throwing a NITException. This kind of error can be catched by the typical try catch blocks.
  – Example: calling dev.setParamValueOf("A bad param name", " or a bad param value") will throw a NITException.
- Errors arriving in the internal threads, during the capture phase or in the pipeline are reported by a call to NITConfigObserver.onInternalError.

5. Supported platforms

NITLibrary 3.x supports the following platforms:

- **Architectures:**
  - **x86**

    We recommend at least a cpu core I5 with 4GB of memory.
    Best performance will be obtained with 8 cores.

    - **Windows:**
      32bits and 64bits versions of Windows 8 and Windows 10.
      Compiler: Visual Studio from VS2010 to VS2019.

    - **Linux:**
      64bits versions of Debian 9, Debian 11 Ubuntu20.04 and Ubuntu18.04

It is recommended to use the “High Performance” power plan in the “Power Options” settings of your PC.
Also it is recommended to disable the “USB selective suspend settings” in the “Advanced power settings” for Usb cameras.

On laptop computers it is recommended to use the computer plugged in with at least a battery charge of 50%.
On certain laptops, the performance is degraded if the battery is not enough charged due to the current consumption needed to charge the battery.

To use Gige cameras a Gigabit Ethernet NIC card jumbo frames capable is needed.
We recommand NIC cards of the Intel Pro 1000 familly.

  - **ARM (AARCH64)**

    - **Linux:**
      64bits versions of Ubuntu18.04-Mate.

      The following boards have been tested:
      - **NVidia Jetson TX**
      - **Odroid N2+**
        Lack of jumbo frames in the current kernel imply limited performances for Gige cameras.

      - **RockPro64**
        Lack of jumbo frames in the current kernel imply limited performances for Gige cameras.

- **Raspberry PI4**
  Our tested version has 2GB of RAM. This limits the performances.
  Better results should be obtained with an RPI4 4GB.

6. Installation

A python version corresponding to the NITLibrary must be present on the host.
Numpy corresponding to the Python version must also be present.
The library is named with the following convention:
NITLibrary_x64_<library-version>_py<python-version>.pyd
Example: NITLibrary_x64_320_py39.pyd correspond to NITLibrary 3.2.0 for python 3.9.x

6.1. Windows:

Package layout:
NITLibrary-Python 3.x.x

```
├────bin
│ ... └────PYTHON35
│ ... │ .....NITLibrary_x64.dll
│ ... │ .....NITLibrary_x64_3xx_py35.pyd
│ ... │
│ ... └────PYTHON36
│ ... │ .....NITLibrary_x64.dll
│ ... │ .....NITLibrary_x64_3xx_py36.pyd
│ ... │
│ ... └────PYTHON38
│ ... │ .....NITLibrary_x64.dll
│ ... │ .....NITLibrary_x64_3xx_py38.pyd
│ ... │
│ ... └────PYTHON39
│ ... │ .....NITLibrary_x64.dll
│ ... │ .....NITLibrary_x64_3xx_py39.pyd
├────doc
│ ... │ ......documentation
│
├──── Samples
```

Usb Driver installation:
If you use an USB camera, a Cypress driver has to be installed.
This step is not mandatory if the driver was already installed by another software component.

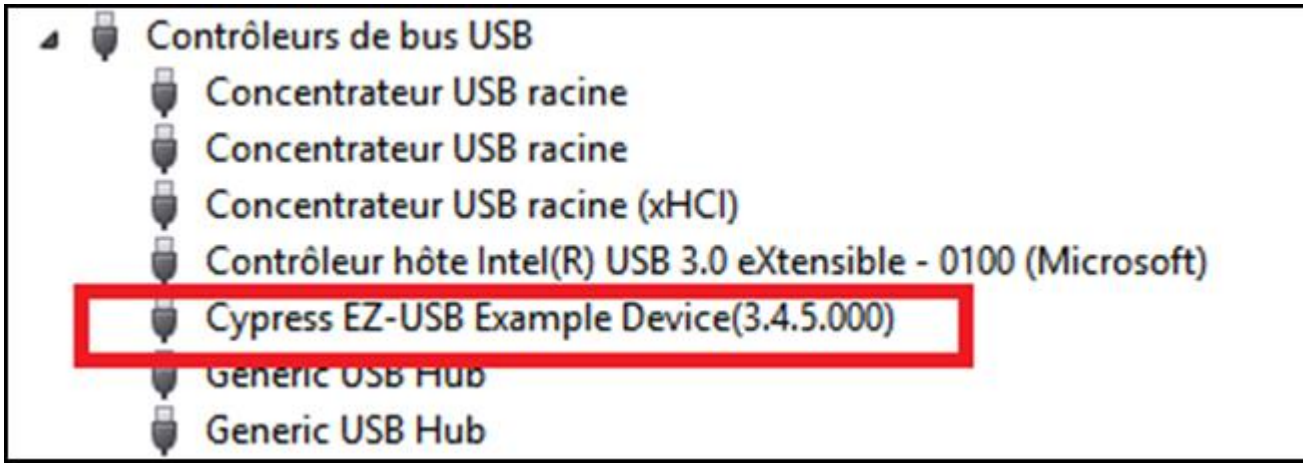
You will find a Driver folder on the software package.
The folder contains drivers for multiple Windows platform.
Please follow these steps to install correct driver:

1. Connect the camera board to the PC USB port.
   A Dialog Box inviting the user to install a new device driver should then pop up.

In the event this screen does not show up you can have access to the new device in the system device manager in the system options.

2. Then Right Click on the unknown device to install the driver and follow the installation procedure as follow:
3. Select Update Driver Software …
4. Select “Browse my computer for driver software”
5. Browse the correct platform driver from “…\DRIVERS”.

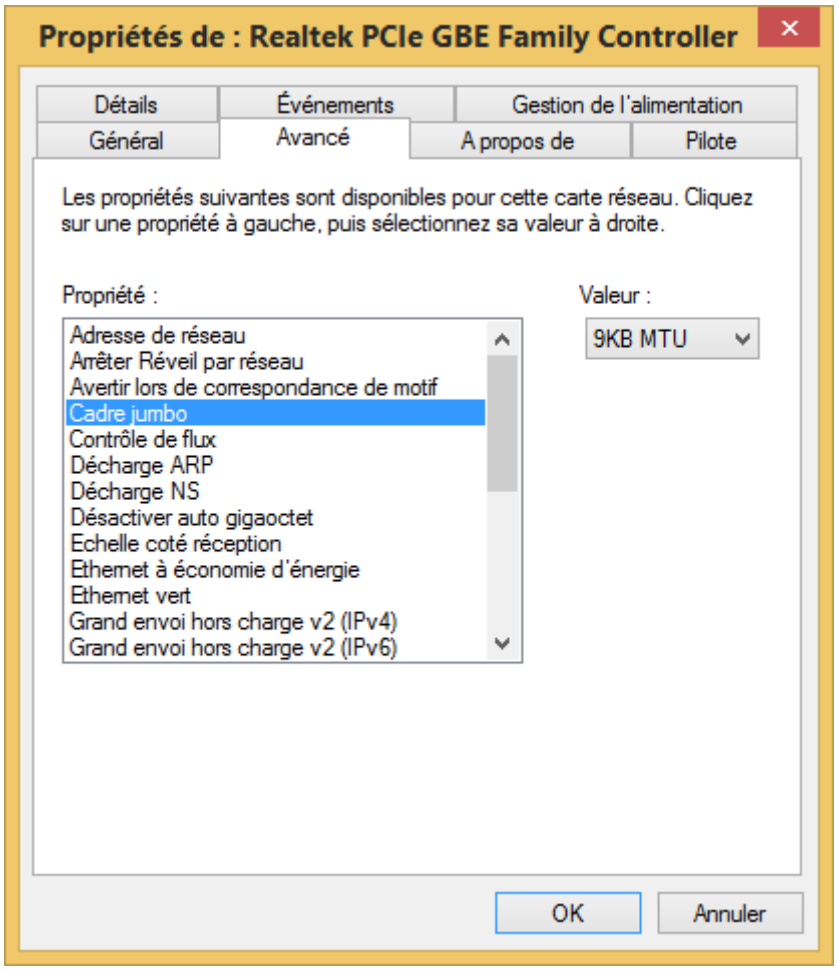
Now verify in the device manager that you have a Cypress USB device in the list.



Usb Driver Installation

Gige installation:

If you use a Gige camera, you have to activate Jumbo frames.
Set it to the max available jumbo frames size(9000).



Gige Installation

The application that use the SDK has to be allowed by the firewall to have the streaming available.
Avoid use of switches between the camera and the host. Recommended configuration is having the camera directly plugged in the the NIC card.

6.2. Linux:

Package layout:

The complete bundle is provided as a tar file containing a deb package and a sample code folder.

- Untar the file
- The sample code folder can be leaved in place.
- To install the deb package, run 'sudo apt-get <Name of the NITLibrary package>.
  – NITLibrary.so.xxx will be installed in /usr/local/lib.
  – Header files will be installed in /usr/local/include/NITLibrary-xxx.
  – Documentation will be installed in /usr/share/NITLibrary.
  – For usb devices a rule file(88-cyusb.rules) is installed in /etc/udev/rules.d/

Usb installation:

By default, USB-FS on Linux systems only allows 16 MB of buffer memory for all USB devices.
This may result in image acquisition issues from high-resolution cameras or multiple-camera set ups.
This limit can be increased to make use of the camera's full capabilities.
A minimal value should be 1000 MB.
The configuration of the 'usbfs_memory_mb' who govern the allocated memory by the operating system is dependant of the Linux flavor and board type.
For systems using Grub as bootloader, a good choice is to modify '/etc/default/grub' and to append the text 'usbcore.usbfs_memory_mb=1000' to the line beginning with 'GRUB_CMDLINE_LINUX_DEFAULT'.
After the modification of the file, you have to call 'sudo update-grub', followed by 'sudo modprobe usbcore usbfs_memory_mb=1000' and finally reboot.
For other situations, we invite you to google 'usbfs_memory_mb <your board model>'.

Gige installation:

If you use a Gige camera, you have to activate Jumbo frames.
Set the mtu to the max available jumbo frames size(9000). Avoid use of switches between the camera and the host. Recommended configuration is having the camera directly plugged in the the NIC card.

The network interface can have to be set to Local Link if you use the camera in LLA mode.

7. NUC and BPR

7.1. NUC

NUC stands for Non Uniformity Correction.

To correct the non uniformities of the SWIR camera sensor, a non uniformity correction is applied to the raw output of the camera.
Non uniformities are dependant from the pixel clock, exposure time and for certain cameras(NSC1601, NSC1902), temperature, gain and sensor response.
When you receive one of our SWIR cameras, you receive also a folder named with the serial number of the camera (Example: SN160427).
This directory contains a hierarchy of subfolders ending with NUC files tailored for your camera, for certain exposure times.
The directory containing the Nuc files for the camera is named “the official NUC folder” in the next part of this document.
The non uniformity correction work in two modes “Interpolated NUC” (the default mode) and “Manual NUC”.

When a NITDevice is created, it try to find the official NUC folder in the following locations.

- Windows: The folder containing the executable.
- Linux: /var/lib/NIT/.

If the folder is found, NUCs are automatically activated in automatic NUC mode.
Else, NUCs are not set.

Out of the NITDevice creation, NUCs are governed by the following modes:

- *Automatic NUC*:

  In this mode, the NUC file used is changed automatically when a dependant parameter like 'Exposure Time' is changed.
  This mode is active if the NITDevice constructor found the official NUC folder or if NITDevice.setNucDirectory is called with a valid path.

  - If a Nuc file exists for the current parameters:
    This NUC file is applied.
    In this case NITConfigObserver.onNucChanged is called with the string "Exact nuc applied <nucFile>" and a status of 0.

- If no NUC file exists for the current parameters:
  - If a NUC file exists for an exposure time below **and** above the exposure time selected:
    An interpolation is done with the two files to provide an interpolated NUC.
    In this case NITConfigObserver.onNucChanged is called with the string "<nucPath>: interpolated nuc applied <expo low> - <expo high>" and a status of 1.
  - If the exposure time is below the lowest exposure time for which a NUC file exists:
    This NUC file is applied.
    In this case NITConfigObserver.onNucChanged is called with the string "First known nuc applied <nucFile>" and a status of -1.
  - If the exposure time is above the highest exposure time for which a NUC file exists:
    This NUC file is applied.
    In this case NITConfigObserver.onNucChanged is called with the string "Last known nuc applied <nucFile>" and a status of -1.
- If no NUC path exists for the current parameter set:
  No nuc is applied.
  In this case NITConfigObserver.onNucChanged is called with the string "<nucPath>" and a status of 2.

- *Manual NUC*:

  In this mode, the NUC file is fixed by the user and doesn't change with the parameters.
  This mode is active if NITDevice.setNucFile is called with a valid file.
  Each time a parameter is changed, NITConfigObserver.onNucChanged is called with the string "<nucPath>" and a status of 3.

Regardless of the NUC mode, NUC processing can be activated/deactivated by calling NITDevice.activateNuc().

7.2. *BPR*

BPR stands for Bad Pixel Recovery.

Some pixels of the sensor can be defective for certain parameters but not for others. The values of these pixels are replaced by an interpolation of surrounding pixels.
Bad pixels are dependant from pixelClock and for SenS cameras from sensor response.

The official NUC folder contains also BPR files.

At construction of a NITDevice object, if an official NUC folder is found, the BPR is automatically activated in automatic mode.

As for NUC management, BPR can work in two modes:

- *Automatic BPR*:

  In this mode, the BPR file used is changed automatically when a dependant parameter like 'Pixel Clock' is changed.
  This mode is active if the NITDevice constructor found the official NUC folder or if NITDevice.setNucDirectory is called with a valid path and the parameter is_bpr_path is set to true or if NITDevice.setBprDirectory is called with a valid path.

- *Manual BPR*:

  In this mode, the BPR file is fixed by the user and doesn't change with the parameters.
  This mode is active if NITDevice.setBprFile is called with a valid file.

Regardless of the NUC mode, NUC processing can be activated/deactivated by calling NITDevice.activateNuc().

Specific GIGE cameras

These cameras embed NUC and BPR files and processing.

This processing is governed by parameters passed directly to the camera(see camera documentation)

. At construction of the NITDevice object, the SDK processing is deactivated. That is NITDevice.activateNuc(false) is called.

This deactivate the SDK processing of NUC(not the embedded processing).

It is possible to use the above process for Gige cameras.

When you call NITDevice.activateNuc(true), the embedded NUC is automatically deactivated.

When you call NITDevice.setParamValueOf("NucActivate", 1), the embedded NUC is activated and the SDK NUC is deactivated.

8. Library Reference

8.1. Namespace List

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

NITLibrary	Main namespace of the library All other namespaces are in this namespace.
Demosaicing	Namespace of the functions relative to demosaicing, only applicable to color cameras
Gige	Namespace of helpers for **Gige** cameras
NITToolBox	Namespace of the library filters

8.2. NITLibrary Namespace Reference

Main namespace of the library.

Namespaces

Demosaicing

namespace of the functions relative to demosaicing, only applicable to color cameras.

Gige

namespace of helpers for **Gige** cameras.

NITToolBox

namespace of the library filters

Classes

class	**NITConfigObserver**
	Permits to the user of the SDK to be informed of changes in an USB camera.
class	**NITConfigObserverGige**
	Permits to the user of the SDK to be informed of changes in a **Gige** camera.
class	**NITDevice**
	Main class which describe the active camera.

class	**NITFilter**
	Base class of all the filters.

class	**NITFrame**
	Class who embed the data of the captured frame.

class	**NITManager**
	Manages all the connected devices and allow the instantiation of a **NITDevice** object to control the camera.

class	**NITObserver**
	Base class of all the observers.

Enumerations

enum	**ConnectorType** { NO_CONNECTOR, **USB_2**, **USB_3**, GIGE=4 }

Enumeration Type Documentation

ConnectorType

NO_CONNECTOR	
USB_2	Indicates the camera is connected thru an USB 2.0 connector.
USB_3	Indicates the camera is connected thru an USB 3.0 connector.
GIGE	Indicates the camera is connected thru an Gige connector.

8.2.1.NITLibrary.NITConfigObserver Class Reference

Permits to the user of the SDK to be informed of changes in an USB camera.

This class is pure virtual.
To receive informations about the camera changes, you have to derive from this class and overload the functions.
The derived class object must then be connected to a **NITDevice**.

As soon a the object is connected to a **NITDevice**, onParamRangeChanged is called for each parameter of the camera.
This permit to obtain the name of all supported parameter, there current range and current value.

def **onParamChanged** (param_name, str_value, num_value)

Called when a parameter is changed.

A parameter can change because a call was made to **NITDevice.setParamValueOf()** or **NITDevice.setFps()** or because the parameter is a dependant parameter(it depend from the state of another parameter).

Parameters

param_name	Name of the parameter who was changed.
str_value	New value of the parameter in string format.
num_value	New value of the parameter in numeric format.

Threading:

This function is called in the thread where the **NITDevice.setParamValueOf()** function was called.

def **onParamRangeChanged** (param_name, str_values, num_values, array_size, cur_string_value, cur_num_value)

Called when a parameter range is changed.

A parameter range can change because the parameter is a dependant parameter(it depend from the state of another parameter).

Parameters

param_name Name of the parameter who was changed.

str_values Array of new values of the parameter in string format.

num_values Array of new values of the parameter in numeric format.

array_size Size of the arrays.

cur_str_val New current value of the parameter in string format.

cur_num_val New current value of the parameter in numeric format.

Threading:

This function is called in the thread where the **NITDevice.setParamValueOf()** function was called.

def **onFpsChanged** (new_fps)

Called when the frame rate is changed.

The frame rate can change because a call was made to **NITDevice.setParamValueOf()** or **NITDevice.setFps()**.

Parameters

new_fps new value of the frame rate.

Threading:

This function is called in the thread where the **NITDevice.setParamValueOf()** or **NITDevice.setFps()** function were called.

def **onFpsRangeChanged** (new_fpsMin, new_fpsMax, new_fps)

Called when the current frame rate range is changed.

The frame rate range can change because a call was made to **NITDevice.setParamValueOf()** or **NITDevice.setFps()**.

Parameters

new_fpsMin,new_fpsMax New value of the frame rate range.

new_fps New value of the frame rate.

Threading:

This function is called in the thread where the

NITDevice.setParamValueOf() or **NITDevice.setFps()** function were called.

def **onNewFrame** (status)

Called each time a new frame is captured.

If the status parameter is not 0, it mean the frame was not complete or corrupted.
In this case the frame is dropped. The connected filters have no knowledge of this dropped frame.

Parameters

status Status of the incoming frame. 0 if frame Ok, else not 0.

Threading:

This function is called from one of the streaming threads

def **onNucChanged** (nuc_str, status)

Called when NUC is changed.

Each time some parameters like exposure are changed, the current nuc if enabled is changed.
This function notifies the application of the changes.
A status number indicates which kind of nuc is applied:
If status is -1: The selected exposure time is below or above the exposure times for which a Nuc file exists. In this case
the nearest possible Nuc is applied.
If status is 0: The Nuc is an exact Nuc coming from one of the Nuc files for the current exposure.
If status is 1: The Nuc is interpolated with the nearest Nuc files corresponding to the current exposure.
If status is 2: There is no nuc in this path.
If status is 3: The Nuc is a fixed Nuc the one selected by a call to setNucFile().

Parameters

nuc_str a string representing the current NUC.

status status of the applied nuc

Threading:

This function is called in the thread where the **NITDevice.setParamValueOf()** or **NITDevice.setFps()** function were called.

8.2.2.NITLibrary.NITConfigObserverGige Class Reference

Permits to the user of the SDK to be informed of changes in a **Gige** camera

This class is pure virtual and derived from **NITConfigObserver**.
To receive informations about the camera changes, you have to derive from this class and overload the functions.
The derived class object must then be connected to a **NITDevice**.

As soon the object is connected to a **NITDevice**, onNewGroup is called for each **Gige** group of parameters and onNew*Param is called for each parameter of the camera. This permits to obtain the name of all supported parameters, their current range and current value.

def **onNewGroup** (group_name)

Called when the **NITConfigObserverGige** is connected to the camera.

Parameters

group_name name of the gige group of parameters.

Threading:

This function is called in the thread where the object was connected to the **NITDevice**.

def **onNewBoolParam** (group_name, param)
param is of type
NITLibrary.Gige.BoolParam

def **onNewIntParam** (group_name, param)
param is of type
NITLibrary.Gige.IntParam

def onNewFloatParam (group_name, param)
param is of type
NITLibrary.Gige.FloatParam

def **onNewEnumParam** (group_name, param)
param is of type
NITLibrary.Gige.EnumParam

def **onNewStringParam** (group_name, param)
param is of type
NITLibrary.**Gige.StringParam**

def **onNewCommandParam** (group_name, param)
param is of type
NITLibrary.**Gige.CommandParam**

Called when the **NITConfigObserverGige** is connected to the camera.

Parameters

group_name name of the gige group of this parameters.

param a structure giving informations about the parameter.

Threading:

These functions are called in the thread where the object was connected to the **NITDevice**.

def **onParamChanged** (param_name, const char *str_value, float num_value) See NITConfigObserver

def **onParamRangeChanged** (param_name, string_values, numeric_values, array_size, cur_string_value, cur_numeric_value) See NITConfigObserver

def **onFpsChanged** (new_fps) See NITConfigObserver

def **onFpsRangeChanged** (new_fpsMin,new_fpsMax, new_fps) See NITConfigObserver

def **onNucChanged** (nuc_str, status)

Called when external NUC is active and changed.

Gige cameras embeds Nucs(this are internal Nucs).
This internal Nucs can be deactivated and external Nucs can be activated in the **NITDevice** object.
This function applies only to external Nucs.
See NITConfigObserver

def **onNewFrame** (status) See NITConfigObserver

8.2.3.NITLibrary.NITDevice Class Reference

Main class which describe the active camera.

Thru this class, you can set and get the camera parameters and initiate the capture of frames.
You can't instantiate directly this class. Instead, **NITManager** will return an instance of this class when you call one of the open functions.

See also

NITManager.openOneDevice(), **NITManager.openDevice()**, **NITManager.openBySerialNumber()**

def **sensorWidth** () const

Return the sensor width.

def **sensorHeight** () const

Return the sensor height.

def **connectorType** () const

Return the type of the connector on which the camera is connected.

Can return one of NITLibrary.ConnectorType.USB_2, NITLibrary.ConnectorType.USB_3 or NITLibrary.ConnectorType.GIGE
This is the connector on the host side, notably for USB cameras if an USB3 camera is connected to an USB2 connector on the host side, this function will return USB_2.

def **connectorNumber** ()

Return the index of the camera in the **NITManager** list.

def **commercialName** ()

Return the commercial name of the camera.

def **modelId** ()

Return the sensor model of the camera. Like “NSC1003”.

def **firmwareVersion** ()

Return the FGPA firmware version of the camera as string.

def **serialNumber** ()

Return the serial number of the camera as string.

def **setNucDirectory** (dir_path, is_bpr_path=true)

Set the directory containing the NUC files.

The path (if not empty) must point to a writeable directory containing a

directory named 'NUC'.
If directory path is valid, the automatic nuc interpolation is activated.
If directory path is empty, the nuc filtering is deactivated.
If is_bpr_path is true, the bad pixel recovery is also activated with the files contained in this path.

Parameters

dir_path	path to the directory
is_bpr_path	true if the directory contain also the bpr files default value = true if not set

Exceptions

NITException	if the directory doesn't exist
NITException	if the directory doesn't contain a directory named 'NUC'
NITException	if the directory isn't writeable

def **setNucFile** (file_path)

Set a NUC file.

The path (if not empty) must point to a valid NUC file.
The nuc interpolation is deactivated and the NUC in this file is applied regardless the parameter changes.
If file_path is empty, the nuc filtering is deactivated.

Parameters

file_path	path to the NUC file

Exceptions

NITException	if the file doesn't exist
NITException	has not an 'yml' extension

def **setBprDirectory** (dir_path)

Set the directory containing the BPR files.

The path (if not empty) must point to a directory containing a directory named 'NUC'.
If directory path is valid, the automatic bad pixel correction is activated.

If directory path is empty, the bad pixel correction filtering is deactivated.

Parameters

dir_path path to the directory

Exceptions

NITException if the directory doesn't exist

NITException if the directory doesn't contain a directory named 'NUC'

def **setBprFile** (file_path)

Set a BPR file.

The path (if not empty) must point to a valid BPR file.
The bad pixel correction in this file is applied regardless the parameter changes.
If file_path is empty, the bad pixel correction is deactivated.

Parameters

file_path path to the BPR file

Exceptions

NITException if the file doesn't exist

NITException has not an 'yml' extension

def **activateNuc** (activate=true)

Activate/Deactivate the nuc processing.

Attention

The nuc processing is only activated if a valid nuc path or nuc file exists.

Parameters

activate if true try to activate the nuc processing, else deactivate the nuc processing
default value = true if not set

def **activateBpr** (activate=true)

Activate/Deactivate the bpr processing.

Attention

The bpr processing is only activated if a valid bpr path or bpr file exists.

Parameters

activate if true try to activate the bpr processing, else deactivate the bpr processing
default value = true if not set.

def **nucActive** ()

return the state of the nuc processing. true if active, else false

def **bprActive** ()

return the state of the bpr processing. true if active, else false

def **setFps** (new_fps)

Set the frame rate.

Set the nearest frame rate possible in the range delimited by **NITDevice.minFps()** and **NITDevice.maxFps()**

Parameters

new_fps

Emit:
NITConfigObserver.onFpsChanged

def **fps** ()

Return the current frame rate.

def **minFps** ()

Return the currently minimum frame rate.

def **maxFps** ()

Return the currently maximum frame rate.

def **setRoi** (offset_x, offset_y, width, height)

Set the position and dimensions of the returned frames.

This function permit to set the region of interest atomically. When you call setParamValueOf with 'Number of lines', the SDK sum this new value with the current value of 'First line'.
If this sum is above the sensor height, an exception is thrown. This can lead to tricky situations. Let say sensor height is 512 and the current 'First line' is 10 and the current 'Number of lines' is 502.
If you want to set 'Number of Lines to 507, you first have to set 'First line'

below 6 and then set 'Number of lines' to 507.
If you do it the other side, the SDK will throw.

setRoi permit to avoid this situation. The sum is made with the parameters passed to the function.

Parameters

offset_x,offset_y,width,height roi elements.

Emit:

NITConfigObserver.onParamChanged for each parameter who as effectively changed.

Exceptions

NITException if the offset_x + width or offset_y + height is above the sensor dimensions

def **paramValueOf** (paramName)

Return the current value of the parameter paramName in numeric format.

The current value is the value in the camera.

```
clock = dev.paramValueOf( "PixelClock" ) #Return the value in the camera let say
25.0
dev.setParamValueOf( "PixelClock", 50.0 ) #The new value is not yet sent to the
camera
clock = dev.paramValueOf( "PixelClock" ) #Return the value in the camera which
is always 25.0
dev.updateConfig() #The new value is sent to the camera
clock = dev.paramValueOf( "PixelClock" ) #Return the value in the camera which
now 50.0
```

For parameters where the numeric format has a meaning, the returned value is the current value.

```
expo = dev.paramValueOf( "ExposureTime" ) #can return 200.0 for example
```

For parameters that are enumerations, the return value is the index of the enumeration.

```
current_mode = dev.paramValueOf( "Mode" ) #can return 0 if capture mode is
'Global Shutter'
```

Parameters

paramName For a list of available parameters see the page corresponding to your camera.

Exceptions

NITException if paramName is not supported by the current camera.

def **paramStrValueOf** (paramName)

Return the current value of the parameter paramName in string format.

The current value is the value in the camera.

```
clock_str = dev.paramStrValueOf( "PixelClock" ) #Return the value in the camera
let say "25MHz"
dev.setParamValueOf( "PixelClock", "50MHz" ) #The new value is not yet sent to
the camera
clock_str = dev.paramStrValueOf( "PixelClock" ) #Return the value in the camera
which is allways "25MHz"
dev.updateConfig(); #The new value is sent to the camera

clock_str = dev.paramStrValueOf( "PixelClock" ) #Return the value in the camera
which now is "50MHz"
```

Parameters

paramName For a list of available parameters see the page corresponding to your camera.

Exceptions

NITException if paramName is not supported by the current camera.

def **setParamValueOf** (paramName, value)

Prepare to change the current value of the parameter paramName

value can be of type integer, double or string.

USB: The change is sent to the camera on the next call to **NITDevice.updateConfig()**.
GIGE: The change is sent to the camera immediatly.
For some parameters the change may induce modifications of other parameters.
The return value permit to chain calls.

```
dev.setParamValueOf( "OneParameter", OneValue ).setParamValueOf(
"AnotherParameter", "anotherValue" ).updateConfig()
```

Parameters

paramName For a list of available parameters see the page corresponding to your camera.

value The new value to apply.

Returns

A reference to this NITDevice

Exceptions

NITException if paramName is not supported by the current camera.

NITException if value is ill-formed.

NITException if value is out of range or not applicable to the current camera.

def **updateConfig** ()

Call the **NITConfigObserver** functions to notify the parameter changes.

USB: Send the changed parameters to the camera.

Emit:

NITConfigObserver.onParamChanged for all changed parameters
NITConfigObserver.onParamRangeChanged if needed
NITConfigObserver.onFpsChanged if needed
NITConfigObserver.onFpsRangeChanged if needed

def **start** ()

Start the capture.

The streaming run until **NITDevice.stop()** is called.

Emit:

NITConfigObserver.onNewFrame() For each received frame

Threading:

The capture don't run in the current thread

Exceptions

NITException if the streaming is currently running

def **captureNFrames** (n, error_increment_count=false)

Start the capture until n frames are captured.

The streaming stop automatically when the number of frames has been captured. This function must be coupled with **NITDevice.waitEndCapture()**

```
dev.captureNFrames( 10 )
dev.waitEndCapture()
```

Parameters

n The number of frames to capture

error_increment_count If false, dropped frame don't increment the counter
default value = false if not set

Emit:

NITConfigObserver.onNewFrame() For each received frame

Threading:

The capture don't run in the current thread

Exceptions

NITException if the streaming is currently running

def **captureForDuration** (milliseconds)

Start the capture for duration.

The streaming stops automatically when the time has elapsed. This function must be coupled with **NITDevice.waitEndCapture()**

```
dev.captureForDuration( 1000 ) #Capture for 1 second
dev.waitEndCapture()
```

Parameters

milliseconds The number of milliseconds to capture

Emit:

NITConfigObserver.onNewFrame() For each received frame

Threading:

The capture don't run in the current thread

Exceptions

NITException if the streaming is currently running

def **stop** ()

Stop the capture.

def **waitEndCapture** (timeout=-1)

Block the current thread until the capture is done.

This function is coupled
with **NITDevice.captureNFrames()** or **NITDevice.captureForDuration()**

```
dev.captureForDuration( 1000 ) #Capture for 1 second
dev.waitEndCapture()
```

Parameters

timeout The maximum number of milliseconds to wait for the end of capture.
If -1, the function waits until the job is done.
If a timeout occurs, the streaming is stopped.

Returns

false if the function timed out, else true if the job was honored

def **operator<<** (NITFilterDerivedObject)

Permit to connect a **NITFilter** derived object.

Returns

The reference to the **NITFilter** object for chaining

def **operator<<** (**NITObserver**)

Permit to connect a **NITObserver** derived object.

def **operator<<** (**NITConfigObserver**)

Permit to connect a **NITConfigObserver** derived object.

def **getCurrentNucPath** ()

Return the current path of the NUC.

8.2.4.NITLibrary.NITFilter Class Reference

Base class of all the filters.

A filter receive a frame through the function **NITFilter.onNewFrame()** and modify it in-place.

Filters can be chained together and to **NITObserver** objects.
If more than one processing object is connected to a **NITFilter**, each added processing object imply the creation of a new branch of the pipeline running in his own thread.

Finally a **NITFilter** can be active (default at construction), in this case the transformation of the frame is done
or it can be inactive in which case the frame is simply passed to the next stage.

All the filters of the **NITToolBox** derive from **NITFilter**.

To apply your own processing to the frames; you have to derive a class
from **NITUserFilter** who derives from **NITFilter** and overload
the **onNewFrame()** function.

def **operator<<** (next_filter)

operator to connect a **NITFilter** to this **NITFilter**. Returns next_filter to permit chaining.

def **operator<<** (next_observer)

operator to connect a **NITObserver** to this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state If true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

Return the current state of the **NITFilter**.

def **onNewFrame** (frame)

Called each time a new frame is captured. You must return the modified **NITFrame**.

Parameters

frame frame to process of type **NITFrame**

Threading:

This function can be called from different threads.

8.2.5.NITLibrary.NITFrame Class Reference

Class who embed the data of the captured frame.

When a frame is captured by a NITdevice, a **NITFrame** is constructed and passed to the pipeline.

The frame data is 16 bytes aligned.
Each pixel occupies 4 bytes.

enum **ePixType** { **RGBA** , **FLOAT** }

Enumerator	
RGBA	pixel format for color images. One byte per channel. 32 bits
FLOAT	pixel format for grayscale images. [0.0-1.0] range. 32 bits

def **bitsPerPixel** ()

Return the number of bits per pixel of the sensor

This value corresponds to the number of bits per pixel of the sensor regardless of the pixelType.

Returns
return the number of bits per pixel

def **data** ()

Return a pointer to the pixel data

Attention
If the pixel type is RGBA the returned value has to be casted to unsigned int to traverse each pixel or unsigned char to traverse each pixel element.

Returns
return a pointer to the pixel data

def **rows** ()

Return the number of rows of the frame.

def **columns** ()

Return the number of columns of the frame.

def **temperature** ()

Return the temperature of the camera when the frame was captured

Only for SWIR cameras. Always 0 for other cameras.

Returns
return the temperature of the camera in Celsius

def **pixelType** ()

Return the pixel format of the frame (**ePixType** enumeration)

def **id** ()

Return the frame number of this frame

- USB cameras: This value is reset at **NITDevice** construction. max value is MAX_UNSIGNED_LONG_LONG
- GIGE cameras: This value is reset when the streaming port is opened. max value is MAX_UNSIGNED_SHORT

Returns

return the frame number of this frame

def **gigeTimestamp** ()

Return the time in microseconds elapsed since the last reset of the Gige/Usb Timestamp

For USB cameras, this value is the timestamp provided by the camera for SenS 1280V series. For other USB cameras, value is 0.

Returns

return the time in microseconds elapsed since the last reset of the Gige/Usb Timestamp

def **setPixelType** (pix_type)

Set the pixel format of the frame

This function is used by NITFilters who transform gray images to color images or the inverse.

Parameters

pix_type The new pixel format

Returns

return the pixel format of the frame

8.2.6.NITLibrary.NITManager Class Reference

Manages all the connected devices and allow the instantiation of a **NITDevice** object to control the camera.

This class is a singleton. A unique instance of this class is automatically created on the first call to NITManager.getInstance(). **NITManager** maintain a list of connected cameras. At the call of one of the functions dealing with cameras, if the list is empty, it is populated. The list of connected cameras is populated in the order the cameras are discovered.

@staticmethod def **getInstance**()

Instantiate the only **NITManager** object.

def **reset** ()

Reset the **NITManager** object. After this operation, all cameras are destroyed.

def **deviceCount** ()

Return the number of device discovered.

When the maintained list is empty, calling this function, will try to discover the connected cameras.

Returns

The number of connected cameras the last time the list was populated.

def **stringDescriptor** (index)

Return a string describing the device at index.

Parameters

index of the device.

Returns

A string describing the camera

Exceptions

NITException if the maintained list is empty.

NITException if the index is out of the maintained list bounds.

def **listDevices** ()

Return a string describing all the devices discovered.

When the maintained list is empty, calling this function, will try to discover the connected cameras.

Returns

A string describing the cameras

Exceptions

NITException if the maintained list is empty after discovery.

def **openDevice** (index)

Open the device at the given index.

When the maintained list is empty, calling this function, will try to discover the connected cameras.

Parameters

index The index of the device in the maintained list.

Returns

A pointer to the open device.

Exceptions

NITException if the maintained list is empty.

NITException if the index is out of the maintained list bounds.

NITException if the model of the camera cannot be detected.

NITException if the camera at index is not a NIT camera(Gige only).

def **openOneDevice** ()

Open the first discovered device.

When the maintained list is empty, calling this function, will try to discover the connected cameras.

Returns

A pointer to the open device.

Exceptions

NITException if the maintained list is empty.

NITException if the model of the camera cannot be detected.

NITException if the camera at index is not a NIT camera(Gige only).

def **openBySerialNumber** (sn)

Open a discovered device by serial number.

When the maintained list is empty, calling this function, will try to discover the connected cameras.

Parameters

sn The serial number of the queried device.

Returns

A pointer to the open device.

Exceptions

NITException if the maintained list is empty.

NITException if the model of the camera cannot be detected.

NITException if no camera with the given serial number exists.

@staticmethod **getSDKVersion** ()

Return the current SDK version.

Can be called without instantiating an object:

```
NITLibrary.NITManager.getSDKVersion()
```

Returns

string representing the SDK version in the form 3.x.y.

@staticmethod **use** (connector_types, min_number_of_gige_cameras=1, max_gige_discovery_time=15)

Set on which connector type **NITManager** will try to discover cameras.

Can be called without instantiating an object:

```
NITLibrary.NITManager.use( NITLibrary.USB_3|NITLibrary.GIGE )
```

If connector_types contains one of USB_2 or USB_3, **NITManager** try to find USB cameras.
If connector_types contains GIGE, **NITManager** try to find **Gige** cameras. The discovery process for **Gige** cameras is guided by the values of min_number_of_gige_cameras and max_gige_discovery_time.
If min_number_of_gige_cameras == 0, **NITManager** try to find gige cameras until max_gige_discovery_time seconds as elapsed.
If max_gige_discovery_time == 0, **NITManager** try to find min_number_of_gige_cameras.
If both parameters are not 0, **NITManager** try to find min_number_of_gige_cameras. If max_gige_discovery_time as elapsed, **NITManager** returns with the number of found cameras.

Parameters

connector_types	ORed ConnectorType.
min_number_of_cameras	The minimum number of **Gige** cameras to discover.
max_gige_discovery_time	The maximum time to search for **Gige** cameras(seconds).

Exceptions

NITException	if GIGE connector type is set and min_number_of_gige_cameras and max_gige_discovery_time are both 0.

8.2.7.NITLibrary.NITObserver Class Reference

Base class of all the observers.

An observer receive a const frame through the function **NITObserver.onNewFrame()**.

These instances of this class are used at the end of a pipeline branch.
The frame data must be copied for further use. The copied data is not managed by the SDK.

Observers can be connected to **NITFilter** objects or directly to a **NITDevice**.
All the observers of the **NITToolBox** derive from **NITObserver**.

To apply your own processing to the frames; you have to derive a class from **NITUserObserver** who derives from **NITObserver** and overload the **onNewFrame()** function.

def **disconnect** ()

disconnect this observer from the pipeline

def **onNewFrame** (frame)

Called each time a new frame is captured.

Parameters

frame frame to copy or to collect information from

Threading:

This function can be called from different threads.

8.2.8.NITLibrary.NITStackedBlock Union Reference

Helper class used to manage the 'Stacked Block_x' parameter available on some NIT cameras.

This union permit to fill the offsetY and height parameter of a stacked block and to get an unsigned int or string formatted to pass to NITLibrary.NITDevice.setParamValueOf(). The result of NITLibrary.NITDevice.paramValueOf() and NITLibrary.NITDevice.paramStrValueOf() can also be passed to this union to get the encoded offsetY and height.

def __init__ (self, offset_y, height)

Construct a **NITStackedBlock** object.

Use this constructor to get an unsigned int to pass to setParamValueOf

Parameters

offset_y Offset of the stacked block relative to the preceding enabled stacked block (for the first enabled stacked block, the offset is relative to the first line of the sensor.

height Height of the block.

def __init__ (self, raw)

Construct a **NITStackedBlock** object.

Use this constructor to get offset and height from the unsigned int returned by paramValueOf

Parameters

raw concatenated values of offsetY(high part) and height(low part)

def **to_int** ()

cast operator

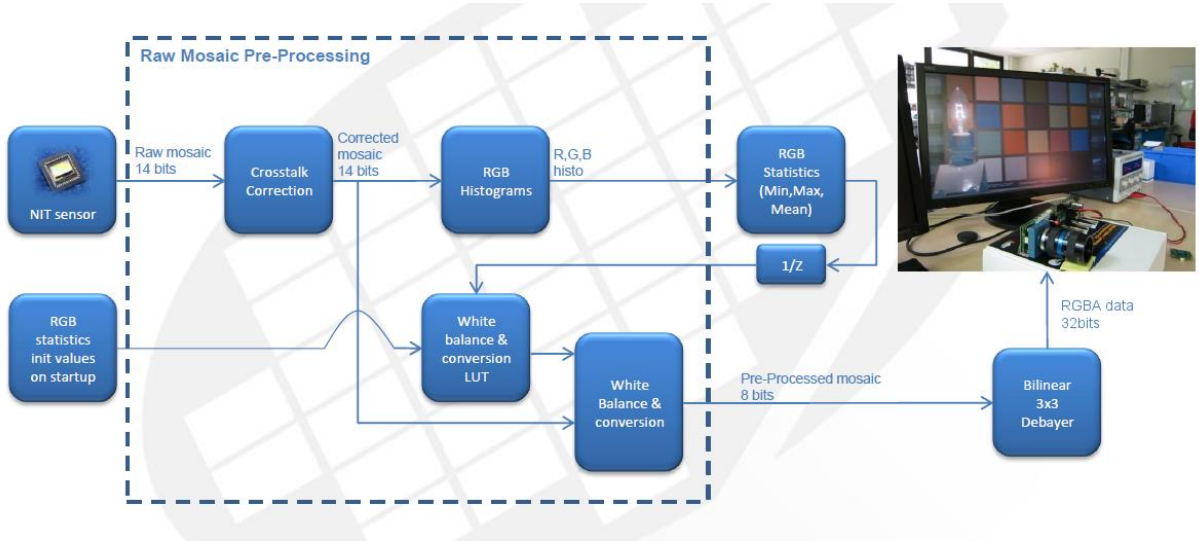
def **to_string** ()

8.3. NITLibrary.Demosaicing Namespace Reference

namespace of the functions relative to demosaicing, only applicable to color cameras.

USB:
NIT visible USB color cameras output RAW bayer data.
By default, this data is passed to the pipeline filters as float gray pixels.
If demosaicing filter is activated, the RAW bayer data is demosaiced. As a result, data is passed to the pipeline filters as RGBA pixels (32bits).

GIGE:
NIT visible GIGE color cameras can output data as correction_Color data. In this case, an internal debayering is applied.
One can also set the output format of the camera to Bayer_GR16. In this case the output is RAW data(internal debayering is not active).
In this case, the SDK debayering can be applied by activationg it.



def **activate** (device, b_activate=true)

activate or deactivate the demosaicing filter

GIGE: Activating SDK debayering disable automatically internal debayering

Parameters

device The device for which demosaicing must be activated. This device must be color capable.

b_activate If true the demosaicing is activated, else the demosaicing is deactivated

Exceptions

NITException if device is not a device supporting Demosaicing(currently only MC1003 can be demosaiced).

def **active**(device)

return the demosaicing state

Returns

true if demosaicing active, else false.

Exceptions

NITException if device is not a device supporting Demosaicing(currently only MC1003 can be demosaiced).

def **setLuminosityGain** (device, value)

Set the value of the luminosity gain involved in the crosstalk compensation process.

Parameters

device The device for which demosaicing must be activated. This device must be color capable.

value to be set. range: [1.0,15.0], default value: 5.0

Exceptions

NITException if device is not a device supporting Demosaicing(currently only MC1003 can be demosaiced).

NITException if value is out of range.

def **getLuminosityGain** (device)

return the current luminosity gain value.

Parameters

device The device for which getLuminosityGain must return a value. This device must be color capable.

Returns

return the current luminosity gain value.

Exceptions

NITException if device is not a device supporting Demosaicing(currently only MC1003 can be demosaiced).

def **setColorSaturation** (device, value)

Set the value of the color saturation involved in the last stage of the demosaicing process.

Parameters

device The device for which demosaicing must be activated. This device must

be color capable.

value to be set. range: [0.1,5.0], default value: 1.0

Exceptions

NITException if device is not a device supporting Demosaicing(currently only MC1003 can be demosaiced).

NITException if value is out of range.

def **getColorSaturation** (device)

return the current color saturation value.

Parameters

device The device for which getColorSaturation must return a value. This device must be color capable.

Returns

return the current color saturation value.

Exceptions

NITException if device is not a device supporting Demosaicing(currently only MC1003 can be demosaiced).

8.4. NITLibrary.Gige Namespace Reference

Namespace of helpers for **Gige** cameras.

Classes

class	**Param**
	base class for **Gige** cameras parameter definition.
class	**IntParam**
	class who define integer parameters for **Gige** cameras.
class	**FloatParam**
	class who define floating point parameters for **Gige** cameras.
class	**BoolParam**
	class who define boolean parameters for **Gige** cameras.
class	**EnumParam**
	class who define enum parameters for **Gige** cameras.
class	**StringParam**
	class who define string parameters for **Gige** cameras.
class	**CommandParam**
	class who define command parameters for **Gige** cameras.

Functions

def **waitBeginOfFrame** (device, timeout)

Function allowing to be informed of the reception of the first packet of a frame.

This is the shortest event we can receive after exposure time.
This function is coupled with **NITDevice.captureNFrames()** and have only a meaning when the number of frames to capture is one.
The timer starts when **NITDevice.captureNFrames()** is called.

```
dev->captureNFrame( 1 ); //Capture 1 frame
dev->waitBeginOfFrame(20);
```

Parameters

device The gige device on which function will be applied.

timeout The maximum number of milliseconds to wait for the reception of the first packet.

Returns

false if the function timed out, else true if the first packet was received.

Exceptions

NITException if the device is not **Gige**

dev **getXml** (device)

Return a string containing the xml of the **Gige** camera.

Parameters

device The gige device on which function will be applied.

Returns

the xml content.

Exceptions

NITException if the device is not **Gige**

dev **openByMacAddress** (mac_address)

Open a discovered **Gige** device by mac address.

When the maintained list is empty, calling this function, will try to discover the connected cameras.
Mac address of the camera can be found on the rear of the camera.
Mac address string must be in the form 70:b3:d5:2a:c0:00 (columns are mandatory).

Parameters

mac_address The mac address of the queried device.

Returns

A pointer to the open device.

Exceptions

NITException if the maintained list is empty.

NITException if the model of the camera cannot be detected.

NITException if no camera with the given mac address exists.

dev **openByIpAddress** (ip_address)

Open a discovered **Gige** device by ip address.

When the maintained list is empty, calling this function, will try to discover the connected cameras.

Parameters

ip_address The ip address of the queried device.

Returns

A pointer to the open device.

Exceptions

NITException if the maintained list is empty.

NITException if the model of the camera cannot be detected.

NITException if no camera with the given ip address exists.

dev **openByUserDefinedName** (user_name)

Open a discovered **Gige** device by user defined name.

When the maintained list is empty, calling this function, will try to discover the connected cameras.

Parameters

user_name The user defined name of the queried device.

Returns

A pointer to the open device.

Exceptions

NITException if the maintained list is empty.

NITException if the model of the camera cannot be detected.

NITException if no camera with the given mac address exists.

dev **forceIp** (mac_address, ip_address, mask, gateway)

Change the ip configuration of the device with the given mac address.

Mac address of the camera can be found on the rear of the camera.
Mac address string must be in the form 70:b3:d5:2a:c0:00 (columns are mandatory).
This function tries to connect to the camera with given mac_address regardless of the discovery of this device.
The setted IP configuration is active until the camera is powered off.

Parameters

mac_address The mac address of the device to update.

ip_address The new IP address.

mask The new subnetwork mask.

gateway The new gateway.

Returns

A pointer to the open device.

Exceptions

NITException if the maintained list is empty.

NITException if the model of the camera cannot be detected.

NITException if no camera with the given mac address exists.

8.4.1.NITLibrary.Gige.IntParam Class Reference

class who define integer parameters for **Gige** cameras.

enum **eRepresentation** { **Linear** , **Logarithmic** , **Boolean** , **PureNumber** , **HexNumber** , **IPV4Address** , **MACAddress** }

Enumerator	
Linear	Slider with linear behavior.
Logarithmic	Slider with logarithmic behaviour.
Boolean	Check box.
PureNumber	Decimal number in an edit control.
HexNumber	Hex number in an edit control.
IPV4Address	IP-Address.
MACAddress	MAC-Address.

enum **eVisibility** { **Beginner** , **Expert** , **Guru** }

Visibility of the parameters.

Enumerator	
Beginner	Visible by all users
Expert	Visible by experts and Gurus
Guru	Visible by Gurus only

def **minValue** ()

return the minimum possible value of the parameter.

def **maxValue** ()

return the maximum possible value of the parameter.

def **step** ()

return the increment step of the parameter.

def **isContinuous** ()

return true if the parameter can take any integer value between minValue and maxValue

def **Representation** ()

return the representation of the parameter. (eRepresentation)

def **name** ()

return the name of the parameter

def **isMutable** ()

return if the parameter can be modified. A parameter is not mutable if it is not writeable or if it can have only one possible value.

def **isReadable** ()

return if the parameter can be read

def **Visibility** ()

return by which kind of user the parameter can be visible. Return an eVisibility enumeration

8.4.2.NITLibrary.Gige.FloatParam Class Reference

class who define floating point parameters for **Gige** cameras.

enum **eVisibility** { **Beginner** , **Expert** , **Guru** }

Visibility of the parameters.

Enumerator	
Beginner	Visible by all users
Expert	Visible by experts and Gurus
Guru	Visible by Gurus only

def **minValue** ()

return the minimum possible value of the parameter.

def **maxValue** ()

return the maximum possible value of the parameter.

def **step** ()

return the increment step of the parameter.

def **isContinuous** ()

return true if the parameter can take any floating point value between minValue and maxValue

def **name** ()

return the name of the parameter

def **isMutable** ()

return if the parameter can be modified. A parameter is not mutable if it is not writeable or if it can have only one possible value.

def **isReadable** ()

return if the parameter can be read

def **Visibility** ()

return by which kind of user the parameter can be visible. Return an eVisibility enumeration

8.4.3.NITLibrary.Gige.BoolParam Class Reference

class who define boolean parameters for **Gige** cameras.

enum **eVisibility** { **Beginner** , **Expert** , **Guru** }

Visibility of the parameters.

Enumerator	
Beginner	Visible by all users
Expert	Visible by experts and Gurus
Guru	Visible by Gurus only

def **name** ()

return the name of the parameter

def **isMutable** ()

return if the parameter can be modified. A parameter is not mutable if it is not writeable or if it can have only one possible value.

def **isReadable** ()

return if the parameter can be read

def **Visibility** ()

return by which kind of user the parameter can be visible. Return an eVisibility enumeration

8.4.4.NITLibrary.Gige.EnumParam Class Reference

class who define enumeration parameters for **Gige** cameras.

enum **eVisibility** { **Beginner** , **Expert** , **Guru** }

Visibility of the parameters.

Enumerator	
Beginner	Visible by all users
Expert	Visible by experts and Gurus
Guru	Visible by Gurus only

def **name** ()

return the name of the parameter

def **isMutable** ()

return if the parameter can be modified. A parameter is not mutable if it is not writeable or if it can have only one possible value.

def **isReadable** ()

return if the parameter can be read

def **Visibility** ()

return by which kind of user the parameter can be visible. Return an eVisibility enumeration

8.4.5.NITLibrary.Gige.StringParam Class Reference

class who define string parameters for **Gige** cameras.

enum **eVisibility** { **Beginner** , **Expert** , **Guru** }

Visibility of the parameters.

Enumerator	
Beginner	Visible by all users
Expert	Visible by experts and Gurus
Guru	Visible by Gurus only

def **name** ()

return the name of the parameter

def **isMutable** ()

return if the parameter can be modified. A parameter is not mutable if it is not writeable or if it can have only one possible value.

def **isReadable** ()

return if the parameter can be read

def **Visibility** ()

return by which kind of user the parameter can be visible. Return an eVisibility enumeration

8.4.6.NITLibrary.Gige.CommandParam Class Reference

class who define command parameters for **Gige** cameras.

Command parameters can only have the value 1.

enum **eVisibility** { **Beginner** , **Expert** , **Guru** }

Visibility of the parameters.

Enumerator	
Beginner	Visible by all users
Expert	Visible by experts and Gurus
Guru	Visible by Gurus only

def **name** ()

return the name of the parameter

def **isMutable** ()

return if the parameter can be modified. A parameter is not mutable if it is not writeable or if it can have only one possible value.

def **isReadable** ()

return if the parameter can be read

def **Visibility** ()

return by which kind of user the parameter can be visible. Return an eVisibility enumeration

8.5. NITLibrary.NITToolBox Namespace Reference

Namespace of the library filters and observers

Classes

class	**NITAddText**
	Overlay a text on the current frame.
class	**NITAGCROI**
	Filter who apply an histogram stretching based on predefined regions of interest.
class	**NITAGCROICustom**
	Filter who apply an histogram stretching based on a region of interest. This filter is not applied on color input frames.
struct	**Roi**
	A simple structure to pass rectangular region of interest to the **NITAGCROI** filter.
class	**NITAutomaticGainControl**
	Filter who applies an histogram stretching based on the full frame. This filter is not applied on color input frames.
class	**NITColorMap**
	Apply a color Lut on grayscale frames.
class	**NITFlip**
	Class to change the orientation of a frame.
class	**NITGainControl**
	Base class for all the histogram stretching classes.
class	**NITGamma**
	Permit to adjust the gamma of the image.
class	**NITImageOverlay**
	Permit to overlay an image on part of the frame.
class	**NITManualGainControl**
	Filter who apply a stretching we a preset min and max value. This filter is not applied on color input frames.
class	**NITPlayer**
	Class to display frames.
class	**NITRecordAvi**
	This class record the frame data in an AVI file.
class	**NITSnapshot**
	This class record frames in image files.
class	**NITTimestamp**

Overlay a date time string on the current frame.

8.5.1.NITLibrary.NITToolBox.NITAddText Class Reference

Overlay a text on the current frame.

The overlay is done at the position and with the font specified. If due to the size and position of the overlay, the overlay is to big to be rendered in the current frame, the overlay is cropped.

def **__init__**(self, text, x_pos, y_pos, font_name, font_size)

Construct a **NITAddText** object.

Parameters

text	string to be overlayed. If empty, no string is displayed.
x_pos	x position of the text (pixels).
y_pos	y position of the text (pixels).
font_name	name of the TrueType font to use(as displayed in any text editor). If the name is not recognized by the system or is not a TrueType font, the default system TrueType font is used. Default font is Arial on Windows. Default font is LiberationMono on Linux (if LiberationMono is missing the best fitting font is chosen)
font_size	size of the font in points.

def **setText** (text)

Change the current text.

Parameters

text string to be overlayed. If text is empty, no text is overlay-ed.

def **setFontName** (font_name)

Change the current font.

Parameters

font_name	name of the TrueType font to use(as displayed in any text editor). If the name is not recognized by the system or is not a TrueType font, the default system TrueType font is used(Arial on Windows).

def **setFont** (font_name, font_size)

Change the current font and size.

Parameters

font_name	name of the TrueType font to use(as displayed in any text editor). If the name is not recognized by the system or is not a TrueType font, the default system TrueType font is used(Arial on Windows).

font_size size of the font in points.

def **setFontSize** (font_size)

Change the current font size.

Parameters

font_size size of the font in points.

def **getText** ()

return the current overlay-ed text.

def **getFontName** ()

return the current font name.

def **getFontSize** ()

return the current font size.

def **setPosition** (x_pos, y_pos)

set the position of the text.

Parameters

x_pos,y_pos x,y position of the upper left corner of the text

def **setPosX** (x_pos)

set the x position.

Parameters

x_pos position of the left side of the text

def **setPosY** (y_pos)

set the y position.

Parameters

y_pos position of the upper side of the text

def **setTransparentBackground** (b_set=true)

activate/deactivate background transparency.

Parameters

b_set if true only the text is displayed.
If false the text is displayed on a white background (default true).

def **getPosition** ()

return the position of the text as tuple [x_pos, y_pos]. More...

def **getPosX** ()

return the x position of the text.

def **getPosY** ()

return the y position of the text.

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**.

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**.

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

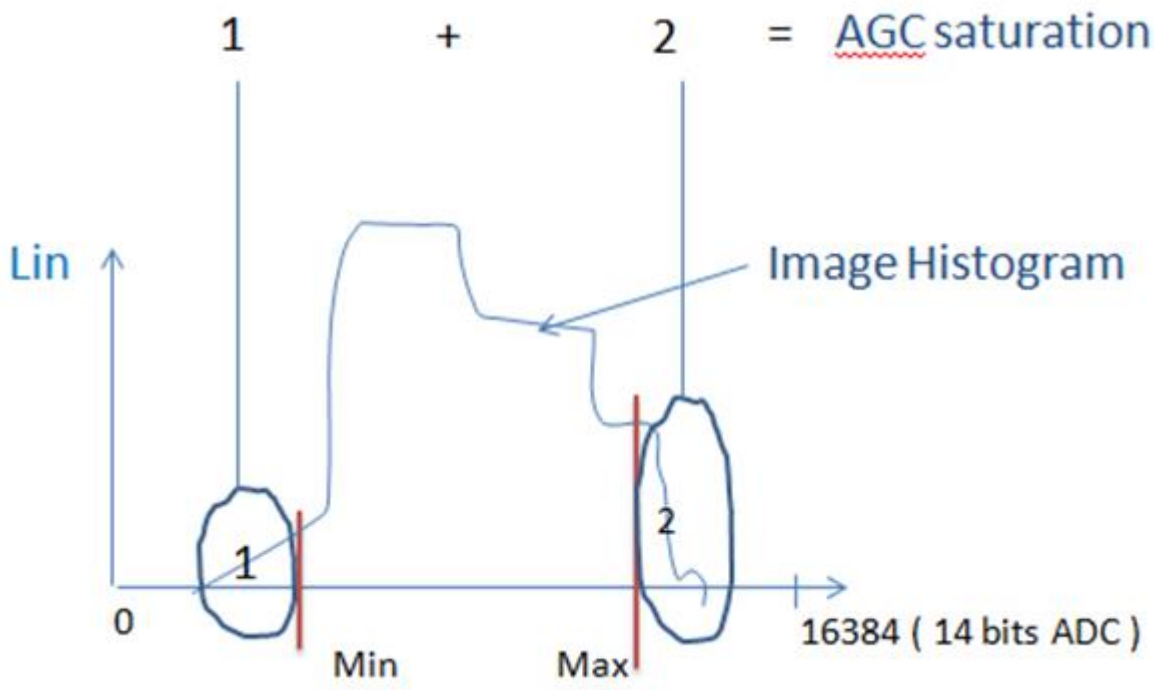
def **active** ()

return the current state of the **NITFilter**

8.5.2.NITLibrary.NITToolBox.NITAGCROI Class Reference

Filter who applies a histogram stretching based on predefined regions of interest.

This filter calculates a histogram on the center half or quarter of a frame.
This filter is not applied on color input frames.
The filter ignores the very first and last pixels having extreme values. It corresponds to the areas marked 1 and 2 in the image below.



Histogram (ignored pixels detail)

You can fine tune these areas by changing the ignored_below and ignored_above parameters.
From the calculated histogram, we extract the corrected min and max value of the image and apply a stretching.
On input the frame must be raw float[0,1] data.
This filter applies only to gray frames.

enum **eRoiType** { **half** , **quarter** }

Enumerator	
half	The histogram is calculated on the center half of the frame
quarter	The histogram is calculated on the center quarter of the frame

def __init__(self, roi_type, ignored_below=200, ignored_above=200)

Construct a **NITAGCROI** object.

Parameters

roi_type	predefined region of interest to use to calculate the histogram.(eRoiType enumeration)
ignored_below,ignored_above	number of pixels to ignore on each side of the histogram to decide the min and max value of the histogram.

def **setRoiType** (roi_type)

Set a new predefined region of interest.

Parameters

roi_type	predefined region of interest to use to calculate the histogram.(eRoiType enumeration)

def **getRoiType** ()

return the current predefined region of interest. (eRoiType enumeration)

def **setIgnoredPixels** (pixels_below, pixels_above)

set the ignored pixels in the calculation process of the min max of the histogram

Parameters

pixels_below,pixels_above	number of pixels to ignore on each side of the histogram to decide the min and max value of the histogram.

def **getIgnoredPixels** ()

return the current ignored pixels as tuple [pixels_below, pixels_above]

def **getMinMaxValue** ()

return the last calculated min and max value of the histogram as tuple [min_value, max_value]

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

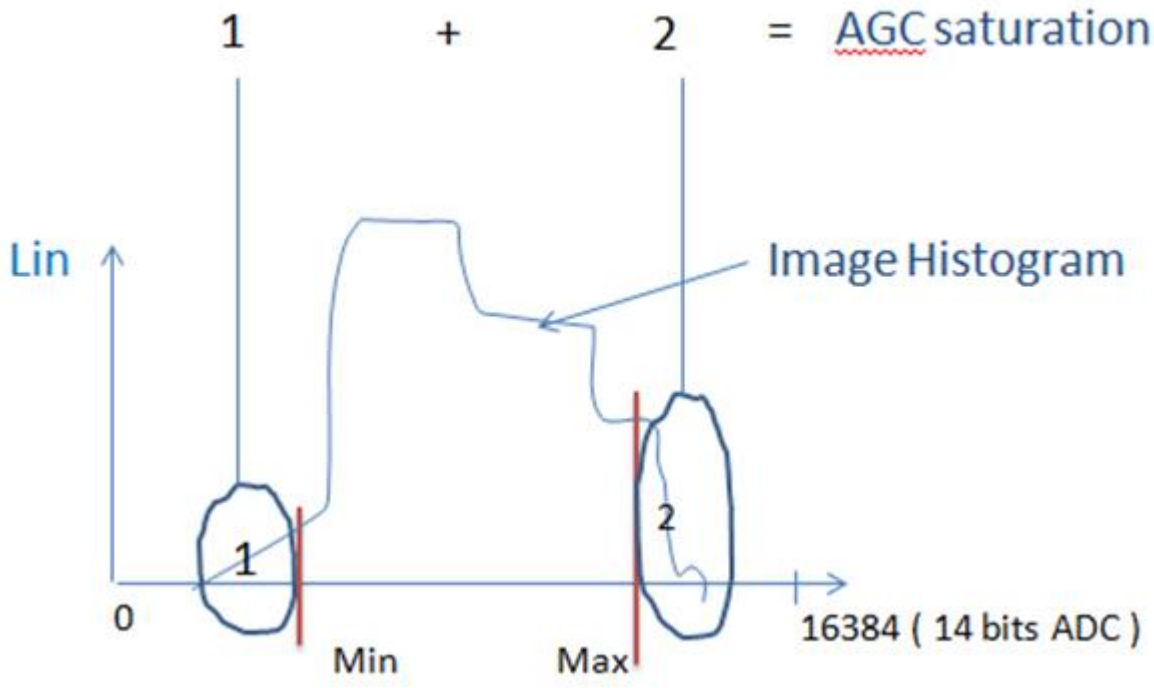
def **active** ()

return the current state of the **NITFilter**

8.5.3.NITLibrary.NITToolBox.NITAGCROICustom Class Reference

Filter who applies a histogram stretching based on a region of interest.

This filter is not applied on color input frames.
The filter ignores the very first and last pixels having extreme values. It corresponds to the areas marked 1 and 2 in the image below.



Histogram (ignored pixels detail)

You can fine tune these areas by changing the ignored_below and ignored_above parameters.

From the calculated histogram, we extract the corrected min and max value of the image and apply a stretching.
On input the frame must be raw float[0,1] data.
This filter applies only to gray frames.

def __init__(self, roi, ignored_below=200, ignored_above=200)

Construct a **NITAGCROICustom** object.

Parameters

roi	region of interest to use to calculate the histogram.(Roi object)
ignored_below,ignored_above	number of pixels to ignore on each side of the histogram to decide the min and max value of the histogram.

def **setRoi** (roi)

Set a new region of interest.

Parameters

roi region of interest to use to calculate the histogram.(Roi object)

def **getRoi** ()

return the current region of interest. (Roi object)

def **setIgnoredPixels** (pixels_below, pixels_above)

set the ignored pixels in the calculation process of the min max of the histogram

Parameters

pixels_below,pixels_above	number of pixels to ignore on each side of the histogram to decide the min and max value of the histogram.

def **getIgnoredPixels** ()

return the current ignored pixels as tuple [pixels_below, pixels_above]

def **getMinMaxValue** ()

return the last calculated min and max value of the histogram as tuple [min_value, max_value]

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**

8.5.4.NITLibrary.NITToolBox.Roi Struct Reference

A simple structure to pass rectangular region of interest to the **NITAGCROI** filter.

def __**init**__ (self)

default constructor. All coordinates are set to 0

def __**init**__ (self, **xLeft**, **xRight**, **yTop**, **yBottom**)

constructor.

Parameters

xLeft Left coordinate
xRight Right coordinate
yTop Top coordinate
yBottom Bottom coordinate

def **empty** ()

return the emptiness of the ROI.
An empty ROI has his width and/or height set to 0.

Returns
true if empty or false otherwise

def **valid** ()

return if the ROI is valid An valid ROI has at minimum one of his coordinates not 0

Returns
true if valid or false otherwise

self.**xLeft**

self.**xRight**

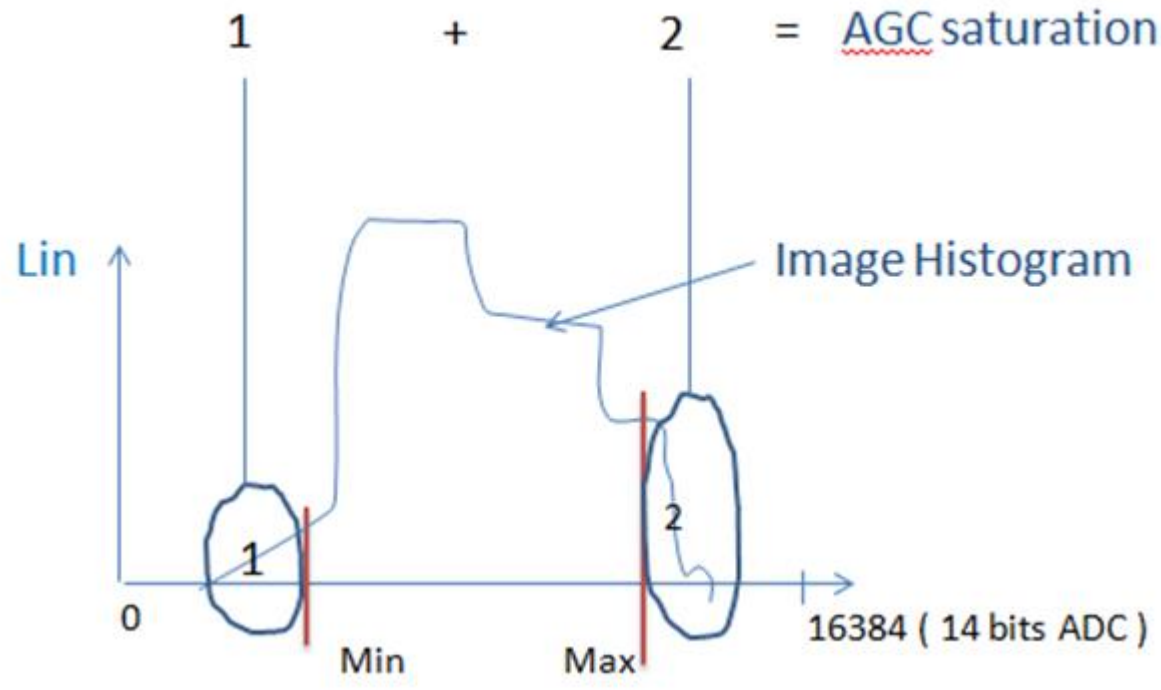
self.**yTop**

self.**yBottom**

8.5.5.NITLibrary.NITToolBox.NITAutomaticGainControl Class Reference

Filter who applies a histogram stretching based on the full frame.

This filter is not applied on color input frames.
The filter ignores the very first and last pixels having extreme values. It corresponds to the areas marked 1 and 2 in the image below.



Histogram (ignored pixels detail)

You can fine tune these areas by changing the ignored_below and ignored_above parameters.
From the calculated histogram, we extract the corrected min and max value of the image and apply a stretching.
On input the frame must be raw float[0,1] data.
This filter applies only to gray frames.

def __init__(self, ignored_below=200, ignored_above=200)

Construct a **NITAutomaticGainControl** object.

Parameters

ignored_below,ignored_above number of pixels to ignore on each side of the histogram to decide the min and max value of the histogram.

def **setIgnoredPixels** (pixels_below, pixels_above)

set the ignored pixels in the calculation process of the min max of the histogram

Parameters

pixels_below,pixels_above number of pixels to ignore on each side of the histogram to decide the min and max value of the histogram.

def **getIgnoredPixels** ()

return the current ignored pixels as tuple [pixels_below, pixels_above]

def **getMinMaxValue** ()

return the last calculated min and max value of the histogram as tuple [min_value, max_value]

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**

8.5.6.NITLibrary.NITToolBox.NITColorMap Class Reference

Apply a color Lut on grayscale frames.

enum { **NONE** , **AUTUMN** , **BONE** , **JET** , **WINTER** , **RAINBOW** , **OCEAN** , **SUMMER** , **SPRING** , **COOL** , **HSV** , **PINK** , **HOT** , **NIGHT_VISION** }

color map enumeration

Enumerator	
NONE	no colormap applied
AUTUMN	autumn colormap
BONE	bone colormap
JET	jet colormap
WINTER	winter colormap
RAINBOW	rainbow colormap
OCEAN	ocean colormap
SUMMER	summer colormap
SPRING	spring colormap
COOL	cool colormap
HSV	hsv colormap
PINK	pink colormap
HOT	hot colormap
NIGHT_VISION	night vision colormap

def __init__ (self, color_map=**NONE**)

Construct a **NITColorMap** object.

Parameters

color_map index of the color Lut to apply.
default value 0

def **setColorMap** (new_color_map)

Change the color Lut.

Parameters

new_color_map index of the color Lut to apply

Exceptions

NITException if the color_map index is out of range

def **getColorMap** ()

return the current color map index

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**

8.5.7.NITLibrary.NITToolBox.NITFlip Class Reference

Class to change the orientation of a frame.

enum **eFlip** { NONE, HORZ, VERT, BOTH }

Enumerator	
NONE	No flip
HORZ	Flip the frame horizontally
VERT	Flip the frame vertically
BOTH	Flip the frame in both directions

def __**init**__ (self, direction)

Contruct a **NITFlip** object.

Parameters

direction direction of the flip(eFlip enumeration)

def **setFlip** (direction)

Change the flip direction.

Parameters

direction new direction of the flip (eFlip enumeration)

def **getFlip** ()

Return the flip direction.

Returns

the current flip direction (eFlip enumeration)

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**.

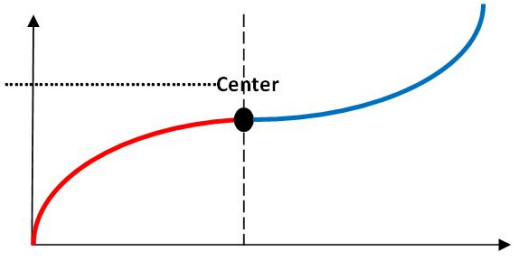
8.5.8.NITLibrary.NITToolBox.NITGamma Class Reference

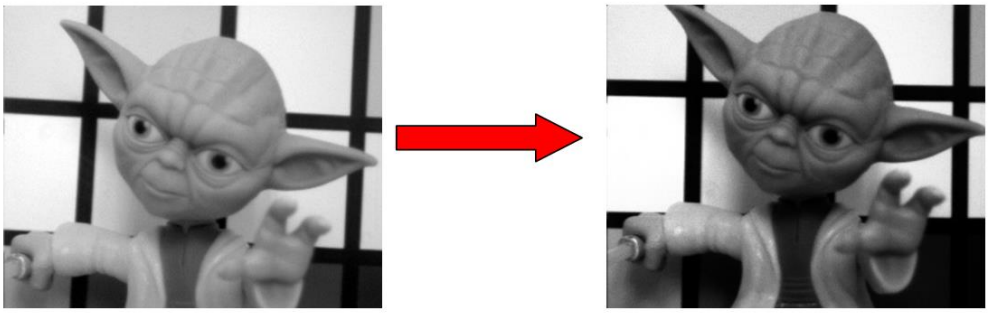
Permit to adjust the gamma of the image.

See also https://en.wikipedia.org/wiki/Gamma_correction

Gamma correction is only supported on gray scale frames.

The gamma curve is described as an "S-Curve" as below: You can modify 2 parameters: the gamma factor and the center of the curve. If the value is less than the center the LUT curve in red is applied and if it is greater than the center value, the LUT curve in blue is applied.





def __**init**__ (self, new_center=0.5, new_gamma=0.5)

Construct a **NITGamma** object.

Parameters

new_center center value in the range [0.0, 1.0]
default value 0.5

new_gamma gamma value in the range [0.1,3.0]
default value 0.5

Exceptions

NITException if new_center or new_gamma are out of range

def **setCenter** (new_center)

set a new center value

Parameters

new_center center value in the range [0.0, 1.0]

Exceptions

NITException if new_center is out of range

def **getCenter** ()

return the current center value

def **setGamma** (new_gamma)

set a new gamma value

Parameters

new_gamma gamma value in the range [0.0, 3.0]

Exceptions

NITException if new_gamma is out of range

def **getGamma** ()

return the current gamma value

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**

8.5.9.NITLibrary.NITToolBox.NITImageOverlay Class Reference

Permit to overlay an image on part of the frame.

__init__ (self, image_file, x_pos, y_pos, opacity=1.0)

Construct a **NITImageOverlay** object.

Currently, the supported image formats are "png", "jpeg", "tiff" and "bmp".
The supported pixel formats in the file are gray(0,255), gray(0.0,1.0), rgb and rgba.
The overlay-ed image is converted to the same format as the frame format.
A transparent color can be applied. In this case, all the pixels in the overlay-ed image with this color are not copied to the frame.
The image is clipped to the boundaries of the frame.

Parameters

image_file The full path to the image to overlay.

x_pos,y_pos The upper left corner of the position of the overlay-ed image in the frame.

opacity The opacity level range [0.0,1.0].
default: 1.0 (no transparency)

Exceptions

NITException if the file doesn't exist.

NITException if the file format is not supported.

NITException if the opacity is out of range.

def **setImageFile** (new_image_file, x_pos, y_pos, opacity=1.0)

Change the overlay-ed image, position and opacity.

Parameters

new_image_file The full path to the image to overlay.

x_pos,y_pos The upper left corner of the position of the overlay-ed image in the frame.

opacity The opacity level range [0.0,1.0].
default: 1.0 (no transparency)

Exceptions

NITException if the file doesn't exist.

NITException if the file format is not supported.

NITException if the opacity is out of range.

def **setImageFile** (new_image_file)

Change the overlay-ed image.

Parameters

new_image_file The full path to the image to overlay.

Exceptions

NITException if the file doesn't exist.

NITException if the file format is not supported.

def **getImageFile** ()

return the overlay-ed image full path

def **setPosition** (x_pos, y_pos)

Change the overlay-ed image position.

Parameters

x_pos,y_pos The upper left corner of the position of the overlay-ed image in the frame.

Exceptions

NITException if the file format is not supported.

NITException if the opacity is out of range.

def **setPosX** (x_pos)

Change the overlay-ed image X position.

Parameters

x_pos The left side of the position of the overlay-ed image in the frame.

def **setPosY** (y_pos)

Change the overlayed image Y position.

Parameters

y_pos The upper side of the position of the overlay-ed image in the frame.

def **setOpacity** (new_opacity)

Change the overlay-ed image opacity.

Parameters

new_opacity The opacity level range [0.0,1.0].

Exceptions

NITException if the opacity is out of range.

def **setTransparentColor** (r, g, b)

Set the color to use as transparency color (color version)

All the pixels in the overlay-ed image with this color are not copied to the frame.
If the frame is gray, the transparent color is converted to the range [0.0, 1.0] with this formula:
transparent_gray = (((float)(r + g + b))/3) / 255

Parameters

r,g,b red, green, and blue value of the transparency color in the range [0,255]

def **setTransparentColor** (gray)

Set the color to use as transparency color.

All the pixels in the overlay-ed image with this color are not copied to the frame.
If the frame is rgba, the transparent color is converted to color in the range [0, 255] with this formula:
*transparent_color = [gray*255, gray*255, gray*255, 0]*

Parameters

gray value of the transparency color in the range [0.0,1.0]

def **enableTransparentColor** (enable)

Enable/Disable the use of the transparency color.

Parameters

enable If true the transparency color is applied, else not

def **getPosition** ()

Return the position of the overlay-ed image as a tuple [posX, posy]

def **getPosX** ()

return the X position of the overlay-ed image.

def **getPosY** ()

return the Y position of the overlay-ed image.

def **getOpacity** ()

return the current opacity value.

def **isTransparentColorEnabled** ()

return if the transparent color is enabled.

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**.

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**.

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**

8.5.10. NITLibrary.NITToolBox.NITManualGainControl Class Reference

Filter who applies a stretching with a pre-settled min and max value.

This filter is not applied on color input frames.
Contrary to **NITAutomaticGainControl** and derived, the stretching is done with fixed values.
No histogram calculation is done.

def **__init__** (self, low_limit, high_limit)

Construct a **NITManualGainControl** object.

Parameters

low_limit,high_limit min value and max value used to produce the stretching.

def **setMinMaxValue** (low_limit, high_limit)

Set new values to compute the stretching.

Parameters

low_limit,high_limit min value and max value used to produce the stretching.

def **getMinMaxValue** ()

return the last calculated min and max value of the histogram as tuple [low_limit,high_limit]

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**

8.5.11. NITLibrary.NITToolBox.NITPlayer Class Reference

Class to display frames.

This observer display the frames in a window.

def __**init**__ (self, title)

Construct a **NITPlayer** object.

Parameters

title Title of the window

def **disconnect** ()

disconnect this observer from the pipeline

8.5.12. NITLibrary.NITToolBox.NITRecordAvi Class Reference

This class record the frame data in an AVI file.

The data is recorded in MP4 format.

The frames are recorded when **startRecording()** is called. The recording continue until **stopRecording()**, **reset()** or **close()** is called or the object is destructed.

def __**init**__ (self)

Construct an uninitialized **NITRecordAvi** object.

If this constructor is called, you must call reset.
If not, the calls to the object functions will be no-ops.

def __**init**__ (self, output_filename, avi_frame_width, avi_frame_height, fps)

Construct a **NITRecordAvi** object.

Parameters

output_filename	The full path to the avi file to create
avi_frame_width,avi_frame_height	Resolution of the recorded frame. If the resolution is lower than the frame resolution, the frame is cropped. If the resolution his higher than the frame resolution, the frame is displayed in top-left side of the avi image.
fps	frame rate at which the recording is done.

Exceptions

NITException if the file cannot be open, or the extension is not avi.

def **reset** (output_filename, avi_frame_width, avi_frame_height, fps)

Reset the current recording.

If this function is called while a recording is started; the recording is stopped immediatly, remaining frames to record are dropped and the file is closed.
If output_filename is the same name as the current file name, the file is rewritten. That is the old data is lost.

Parameters

output_filename	The full path to the avi file to create
avi_frame_width,avi_frame_height	Resolution of the recorded frame. If the resolution is lower than the frame resolution, the frame is cropped. If the resolution his higher than the frame resolution, the frame is displayed in top-left side of the avi image.
fps	frame rate at which the recording is done.

Exceptions

NITException if the file cannot be open. or the extension is not avi.

def **startRecording** ()

Start recording in the current file.

def **stopRecording** (record_last_frames=true)

Stop recording in the current file.

The file is not closed. A new call to **startRecording()** will continue the recording on the current file.

Parameters

record_last_frames	if set to true no new frames are taken into account, but frames already enqueued are honored if set to false enqueued frames are dropped

Exceptions

NITException if the file is currently closed.

def **close** (record_last_frames=true)

Close the current file.

If recording is in progress, recording is first stopped.

Parameters

record_last_frames if set to true no new frames are taken into account, but frames already enqueued are honored
if set to false enqueued frames are dropped

To restart a new recording session you must call **reset()**.

def **getFileName** ()

Return the current full file path.

Returns
the current full file path.

def **aviFrameWidth** ()

Return the current recorded frame width.

This parameter is the width of the recorded image, not the width of the NITFrames who can vary

Returns
the current recorded frame width.

def **aviFrameHeight** ()

Return the current recorded frame height.

This parameter is the height of the recorded image, not the height of the NITFrames who can vary

Returns
the current recorded frame height.

def **isRecording** ()

Return the current recording state.

Returns
true if recording is in progress or false otherwise.

def **isOpen** ()

Return the current file state.

Returns
true if the file is open or false otherwise.

def **disconnect** ()

disconnect this observer from the pipeline

8.5.13. NITLibrary.NITToolBox.NITSnapshot Class Reference

This class record frames in image files.

The supported formats are bmp, jpg, png, tif, with pixel type rgb

The file names have the following form: <directory>/<prefix>_<counter>.<extension>
With <directory>,<prefix>,<counter>,<extension> parameters who can be set.
The function **setCounter()** permits to set the number of digits of the counter. The default value is 0; in this case counter is written without trailing 0.
Example: if counter == 10 and number of digits is 0: The file name look like this: <directory>/<prefix>_10.<extension>
Example: if counter == 10 and number of digits is 4: The file name look like this: <directory>/<prefix>_0010.<extension>

The number of frames recorded are set with the function **snap()**.

def __**init**__(self)

Construct an uninitialized **NITSnapshot** object.

If this constructor is called, you must call reset and, if needed, setJpegQuality.
If not, the calls to the object functions will be no-ops.

def __**init**__ (self, file_directory, prefix, extension, jpeg_quality=95)

Construct a **NITSnapshot** object.

Parameters

file_directory	The existing writable directory where the files must be created.
prefix	Prefix of the file.
extension	Extension of the file. The extension define the image format. extension can be "bmp", "jpg", "jpeg", "png", "tif", "tiff". A dot can precede the name.
jpeg_quality	quality of the jpeg file(uniquely used for jpeg).

Exceptions

NITException	if the directory don't exists.
NITException	if the extension is not supported.
NITException	if the jpeg_quality is above 100.

def **reset** (file_directory, prefix, extension)

Set new file components.

Parameters

file_directory	The existing writable directory where the files must be created.
prefix	Prefix of the file.
extension	Extension of the file. The extension define the image format. extension can be "bmp", "jpg", "jpeg", "png", "tif", "tiff". A dot can

precede the name.

Exceptions

NITException if the directory don't exists.

NITException if the extension is not supported.

def **setJpegQuality** (jpeg_quality)

Set new jpeg quality.

If the image to record is not jpeg, the parameter is ignored.

Parameters

jpeg_quality quality of the jpeg file(uniquely used for jpeg).

Exceptions

NITException if the jpeg_quality is above 100.

def **setCounter** (new_counter, digits)

Set new counter start value.

The counter is the value used to construct the name of the file.
Each time a new file is created this value is incremented by one.
The default value at object creation is 1.

Parameters

new_counter The new value of the counter.

digits number of digits used to convert the counter to string in the file name creation process.

def **getLastFileName** ()

Return the full file name of the last recorded image.

def **getCounterValue** ()

Return the current counter value.

def **snap** (count)

Set a number of images to record.

At creation of the object the current snap count is 0, that is no frame is recorded.
Each time a frame is recorded the snap count is decremented. When the count reach 0, the recording is stopped.

The parameter of this function is added to the current snap count.

Parameters

count Number of images to record.

def **disconnect** ()

disconnect this observer from the pipeline

8.5.14. NITLibrary.NITToolBox.NITTimestamp Class Reference

Overlay a date time string on the current frame.

The date time is format is: *YYYYMMDDTHHMMSS,ffffffff* .
Example: 20200220T125030,562123 and: *dd-mm-yyyy HH.MM.SS:mks* with mks milliseconds on three digits on Windows.
Example: 20_02-2020 12:50:30:562

The overlay is done at the position specified and with the font specified.
If due to the size and position of the overlay, the overlay is to big to be rendered in the current frame, the overlay is cropped.

def __**init**__ (x_pos, y_pos, font_name, font_size)

Construct a **NITTimestamp** object.

Parameters

x_pos x position of the text.

y_pos y position of the text.

font_name name of the TrueType font to use(as displayed in any text editor). If the name is not recognized by the system or is not a TrueType font, the default system TrueType font is used. Default font is Arial on Windows.
Default font is LiberationMono on Linux (if LiberationMono is missing the best fitting font is chosen)

font_size size of the font in points.

def **setFontName** (font_name)

Change the current font.

Parameters

font_name name of the TrueType font to use(as displayed in any text editor). If the name is not recognized by the system or is not a TrueType font, the default system TrueType font is used(Arial on Windows).

def **setFont** (font_name, font_size)

Change the current font and size.

Parameters

font_name name of the TrueType font to use(as displayed in any text editor). If the name is not recognized by the system or is not a TrueType font, the default system TrueType font is used(Arial on Windows).

font_size size of the font in points.

def **setFontSize** (font_size)

Change the current font size.

Parameters

font_size size of the font in points.

def **getFontName** ()

return the current font name.

def **getFontSize** ()

return the current font size.

def **setPosition** (x_pos, y_pos)

set the position of the text.

Parameters

x_pos,y_pos x,y position of the upper left corner of the text

def **setPosX** (x_pos)

set the x position.

Parameters

x_pos position of the left side of the text

def **setPosY** (y_pos)

set the y position.

Parameters

y_pos position of the upper side of the text

def **setTransparentBackground** (b_set=true)

activate/deactivate background transparency.

Parameters

b_set if true only the text is displayed. If false the text is displayed on a white background (default true).

def **getPosition** ()

return the position of the text as tuple [x_pos, y_pos]

def **getPosX** ()

return the x position of the text.

def **getPosY** ()

return the y position of the text.

def **operator<<** (next_filter)

operator to connect a **NITFilter** after this **NITFilter**.

def **operator<<** (next_observer)

operator to connect a **NITObserver** after this **NITFilter**.

def **disconnect** ()

disconnect this **NITFilter** from the pipeline

def **activate** (state)

change the state of the **NITFilter**

Parameters

state if true the **NITFilter** is active, else the frames are directly passed to the next layer without processing.

def **active** ()

return the current state of the **NITFilter**

9. MC-1003 parameters

NITLibrary 3.x series support MC-1003 cameras for USB3 with USB firmware version above or equal to 2.0.0.

The following parameters are supported.

Parameters

Number of Lines From 4 to 1024 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 1280 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 1020 by steps of 4.
(alias OffsetY can be used instead)

First Column From 0 to 1276 by steps of 4.
(alias OffsetX can be used instead)

This parameters permit to set and retrieve the ROI of the camera.
Use the unsigned int form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 1000 );
```

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 1024.
- NITException if the sum 'Number of Columns' + 'First Column' is above 1280.

Mode Capture mode of the camera.
Use the string form of setParamValueOf to set this parameter.
The following capture modes are available: { "Global Shutter", "Rolling", "Differential" }.

- Global Shutter : The entire frame is exposed and

captured at the same instant
see https://en.wikipedia.org/wiki/Rolling_shutter.
- Rolling : The frame is exposed and captured line by line see https://en.wikipedia.org/wiki/Rolling_shutter.
- Differential : The output frame is the difference of two succesive frames in Global Shutter.

When mode is 'Rolling' or 'Differential' Exposure Time is fixed and equal to the frame period.

```
Example: dev.setParamValueOf( "Mode", "Rolling" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Mode'.
- NITConfigObserver.paramRangeChanged 'Exposure Time' if needed.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw: NITException if the string value or the index value doesn't exist.

Pixel Clock

Use the double form of setParamValueOf to set this parameters in numeric format.
The available pixel clocks for this camera are in MHz: { 12.5, 16.66, 20, 25, 33, 40, 50, 66.66, 80 }

```
Example: dev.setParamValueOf( "Pixel Clock", 50.0 );
```

You can also set this values as string: { "12.5MHz", "16.66MHz", "20MHz", "25MHz", "33MHz", "40MHz", "50MHz", "66.66MHz", "80MHz" }
Well formed string must follow this pattern <double value>< zero or more spaces>< mhz in all combinaison of lower/uppercase or nothing >.
Examples: 12.5, 12.5Mhz, 12.5 MHZ, 12.5mhz

```
Example: dev.setParamValueOf( "Pixel Clock", "50MHz" );
```

Emit:

- NITConfigObserver.paramChanged 'Pixel Clock'.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the double value doesn't exist.
- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.

Exposure Time

The camera can only use discrete values as Exposure Time.
If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
Use the double form of setParamValueOf to set this parameters in numeric format.
The available exposure times for this camera are in microseconds: from 100 to 25600 by steps of 100 in Global Shutter mode.
In Rolling and Differential modes, exposure time is fixed by the frame rate.

```
    Example: dev.setParamValueOf( "ExposureTime", 500.0 );
```

You can also set this values as string: Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
    Example: dev.setParamValueOf( "exposure Time", "2500us" );
                dev.setParamValueOf( "exposure time", "2.5ms"
);
```

The value set is rounded to the nearest available value.

```
  If you call dev.setParamValueOf( "exposure Time", "110us" );
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time' with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

Use the string form of setParamValueOf to set this parameters.
The following trigger modes are available: { "Disabled", "Input", "Output" }.

- Disabled : The camera is free running.

- Input : The camera outputs a frame on each rising edge of the trigger input.
- Output : The camera outputs a high signal each time a frame is outputted.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.

Throw:

- NITException if the string value or the index value don't exist.

10. WiDySWIR 640V-S series parameters

NITLibrary 3.x series support WiDySWIR 640V-S cameras series for USB3 with USB firmware version above or equal to 2.0.0.

The following parameters are supported:

Parameters

Number of Lines From 4 to 512 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 640 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 508 by steps of 4.
(alias "OffsetX" can be used instead)

First Column From 0 to 636 by steps of 4.
(alias "OffsetY" can be used instead)

These parameters allow setting and retrieving the ROI of the camera.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 600 );
```

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 512.
- NITException if the sum 'Number of Columns' + 'First Column' is above 640.

Mode Capture mode of the camera.
This parameter is writeable only for WiDySWIR 640V-SP and WiDySWIR 640V-STP cameras.
WiDySWIR 640V-S and WiDySWIR 640V-ST cameras have only Global Shutter mode.
Use the string form of setParamValueOf to set this parameter.
For WiDySWIR 640V-SP and WiDySWIR 640V-STP cameras, the following capture modes are available: {

"Global Shutter", "Gated" }.

- Global Shutter : The entire frame is exposed and captured at the same instant see https://en.wikipedia.org/wiki/Rolling_shutter.
- Gated : Global shutter mode with very short Exposure Times.

```
Example: dev.setParamValueOf( "Mode", "Global Shutter" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Mode", 0 );
```

Emit:

- NITConfigObserver.paramChanged 'Mode'.
- NITConfigObserver.paramRangeChanged 'Exposure Time' if needed.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Pixel Clock

The available pixel clocks for this camera are in MHz: { 25, 66.66 }

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "Pixel Clock", 25.0 );
```

You can also set this values as string: { "25MHz", "66.66MHz" }
Well formed string must follow this pattern <double value>< zero or more spaces>< mhz in all combinaison of lower/uppercase or nothing >.
Examples: 25.0, 25Mhz, 25.0 MHZ, 25mhz

```
Example: dev.setParamValueOf( "Pixel Clock", "25MHz" );
```

Emit:

- NITConfigObserver.paramChanged 'Pixel Clock'.
- NITConfigObserver.RangeChanged 'Exposure Time'.
- NITConfigObserver.fpsRangeChanged if needed.

- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the double value doesn't exist.
- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.

Exposure Time

The camera can only use discrete values as Exposure Time. If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
The available exposure times for this camera are dependant of capture mode:
Global Shutter: (values are in microsecond)

Expo Min	Expo Max	Step
100	25000	100
30000	70000	20000
100000	200000	50000

If Trigger Mode is set to External and Exposure Time is set to 200000, the camera uses the trigger width as Exposure Time.

Gated: (values are in microseconds) 0.2, 0.4, 1.0, 2.0, 5.0, 10.0, 15.0, 40.0

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "ExposureTime", 500.0 );
```

You can also set this values as string:
Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
Example: dev.setParamValueOf( "exposure Time", "2500us" );
         dev.setParamValueOf( "exposure time", "2.5ms" );
```

The value set is rounded to the nearest available value.

```
 If you call dev.setParamValueOf( "exposure Time", "110us" );
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time'

with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

The following trigger modes are available: { "Disabled", "Input", "Output" }.

- Disabled : The camera is free running.
- Input : The camera outputs a frame on each rising edge of the trigger input.
- Output : The camera outputs a high signal each time a frame is outputted.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Trigger Delay Input

(only available in "Gated" Mode) This parameter is available only for WiDySWIR 640V-SP and WiDySWIR 640V-STP cameras.
When "Trigger Mode" is set to "Input", a trigger delay can be applied.
This is the delay between the rising edge of the trigger input and the beginning of exposure.
The available trigger delays are in microseconds: from 0.2

to 20.6 by steps of 0.08 in "Gated" mode only.
Default value 0.2.
If Mode is "Global Shutter" value is always 0.0.
Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Input", 0.2 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Input", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Input'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Trigger Delay Output (only available in "Gated" Mode) This parameter is available only for WiDySWIR 640V-SP and WiDySWIR 640V-STP cameras.
When "Trigger Mode" is set to "Output", a trigger delay can be applied.
This is the delay between the rising edge of the trigger output and the beginning of exposure.
The available trigger delays are in microseconds: from -10.0 to 10.4 by steps of 0.08 in "Gated" mode only.
Default value -10.0.
If Mode is "Global Shutter" value is always 0.0.
Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Output", 9.6 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Output", "9.6"
);
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Output'.

Throw:

- NITException if the string value or numeric value

doesn't exist.

Tec Mode

This parameter is available only for WiDySWIR 640V-ST and WiDySWIR 640V-STP cameras.
Permit to activate the regulation of the camera temperature.
The parameter "Temperature TEC" permit to set the target temperature.
The available Modes are: "No Current", "Low Current", "Medium Current" and "Strong Current".
A higher current leads to faster reaching of the target temperature.
"No Current" mean no regulation.

Tec Mode	Regulation Current	Comment
Low Current	40mA	camera can be powered by an USB2 connector
Medium Current	150mA	camera can be powered by an USB3 connector
Strong Current	400mA	camera **must** be powered by an external power supply

Using medium current mode with camera powered by an USB2 connector can damage your host PC.
Using strong current mode with camera powered by an USB3 connector can damage your host PC.
NIT is not liable of any damage incurred by misuse of NIT cameras.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Tec Mode", "Low Current" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Tec Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Mode'.
- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Temperature Tec

This parameter is available only for WiDySWIR 640V-ST and WiDySWIR 640V-STP cameras.
The available target temperatures range from -5°C to 70°C by steps of 5°C.
Use the numeric(double) form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "Temperature Tec", -5.0 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Temperature Tec", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

11. WiDySWIR S320V-S parameters

NITLibrary 3.x series support WiDySWIR S320V-S cameras for USB3 with USB firmware version above or equal to 2.0.0.

The following parameters are supported:

Parameters

Number of Lines From 4 to 256 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 320 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 252 by steps of 4.
(alias "OffsetX" can be used instead)

First Column From 0 to 316 by steps of 4.
(alias "OffsetY" can be used instead)

These parameters permit to set and retrieve the ROI of the camera.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 304 );
```

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 256.
- NITException if the sum 'Number of Columns' + 'First Column' is above 320.

Mode Capture mode of the camera.
One capture mode is available.

- Global Shutter : The entire frame is exposed and captured at the same instant
see https://en.wikipedia.org/wiki/Rolling_shutter.
As only one capture mode is available, this

parameter is read only.

Throw:

- NITException if you try to set this parameter.

Pixel Clock

The available pixel clocks for this camera are in MHz: { 25, 66.66 }

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "Pixel Clock", 25.0 );
```

You can also set this values as string: { "25MHz", "66.66MHz" }
Well formed string must follow this pattern <double value>< zero or more spaces>< mhz in all combinaison of lower/uppercase or nothing >.
Examples: 25.0, 25Mhz, 25.0 MHZ, 25mhz

```
Example: dev.setParamValueOf( "Pixel Clock", "25MHz" );
```

Emit:

- NITConfigObserver.paramChanged 'Pixel Clock'.
- NITConfigObserver.RangeChanged 'Exposure Time'.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the double value doesn't exist.
- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.

Exposure Time

The camera can only use discrete values as Exposure Time. If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
The available exposure times for this camera are dependant of capture mode:
Global Shutter: (values are in microsecond)

Expo Min	Expo Max	Step
100	25000	100
30000	70000	20000
100000	200000	50000

If Trigger Mode is set to External and Exposure Time is set

to 200000, the camera uses the trigger width as Exposure Time.

Gated: (values are in microseconds) 0.2, 0.4, 1.0, 2.0, 5.0, 10.0, 40.0

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "ExposureTime", 500.0 );
```

You can also set this values as string:
Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
Example: dev.setParamValueOf( "exposure Time", "2500us" );
         dev.setParamValueOf( "exposure time", "2.5ms" );
```

The value set is rounded to the nearest available value.

```
If you call dev.setParamValueOf( "exposure Time", "110us" );
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time' with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

The following trigger modes are available: { "Disabled", "Input", "Output" }.

- Disabled : The camera is free running.
- Input : The camera outputs a frame on each rising edge of the trigger input.
- Output : The camera outputs a high signal each time a frame is outputted.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this

case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.

Throw:

- NITException if the string value or the index value doesn't exist.

12. WiDySWIR 320V-S series parameters

NITLibrary 3.x series support WiDySWIR S320V-S series cameras for USB3 with USB firmware version above or equal to 2.0.0.

The following parameters are supported:

Parameters

Number of Lines From 4 to 256 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 320 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 252 by steps of 4.
(alias "OffsetX" can be used instead)

First Column From 0 to 316 by steps of 4. (alias "OffsetY" can be used instead)

These parameters permit to set and retrieve the ROI of the camera.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 304 );
```

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 256.
- NITException if the sum 'Number of Columns' + 'First Column' is above 320.

Mode Capture mode of the camera.
This parameter is writeable only for WiDySWIR 320V-SP cameras.
WiDySWIR 320V-S camera has only Global Shutter mode.
Use the string form of setParamValueOf to set this parameter.
For WiDySWIR 320V-SP camera, the following capture modes are available: { "Global Shutter", "Gated" }.

- Global Shutter : The entire frame is exposed and captured at the same instant
  see https://en.wikipedia.org/wiki/Rolling_shutter.
- Gated : Global shutter mode with very short Exposure Times.

```
Example: dev.setParamValueOf( "Mode", "Global Shutter" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Mode", 0 );
```

Emit:

- NITConfigObserver.paramChanged 'Mode'.
- NITConfigObserver.paramRangeChanged 'Exposure Time' if needed.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Pixel Clock

The available pixel clocks for this camera are in MHz: { 12.5, 25.0 }

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "Pixel Clock", 25.0 );
```

You can also set this values as string: { "12.5MHz", "25MHz" }
Well formed string must follow this pattern <double value>< zero or more spaces>< mhz in all combinaison of lower/uppercase or nothing >.
Examples: 25.0, 25Mhz, 25.0 MHZ, 25mhz

```
Example: dev.setParamValueOf( "Pixel Clock", “25MHz” );
```

Emit:

- NITConfigObserver.paramChanged 'Pixel Clock'.
- NITConfigObserver.RangeChanged 'Exposure Time'.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the double value doesn't exist.
- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.

Exposure Time

The camera can only use discrete values as Exposure Time. If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
The available exposure times for this camera are dependant of capture mode:
Global Shutter: (values are in microsecond)

Expo Min	Expo Max	Step
100	25000	100
30000	70000	20000
100000	200000	50000

If Trigger Mode is set to External and Exposure Time is set to 200000, the camera use the trigger width as Exposure Time.

Gated: (values are in microsecond) 0.2, 0.4, 1.0, 2.0, 5.0, 10.0, 15.0, 40.0

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "ExposureTime", 500.0 );
```

You can also set this values as string:
Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
Example: dev.setParamValueOf( "exposure Time", "2500us" );
         dev.setParamValueOf( "exposure time", "2.5ms" );
```

The value set is rounded to the nearest available value.

```
 If you call dev.setParamValueOf( "exposure Time", "110us" );
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time' with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

The following trigger modes are available: { "Disabled", "Input", "Output" }.

- Disabled : The camera is free running.
- Input : The camera outputs a frame on each rising edge of the trigger input.
- Output : The camera outputs a high signal each time a frame is outputted.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Trigger Delay Input

(only available in "Gated" Mode) This parameter is available only for WiDySWIR 320V-SP cameras.
When "Trigger Mode" is set to "Input", a trigger delay can be applied.
This is the delay between the rising edge of the trigger input and the beginning of exposure.
The available trigger delays are in microseconds: from 0.2 to 20.6 by steps of 0.08 in "Gated" mode only.
Default value 0.2.
If Mode is "Global Shutter" value is always 0.0.
Use the numeric form of setParamValueOf to set this

parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Input", 0.2 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Input", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Input'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Trigger Delay Output (only available in "Gated" Mode) This parameter is available only for WiDySWIR 320V-SP cameras.
When "Trigger Mode" is set to "Output", a trigger delay can be applied.
This is the delay between the rising edge of the trigger output and the beginning of exposure.
The available trigger delays are in microseconds: from -10.0 to 10.4 by steps of 0.08 in "Gated" mode only.
Default value -10.0.
If Mode is "Global Shutter" value is always 0.0.
Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Output", 9.6 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Output", "9.6"
);
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Output'.

Throw:

- NITException if the string value or numeric value doesn't exist.

13. WiDySenS 640V-S series parameters

NITLibrary 3.x series support WiDySenS 640V-S cameras series for USB3 with USB firmware version above or equal to 2.0.0.

The following parameters are supported:

Parameters

Number of Lines From 4 to 512 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 640 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 508 by steps of 4.
(alias "OffsetX" can be used instead)

First Column From 0 to 636 by steps of 4.
(alias "OffsetY" can be used instead)

These parameters permit to set and retrieve the ROI of the camera.
Use the unsigned int form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 600 );
```

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 512.
- NITException if the sum 'Number of Columns' + 'First Column' is above 640.

Mode Capture mode of the camera.
This parameter is writeable only for WiDySenS 640V-STP cameras.
WiDySenS 640V-ST cameras have only Global Shutter mode.
Use the string form of setParamValueOf to set this parameter.
For WiDySenS 640V-STP cameras, the following capture

modes are available: { "Global Shutter", "Gated" }.

- Global Shutter : The entire frame is exposed and captured at the same instant see https://en.wikipedia.org/wiki/Rolling_shutter.
- Gated : Global shutter mode with very short Exposure Times.

```
Example: dev.setParamValueOf( "Mode", "Global Shutter" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Mode", 0 );
```

Emit:

- NITConfigObserver.paramChanged 'Mode'.
- NITConfigObserver.paramRangeChanged 'Exposure Time' if needed.
- NITConfigObserver.paramChanged 'Temperature Tec'.
- NITConfigObserver.paramChanged 'Integration Mode' if needed.
- NITConfigObserver.paramChanged 'Analog Gain' if needed.
- NITConfigObserver.paramChanged 'Cds' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Pixel Clock

This parameter is readable only.
WiDySenS 640V-S series cameras have a unique 120MHz pixel clock.

Throw:

- NITException if you try to set this parameter.

Integration Mode

The following integration modes are available: { "Itr", "Iwr" }.

- Itr : Integrate Then Read. The image is being read

from the sensor once the exposure is finished and the next exposure can only start once the readout of the current image has been finished.
- Iwr : Integrate While Read. The image is being read from the sensor once the exposure is finished but the next exposure can start before the readout of the current image has been finished. This mode allows to reach higher frame rates.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Integration Mode", "Iwr" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Integration Mode", 1 );
```

'Iwr' integration mode cannot be used if 'Exposure Time' is below 100 microseconds or 'Trigger Mode' is set to 'Input'.

Emit:

- NITConfigObserver.paramChanged 'Integration Mode'.

Throw:

- NITException if the string value or the index value doesn't exist.
- NITException if 'Iwr' is selected and 'Exposure Time' is below 100us.
- NITException if 'Iwr' is selected and 'Trigger Mode' is set to 'Input'.

Sensor Response

The following sensor responses are available: { "Log", "Lin" }.

- Log : Logarithmic response. The output of the sensor is not proportional to the amount of light reaching the pixels. This allow wide dynamic range images(120db).
- Lin : Linear response. The output of the sensor is proportional to the amount of light reaching the pixels and the integration time.

Use the string form of setParamValueOf to set this

parameters.

```
Example: dev.setParamValueOf( "Sensor Response", "Lin" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Sensor Response", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Sensor Response'.
- NITConfigObserver.paramRangeChanged 'Analog Gain'.
- NITConfigObserver.paramChanged 'Analog Gain'. Pass to 'Low' if 'Lin' is selected. Pass to 'Standard' if Log is selected.
- NITConfigObserver.paramChanged 'Cds'. Passed to 'Off'.
- NITConfigObserver.paramRangeChanged 'Exposure Time'.
- NITConfigObserver.fpsRangeChanged.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Analog Gain

Set the gain of the sensor’s pixels.
Lower gain values have higher noise but allow higher dynamic range. High gains will give the highest sensitivity. Available Analog Gains are dependant of the 'Sensor Response':

Sensor Response	Available Analog Gain
Lin	Low
	High(see 'Cds' below for best performance with this setting)
Log	Standard

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Analog Gain", "Low" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Analog Gain", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Analog Gain'.
- NITConfigObserver.paramChanged 'Cds'. Passed to 'Off' if selected 'Analog Gain' is not 'High'.

Throw:

- NITException if the string value or the index value doesn't exist.
- NITException if the 'Analog Gain' selected is not available for the current 'Sensor Response'.

Cds

'Cds': Correlated Double Sampling is a in-sensor noise reduction method in which the signal is sampled twice: once with the sensor in a reset state and once at the end of the sensor exposure. This parameter is only valid with 'Sensor Response' set to 'Lin' and 'Analog Gain' set to 'High'. If you select a different 'Sensor Response' or a different 'Analog Gain', 'Cds' is automatically deactivated.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Cds", "Off" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Cds", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Cds'.

Throw:

- NITException if the string value or the index value doesn't exist.
- NITException if the 'Cds' selected is not available for the current 'Sensor Response' or 'Analog Gain'.

Exposure Time

The camera can only use discrete values as Exposure Time. If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
The available exposure times for this camera are dependant of capture mode and sensor response:
WiDySenS series can also have options ('Fine Exposure' and 'Long Exposure') who can affect the exposure ranges.

Gated Mode: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + ((N-Nmin)*Step)

ExpoMin	ExpoMax	Step	Nmin	Nmax
0.1	0.9	0.1	0	8
1.0	9.0	1.0	9	17

Global Shutter Mode without 'Fine Exposure' option: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + ((N-Nmin)*Step)

Sensor Response	Expo Min	Expo Max	Step	Nmin	Nmax
Lin only	10	90	10	18	26
Lin or Log	100	900	100	27	35
Lin or Log	1000	11000	1000	36	46
Lin only	12000	220000	1000	47	255

If Trigger Mode is set to External and Exposure Time is set to 220000(that is N = 255), the camera use the trigger width as Exposure Time.

Global Shutter Mode with 'Fine Exposure' option: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + ((N1-N1min)*Step) + (N2*FineStep)

Sensor Response	Expo Min	Expo Max	Step	N1min	N1max	FineStep	N2min	N2max
Lin only	10	102.75	10	18	26	0.05	0	255
Lin or Log	100	1002.00	100	27	35	0.40	0	255
Lin or Log	1000	12020.00	1000	36	46	4.00	0	255
Lin only	12000	208020.00	1000	47	254	4.00	0	255

If Trigger Mode is set to External and Exposure Time is set to 220000(that is N1 = 255), the camera use the trigger width as Exposure Time regardless of N2.

Global Shutter Mode with 'Long Exposure' option: (values are in microsecond)
'Long Exposure Option' imply 'Fine Exposure' otpion.
Exposure times are calculated by this formula: Expo = ExpoMin + ((N1-N1min)*Step) + (N2*FineStep)

Sensor Response	Expo Min	Expo Max	Step	N1min	N1max	FineStep	N2min	N2max
Lin only	10	102.75	10	18	26	0.05	0	255
Lin or Log	100	1002.00	100	27	35	0.40	0	255
Lin or Log	1000	12020.00	1000	36	46	4.00	0	255
Lin only	12000	208020.00	1000	47	254	4.00	0	255

Lin only	220000	5320000.00	0	255	255	20000.00	0	255

If Trigger Mode is set to External and Exposure Time is set to 220000(that is N1 = 255), the camera use the trigger width as Exposure Time regardless of N2.

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "ExposureTime", 500.0 );
```

You can also set this values as string:
Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
Example: dev.setParamValueOf( "exposure Time", "2500us" );
             dev.setParamValueOf( "exposure time", "2.5ms" );
```

The value set is rounded to the nearest available value.

```
 If you call dev.setParamValueOf( "exposure Time", "110us" );
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time' with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

The following trigger modes are available: { "Disabled", "Input", "Output" }.

- Disabled : The camera is free running.
- Input : The camera outputs a frame on each rising edge of the trigger input.
- Output : The camera outputs a high signal each time a frame is outputted.
- Soft Trigger(optional): The camera stops free running and outputs the number of frames passed as parameter to the function 'CaptureNFrames'. This number can be in the range [1-8].

Use the string form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Trigger Delay Input

When "Trigger Mode" is set to "Input", a trigger delay can be applied.
This is the delay between the rising edge of the trigger input and the beginning of exposure.
The available trigger delays are in microseconds and dependant from the 'Mode' parameter:
Gated Mode:
from 0.1 to 12.85 by steps of 0.05.
Default value 0.1.
Global Shutter Mode:
from -1280 to 1270 by steps of 10.0.
Default value 0.0.

In 'Global Shutter' Mode, 'Trigger Delay Input' can be negative. To make this possible, the camera use the first trigger to synchronize. This imply the first frame is never transmitted out of the camera. Furthermore the camera suppose the trigger period is constant. dont use 'Trigger Delay Input' below +130 microseconds if the trigger period is not constant or you use the 'CaptureNFrames' function(you will never have the number of frames expected).

Use the numeric form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "Trigger Delay Input", 0.2 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Input", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Input'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Trigger Delay Output

When "Trigger Mode" is set to "Output", a trigger delay can be applied.
This is the delay between the rising edge of the trigger output and the beginning of exposure.
The available trigger delays are in microseconds and dependant from the 'Mode' parameter:
Gated Mode:
from -6.4 to 6.35 by steps of 0.05.
Default value 0.0.
Global Shutter Mode:
from -1280 to 1270 by steps of 10.0.
Default value 0.0.

Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Output", 9.6 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Output", "9.6"
);
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Output'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Tec Mode

This parameter is available only for WiDySenS 640V-ST and WiDySenS 640V-STP cameras.
Permit to activate the regulation of the camera temperature.
The parameter "Temperature TEC" sets the target temperature.
The available Modes are: "No Current", "Low Current", "Medium Current" and "Strong Current".
Higher currents allow to reach the target temperature faster.

"No Current" means no regulation.

Tec Mode	Regulation Current	Comment
Low Current	40mA	camera can be powered by an USB2 connector
Medium Current	150mA	camera can be powered by an USB3 connector
Strong Current	400mA	camera **must** be powered by an external power supply

Using medium current mode with camera powered by an USB2 connector can damage your host PC.
Using strong current mode with camera powered by an USB3 connector can damage your host PC.
NIT is not liable of any damage incurred by misuse of NIT cameras.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Tec Mode", "Low Current" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Tec Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Mode'.
- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Temperature Tec

This parameter is available only for WiDySenS 640V-ST and WiDySenS 640V-STP cameras.
The available target temperatures range from -15°C to 48°C by steps of 1°C.
Default values are 15°C in 'Global Shutter' Mode and 35°C in 'Gated Mode'.

Use the numeric (double) form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "Temperature Tec", -5.0 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Temperature Tec", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

14. WiDySenS 320V-S series parameters

The following parameters are supported:

Parameters

Number of Lines From 4 to 256 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 320 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 248 by steps of 4.
(alias "OffsetX" can be used instead)

First Column From 0 to 312 by steps of 4.
(alias "OffsetY" can be used instead)

This parameter permits to set and retrieve the ROI of the camera.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 200 );
```

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 256.
- NITException if the sum 'Number of Columns' + 'First Column' is above 320.

Mode Capture mode of the camera.
This parameter is writeable only for WiDySenS 320V-STP cameras.
WiDySenS 320V-ST cameras have only Global Shutter mode.
Use the string form of setParamValueOf to set this parameter.
For WiDySenS 320V-STP cameras, the following capture modes are available: { "Global Shutter", "Gated" }.

- Global Shutter : The entire frame is exposed and captured at the same instant

see https://en.wikipedia.org/wiki/Rolling_shutter.
- Gated : Global shutter mode with very short Exposure Times.

```
Example: dev.setParamValueOf( "Mode", "Global Shutter" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Mode", 0 );
```

Emit:

- NITConfigObserver.paramChanged 'Mode'.
- NITConfigObserver.paramRangeChanged 'Exposure Time' if needed.
- NITConfigObserver.paramChanged 'Temperature Tec'.
- NITConfigObserver.paramChanged 'Integration Mode' if needed.
- NITConfigObserver.paramChanged 'Analog Gain' if needed.
- NITConfigObserver.paramChanged 'Cds' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Pixel Clock

This parameter is readable only.
WiDySenS 320V-S series cameras have a unique 120MHz pixel clock

Throw:

- NITException if you try to set this parameter.

Integration Mode

The following integration modes are available: { "Itr", "Iwr" }.

- Itr : Integrate Then Read. The image is being read from the sensor once the exposure is finished and the next exposure can only start once the readout of the current image has been finished.
- Iwr : Integrate While Read. The image is being read from the sensor once the exposure is finished but

the next exposure can start before the readout of the current image has been finished. This mode allows to reach higher frame rates.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Integration Mode", "Iwr" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Integration Mode", 1 );
```

'Iwr' integration mode cannot be used if 'Exposure Time' is below 100 microseconds or 'Trigger Mode' is set to 'Input'.

Emit:

- NITConfigObserver.paramChanged 'Integration Mode'.

Throw:

- NITException if the string value or the index value doesn't exist.
- NITException if 'Iwr' is selected and 'Exposure Time' is below 100us.
- NITException if 'Iwr' is selected and 'Trigger Mode' is set to 'Input'.

Sensor Response

The following sensor responses are available: { "Log", "Lin" }

- Log : Logarithmic response. The output of the sensor is not proportional to the amount of light reaching the pixels. This allow wide dynamic range images(120db).
- Lin : Linear response. The output of the sensor is proportional to the amount of light reaching the pixels and the integration time.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Sensor Response", "Lin" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Sensor Response", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Sensor Response'.
- NITConfigObserver.paramRangeChanged 'Analog Gain'.
- NITConfigObserver.paramChanged 'Analog Gain'. Pass to 'Low' if 'Lin' is selected. Pass to 'Standard' if Log is selected.
- NITConfigObserver.paramChanged 'Cds'. Passed to 'Off'.
- NITConfigObserver.paramRangeChanged 'Exposure Time'.
- NITConfigObserver.fpsRangeChanged.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Analog Gain

Set the gain of the sensor's pixels.
Lower gain values have higher noise but allow higher dynamic range. High gains will give the highest sensitivity.
Available Analog Gains are dependant of the 'Sensor Response':

Sensor Response	Available Analog Gain
Lin	Low
	High(see 'Cds' below for best performance with this setting)
Log	Standard

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Analog Gain", "Low" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Analog Gain", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Analog Gain'.
- NITConfigObserver.paramChanged 'Cds'. Passed to 'Off' if selected 'Analog Gain' is not 'High'.

Throw:

- NITException if the string value or the index value doesn't exist.
- NITException if the 'Analog Gain' selected is not available for the current 'Sensor Response'.

Cds

'Cds': Correlated Double Sampling is a in-sensor noise reduction method in which the signal is sampled twice: once with the sensor in a reset state and once at the end of the sensor exposure. This parameter is only valid with 'Sensor Response' set to 'Lin' and 'Analog Gain' set to 'High'. If you select a different 'Sensor Response' or a different 'Analog Gain', 'Cds' is automatically deactivated.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Cds", "Off" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Cds", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Cds'.

Throw:

- NITException if the string value or the index value doesn't exist.
- NITException if the 'Cds' selected is not available for the current 'Sensor Response' or 'Analog Gain'.

Exposure Time

The camera can only use discrete values as Exposure Time. If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
The available exposure times for this camera are dependant of capture mode and sensor response:
WiDySenS series can also have options ('Fine Exposure' and 'Long Exposure') who can affect the exposure ranges.

Gated Mode: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + ((N-Nmin)*Step)

ExpoMin	ExpoMax	Step	Nmin	Nmax
0.1	0.9	0.1	0	8
1.0	9.0	1.0	9	17

Global Shutter Mode without 'Fine Exposure' option: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + ((N-Nmin)*Step)

Sensor Response	Expo Min	Expo Max	Step	Nmin	Nmax
Lin only	10	90	10	18	26
Lin or Log	100	900	100	27	35
Lin or Log	1000	11000	1000	36	46
Lin only	12000	220000	1000	47	255

If Trigger Mode is set to External and Exposure Time is set to 220000(that is N = 255), the camera use the trigger width as Exposure Time.

Global Shutter Mode with 'Fine Exposure' option: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + ((N1-N1min)*Step) + (N2*FineStep)

Sensor Response	Expo Min	Expo Max	Step	N1min	N1max	FineStep	N2min	N2max
Lin only	10	102.75	10	18	26	0.05	0	255
Lin or Log	100	1002.00	100	27	35	0.40	0	255
Lin or Log	1000	12020.00	1000	36	46	4.00	0	255
Lin only	12000	208020.00	1000	47	254	4.00	0	255

If Trigger Mode is set to External and Exposure Time is set to 220000(that is N1 = 255), the camera use the trigger width as Exposure Time regardless of N2.

Global Shutter Mode with 'Long Exposure' option: (values are in microsecond)
'Long Exposure Option' imply 'Fine Exposure' otpion.
Exposure times are calculated by this formula: Expo = ExpoMin + ((N1-N1min)*Step) + (N2*FineStep)

Sensor Response	Expo Min	Expo Max	Step	N1min	N1max	FineStep	N2min	N2max
Lin only	10	102.75	10	18	26	0.05	0	255
Lin or Log	100	1002.00	100	27	35	0.40	0	255
Lin or Log	1000	12020.00	1000	36	46	4.00	0	255
Lin only	12000	208020.00	1000	47	254	4.00	0	255
Lin only	220000	5320000.00	0	255	255	20000.00	0	255

If Trigger Mode is set to External and Exposure Time is set to 220000(that is N1 = 255), the camera use the trigger width as Exposure Time regardless of N2.

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "ExposureTime", 500.0 );
```

You can also set this values as string:
Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
Example: dev.setParamValueOf( "exposure Time", "2500us" );
            dev.setParamValueOf( "exposure time", "2.5ms" );
```

The value set is rounded to the nearest available value.

```
If you call dev.setParamValueOf ("exposure Time", "110us");
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time' with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

The following trigger modes are available: { "Disabled", "Input", "Output" }.

- Disabled : The camera is free running.
- Input : The camera outputs a frame on each rising edge of the trigger input.
- Output : The camera outputs a high signal each time a frame is outputted.
- Soft Trigger(optional): The camera stops free running and outputs the number of frames passed as parameter to the function 'CaptureNFrames'. This number can be in the range [1-8].

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.

- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Trigger Delay Input

When "Trigger Mode" is set to "Input", a trigger delay can be applied.
This is the delay between the rising edge of the trigger input and the beginning of exposure.
The available trigger delays are in microseconds and dependant from the 'Mode' parameter:
Gated Mode:
from 0.1 to 12.85 by steps of 0.05.
Default value 0.1.
Global Shutter Mode:
from -1280 to 1270 by steps of 10.0.
Default value 0.0.

In 'Global Shutter' Mode, 'Trigger Delay Input' can be negative. To make this possible, the camera use the first trigger to synchronize. This imply the first frame is never transmitted out of the camera. Furthermore the camera suppose the trigger period is constant. Dont use 'Trigger Delay Input' below +130 microseconds if the trigger period is not constant or you use the 'CaptureNFrames' function(you will never have the number of frames expected).

Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Input", 0.2 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Input", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Input'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Trigger Delay Output

When "Trigger Mode" is set to "Output", a trigger delay can

be applied.
This is the delay between the rising edge of the trigger output and the beginning of exposure.
The available trigger delays are in microseconds and dependant from the 'Mode' parameter:
Gated Mode:
from -6.4 to 6.35 by steps of 0.05.
Default value 0.0.
Global Shutter Mode:
from -1280 to 1270 by steps of 10.0.
Default value 0.0.

Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Output", 9.6 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Output", "9.6"
);
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Output'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Tec Mode

This parameter is available only for WiDySenS 320V-ST and WiDySenS 320V-STP cameras.
Permit to activate the regulation of the camera temperature.
The parameter "Temperature TEC" sets the target temperature.
The available Modes are: "No Current", "Low Current", "Medium Current" and "Strong Current".
Higher currents allow to reach the target temperature faster.
"No Current" means no regulation.

Tec Mode	Regulation Current	Comment
Low Current	40mA	camera can be powered by an USB2 connector
Medium Current	150mA	camera can be powered by an USB3 connector
Strong Current	400mA	camera **must** be powered by an external power supply

Using medium current mode with camera powered by an USB2 connector can damage your host PC.
Using strong current mode with camera powered by an USB3 connector can damage your host PC.
NIT is not liable of any damage incurred by misuse of NIT cameras.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Tec Mode", "Low Current" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Tec Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Mode'.
- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Temperature Tec

This parameter is available only for WiDySenS 320V-ST and WiDySenS 320V-STP cameras.
The available target temperatures range from -15°C to 48°C by steps of 1°C.
Default values are 25°C in 'Global Shutter' Mode and 35°C in 'Gated Mode'.

Use the numeric(double) form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "Temperature Tec", -5.0 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Temperature Tec", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value

doesn't exist.

15. (Hipe) SenS 640V-S series parameters

NITLibrary 3.x series support SenS 640V-S cameras series for USB3 with USB firmware version above or equal to 2.0.0.

SenS 640V-S series cameras are SWIR cameras, they need an activated NUC and BPR to output correct results. See **NUC and BPR**.

The following parameters are supported:

Parameters

Number of Lines From 4 to 512 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 640 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 508 by steps of 4.
(alias "OffsetX" can be used instead)

First Column From 0 to 636 by steps of 4.
(alias "OffsetY" can be used instead)

These parameters permit to set and retrieve the ROI of the camera.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 600 );
```

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 512.
- NITException if the sum 'Number of Columns' + 'First Column' is above 640.

Mode Capture mode of the camera.
This parameter is readable only.
SenS 640V-S series cameras have only Global Shutter mode.

Throw:

- NITException if you try to set this parameter.

Pixel Clock

This parameter is readable only.
SenS 640V-S series cameras have a unique 120MHz pixel clock.

Throw:

- NITException if you try to set this parameter.

Integration Mode

The following integration modes are available: { "Itr", "Iwr" }.

- Itr : Integrate Then Read. The image is being read from the sensor once the exposure is finished and the next exposure can only start once the readout of the current image has been finished.
- Iwr : Integrate While Read. The image is being read from the sensor once the exposure is finished but the next exposure can start before the readout of the current image has been finished. This mode allows reaching higher frame rates.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Integration Mode", "Iwr" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Integration Mode", 1 );
```

'Iwr' integration mode cannot be used if 'Exposure Time' is below 100 microseconds or 'Trigger Mode' is set to 'Input'.

Emit:

- NITConfigObserver.paramChanged 'Integration Mode'.

Throw:

- NITException if the string value or the index value doesn't exist.
- NITException if 'Iwr' is selected and 'Exposure Time' is below 100us.
- NITException if 'Iwr' is selected and 'Trigger Mode'

is set to 'Input'.

Analog Gain

Set the gain of the sensor's pixels.
Lower gain values have higher noise but allow higher dynamic range. High gains will give the highest sensitivity.
Available Analog Gains are: { 'Low', 'Medium', 'High' }; Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Analog Gain", "Low" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Analog Gain", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Analog Gain'.

Throw:

- NITException if the string value or the index value doesn't exist.

Exposure Time

The camera can only use discrete values as Exposure Time. If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
The available exposure times for this camera are dependant of capture mode:

Gated Mode: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + (N*Step)

ExpoMin	ExpoMax	Step	Nmin	Nmax
0.1	51.25	0.05	0	1023

Global Shutter Mode: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + (N*Step)

Expo Min	Expo Max	Step	Nmin	Nmax
10	112.3	0.1	0	1023
100	1123.0	1.0	0	1023
1000	11230.0	10.0	0	1023
10000	112300.0	100.0	0	1023
100000	1123000.0	1000.0	0	1023
1000000	11230000.0	10000.0	0	1023
10000000	112300000.0	100000.0	0	1023

If Trigger Mode is set to External and Exposure Time is set

to 0.0, the camera uses the trigger width as Exposure Time.

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "ExposureTime", (double)500.0 );
```

You can also set this values as string:
Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
Example: dev.setParamValueOf( "exposure Time", "2500us" );
         dev.setParamValueOf( "exposure time", "2.5ms" );
```

The value set is rounded to the nearest available value.

```
If you call dev.setParamValueOf( "exposure Time", "110us" );
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time' with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

The following trigger modes are available: { "Disabled", "Input", "Output" }.

- Disabled : The camera is free running.
- Input : The camera outputs a frame on each rising edge of the trigger input.
- Output : The camera outputs a high signal each time a frame is outputted.
- Soft Trigger(optional): The camera stops free running and outputs the number of frames passed as parameter to the function 'CaptureNFrames'. This number can be in the range [1-256].

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this

case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Trigger Delay Input

When "Trigger Mode" is set to "Input", a trigger delay can be applied.
This is the delay between the rising edge of the trigger input and the beginning of exposure.
The available trigger delays are in microseconds and range from -1280 to 1270 by steps of 10.0.
Default value 0.0.

In 'Global Shutter' Mode, 'Trigger Delay Input' can be negative.
To make this possible, the camera uses the first trigger to synchronize. This implies the first frame is never transmitted out of the camera. Furthermore the camera supposes the trigger period is constant.
Don't use 'Trigger Delay Input' below +130 microseconds if the trigger period is not constant or you use the 'CaptureNFrames' function(you will never have the number of frames expected).

Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Input", 0.2 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Input", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Input'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Trigger Delay Output

When "Trigger Mode" is set to "Output", a trigger delay can be applied.
This is the delay between the rising edge of the trigger output and the beginning of exposure.
The available trigger delays are in microseconds and range from -1280 to 1270 by steps of 10.0.
Default value 0.0.

Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Output", 9.6 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Output", "9.6" );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Output'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Tec Mode

Permit to activate the regulation of the camera temperature.
The parameter "Temperature TEC" permit to set the target temperature.
The available Modes are: "No Current" and "Strong Current".
"No Current" mean no regulation.
"Strong Current" need the camera is powered by an external power supply through the Hirose connector.

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Tec Mode", "Strong Current" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Tec Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Mode'.

- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Temperature Tec

SenS 640V-ST cameras:
The available target temperatures range from -15°C to 48°C by steps of 1°C.
Default values is 15°C.

Hipe SenS 640V-ST cameras:
The available target temperatures range from -30°C to 33°C by steps of 1°C.
Default values is -15°C.

Use the numeric(double) form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "Temperature Tec", -5.0 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Temperature Tec", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

16. SenS 1280V-S series parameters

NITLibrary 3.x series support SenS 1280V-S cameras series for USB3 with USB firmware version above or equal to 2.1.0.

The following parameters are supported:

Parameters

Number of Lines From 4 to 1024 by steps of 8.
(alias "Height" can be used instead)

Number of Columns From 64 to 1280 by steps of 8
(alias "Width" can be used instead)

First Line From 0 to 1020 by steps of 4.
(alias "OffsetX" can be used instead)

First Column From 0 to 1276 by steps of 4.
(alias "OffsetY" can be used instead)

These parameters permit to set and retrieve the ROI of the camera.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "NumberOfLines", 600 );
```

One can also use NITDevice.setRoi(offset_x, offset_y, width, height); to set all the dimensions.

Emit:

- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Number of Lines' + 'First Line' is above 1024.
- NITException if the sum 'Number of Columns' + 'First Column' is above 1280.

Stacked Lines Step From 1 to 16 by steps of 1.

These parameters allow activating only a subset of the sensor height on a line basis.
Eg: If Stacked Lines Step = 2, only one line out of two will be outputted by the camera.
If Stacked Lines Step is 1 the image height is defined by

parameter 'Number of Lines'.
If 'Number of Lines' is higher than (sensorHeight() - 'First Line')/'Stacked Line Step', 'Number of Lines' is set to the nearest upper multiple of 8 of (sensorHeight() - 'First Line')/'Stacked Line Step'.

Example: If 'Stacked Line Step' = 2, 'Number of Lines' = 1024 and 'First Line' = 0:

- (sensorHeight() - 'First Line')/'Stacked Line Step' = 512.
- 'Number of Lines' is higher than 512, so it is reset to 512.
- The number of lines outputted by the camera is 512.First line = 0, last line = 1022.

Example: If 'Stacked Line Step' = 2, 'Number of Lines' = 400 and 'First Line' = 0:

- (sensorHeight() - 'First Line')/'Stacked Line Step' = 512.
- 'Number of Lines' is lower than 512, so it is left at 400.
- The number of lines outputted by the camera is 400. First line = 0, last line = 398.

Use the unsigned int form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "Stacked Lines Step", 4 );
```

You can also set this values as text value:

```
Example: dev.setParamValueOf( "Stacked Lines Step", "4" );
```

Emit:

- NITConfigObserver.paramChanged 'Stacked Lines Step'.
- NITConfigObserver.paramChanged 'Number of Lines' if needed.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the value is out of range.

Stacked Block_1

Stacked Block_2

Stacked Block_3

Stacked Block_4

Stacked Block_5

Stacked Block_6

Stacked Block_7

Stacked Block_8

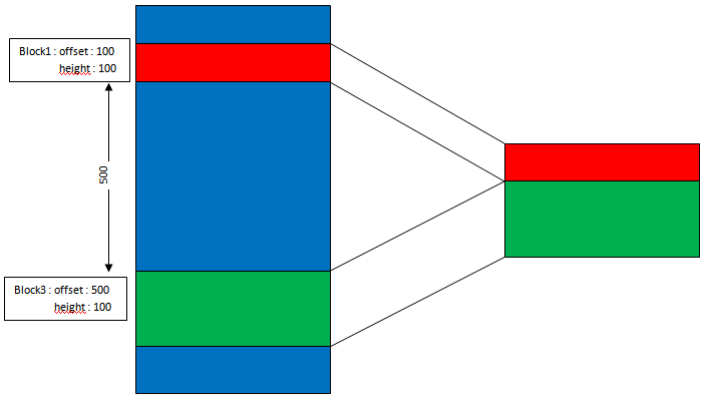
These parameters allow activating only a subset of the sensor height on a block basis.

A stacked block can be enabled or disabled.
The height of the resulting frame is the sum of the heights of the enabled blocks at any given time.

A stacked block is defined by two parameters: offset and height.
offset: The offset of the block relative to the last row of the preceding enabled block by steps of 4(the first enabled block offset is the offset relative to first row of the sensor).
height: The height of the block by steps of 4.
Setting multiple blocks permits to receive only a partial image.

Example: With Stacked Block_1 offset = 100, height = 100 and Stacked Block_3 offset = 500, height = 200:

- The outputted image contains 300 rows beginning at offset 100, with a gap of 500.
- The first row of the resulting frame is row 100 of the sensor.
- The lines are contiguous until row 199 of the sensor is reached.
- After row 199 of the sensor, the frame contains the row 700 of the sensor.
- The last line of the frame corresponds to the row 899 of the sensor.



Stacked Blocks

To be active, the stacked blocks must be enabled by the 'Stacked Blocks Enable' parameter.
The different enabled stacked blocks must not overlap.

Stacked Blocks are not taken into account by the camera until you enable the blocks by 'Stacked Blocks Enable' parameter.
If you modify a Stacked Block value which is enabled, you must set the parameter and enable it a new time for the value to be taken into account.

Unsigned int form of setParamValueOf usage:
As a stacked block has two parameters we pass the offset in the high part of the unsigned int and height in the low part.

```
Example:

unsigned int offset = 150;
unsigned int heigth = 300;

dev.setParamValueOf( "Stacked Block_1", offset<<16|height );
```

The helper class NITStackedBlock allows easing the use of these functions.

```
Example:

unsigned int offset = 150;
unsigned int heigth = 300;

block = NITStackedBlock( offset, height );
dev.setParamValueOf( "Stacked Block_2", block );
```

String form of setParamValueOf usage: As a stacked block has two parameters we pass a string containing offset and height separated by a semi-column(;).

```
Example: dev.setParamValueOf( "Stacked Block_1", "150;300"
);
```

The helper class NITStackedBlock allows easing the use

of these functions.

```
Example:

unsigned int offset = 150;
unsigned int heigth = 300;

NITStackedBlock block( offset, height );

dev.setParamValueOf( "Stacked Block_2", block.to_string() );
```

Emit:

- NITConfigObserver.paramChanged 'Stacked Block_n' with (n = 1 to 8).

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if offset and/or height are out of range.
- NITException if the sum of offset and height is above sensorHeight().
- NITException if offset and/or height are not multiple of 4.
- NITException if the string form of setParamValueOf is not correctly formatted.

Stacked Blocks Enable bitset of 8 bits

Each bit in the bitset corresponds to a stacked block.
Bit 0 correspond to 'Stacked Block_1', Bit 1 correspond to 'Stacked Block_2' and so on.
Bit 0 is the less significand bit.
If bit is 0 block is disabled, if bit is 1, block is enabled.
If all bits are 0, the camera use the ROI parameters and/or the 'Stacked Lines Step' parameter if not one.
The sum of the enabled stacked blocks heights must be a multiple of 8.
Use the unsigned int form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "StackedBlocksEnable", 0x05 );
//Enable Stacked Block_1 and Stacked Block_3
```

You can also set these values as text value. In this case each bit is represented by one character('0' or '1'):

```
Example: dev.setParamValueOf( "Stacked Blocks Enable",
"0101" );
```

Emit:

- NITConfigObserver.paramChanged 'Stacked Blocks Enable'.

- NITConfigObserver.paramChanged 'Number of Rows'.
- NITConfigObserver.fpsRangeChanged if needed.
- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if value is above 255.
- NITException if the enabled blocks overlaps.
- NITException if the sum of the heights is above sensor height.
- NITException if the sum of the heights of the enabled blocks is above sensor height.
- NITException if the sum of of offsets and heights of the enabled stacked blocks is above sensor height.

Mode

Capture mode of the camera

Use the string form of setParamValueOf to set this parameter.
The following capture modes are available: { "Global Shutter", "Gated" }.

- Global Shutter : The entire frame is exposed and captured at the same instant
  see https://en.wikipedia.org/wiki/Rolling_shutter.
- Gated : Global shutter mode with very short Exposure Times.

```
Example: dev.setParamValueOf( "Mode", "Global Shutter" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Mode", 0 );
```

Emit:

- NITConfigObserver.paramChanged 'Mode'.
- NITConfigObserver.paramRangeChanged 'Exposure Time' if needed.
- NITConfigObserver.paramChanged 'Integration Mode' if needed.
- NITConfigObserver.paramChanged 'Analog Gain' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.
- NITConfigObserver.fpsRangeChanged if needed.

- NITConfigObserver.fpsChanged if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Pixel Clock

This parameter is readable only.
SenS 1280V-S series cameras have a unique 120MHz pixel clock.

Throw:

- NITException if you try to set this parameter.

Analog Gain

Set the gain of the sensor's pixels.

Lower gain values have higher noise but allow higher dynamic range.\ High gains will give the highest sensitivity.

Available Analog Gains are: { "Low", "High" }

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Analog Gain", "Low" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Analog Gain", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Analog Gain'.

Throw:

- NITException if the string value or the index value doesn't exist.

Exposure Time

The camera can only use discrete values as Exposure Time. If you set a value not allowed by the camera, the nearest allowed value is used and returned in ParamChanged.
The available exposure times for this camera are dependant of capture mode:

Gated Mode: (values are in microsecond)
Exposure times are calculated by this formula: Expo = ExpoMin + ((N-Nmin)*Step)

ExpoMin	ExpoMax	Step	Nmin	Nmax
0.1	0.9	0.1	0	8
1.0	9.0	1.0	9	17

Global Shutter Mode: (values are in microsecond) Exposure times are calculated by this formula: Expo = ExpoMin + ((N1-N1min)*Step) + (N2*FineStep)

Expo Min	Expo Max	Step	N1min	N1max	FineStep	N2min	N2max
10	102.75	10	18	26	0.05	0	255
100	1002.00	100	27	35	0.40	0	255
1000	221020.00	1000	36	254	4.00	0	255

If Trigger Mode is set to External and Exposure Time is set to 220000(that is N1 = 255), the camera use the trigger width as Exposure Time regardless of N2.

Use the double form of setParamValueOf to set this parameters in numeric format.

```
Example: dev.setParamValueOf( "ExposureTime", 500.0 );
```

You can also set this values as string:
Well formed string must follow this pattern <double value>< zero or more spaces>< ns or us or µs or ms or s >.
A string value with no unit is considered in microseconds.
Examples: 2ms, 1 s, 30000us, 0.3s

```
Example: dev.setParamValueOf( "exposure Time", "2500us" );
         dev.setParamValueOf( "exposure time", "2.5ms" );
```

The value set is rounded to the nearest available value.

```
If you call dev.setParamValueOf( "exposure Time", "110us" );
the value will be rounded to 100us.
```

Emit:

- NITConfigObserver.paramChanged 'Exposure Time' with the value really set.
- NITConfigObserver.fpsRangeChanged.

Throw:

- NITException if the string value is ill formed, that is the unit is unknown or the value is not a number.
- NITException if the capture mode is not 'Global Shutter'.

Trigger Mode

The following trigger modes are available: { "Disabled", "Input", "Output", "Soft Trigger" }.

- Disabled : The camera is free running.
- Input : The camera outputs the number of frames defined by 'Trigger Burst' parameter on each rising edge of the trigger input.
  The first frame is outputted after a delay defined by 'Trigger Input Delay' parameter. The subsequent ones are outputted at the current frame rate.
  If a new trigger arrives while the burst is not complete, the trigger is ignored.
- Output : The camera outputs a high signal each time a frame is outputted.
- Soft Trigger: The camera stops free running and outputs the number of frames passed as parameter to the function 'CaptureNFrames'. This number can be in the range [1-255].

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "Trigger Mode", "Input" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Trigger Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Mode'.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input' if needed.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Output' if needed.

Throw:

- NITException if the string value or the index value doesn't exist.

Trigger Burst

From 1 to 256

The camera outputs the number of frames defined by 'Trigger Burst' parameter on each rising edge of the trigger input.
The first frame is outputted on the rising edge of the trigger input. The subsequent ones are outputted at the

current frame rate.
If a new trigger arrives while the burst is not complete, the trigger is ignored.
"Trigger Delay Input" is forced to 0.

Use the unsigned int form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "Trigger Burst", 15 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Burst", "15" );
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Burst'.
- NITConfigObserver.paramRangeChanged 'Trigger Delay Input'.

Throw:

- NITException if the string value or the index value doesn't exist.

Trigger Delay Input

When "Trigger Mode" is set to "Input", a trigger delay can be applied.
This is the delay between the rising edge of the trigger input and the beginning of exposure.
"Trigger Delay Input" is incompatible with "Trigger Burst" higher than 1.

The available trigger delays are in microseconds and dependant from the 'Mode' parameter:
Gated Mode:
from 0.1 to 12.85 by steps of 0.05.
Default value 0.1.
Global Shutter Mode:
from -1280 to 1270 by steps of 10.0.
Default value 0.0.

In 'Global Shutter' Mode, 'Trigger Delay Input' can be negative.
To make this possible, the camera uses the first trigger to synchronize. This implies the first frame is never transmitted out of the camera. Furthermore the camera supposes the trigger period is constant.
Dont use 'Trigger Delay Input' below +130 microseconds if the trigger period is not constant or you use the 'CaptureNFrames' function(you will never have the number of frames expected).

Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Input", 0.2 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Input", "0.2"
);
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Input'.

Throw:

- NITException if the string value or numeric value doesn't exist.
- NITException if "Trigger Burst" is not one and you try to set "Trigger Delay Input" not 0.

Trigger Delay Output

When "Trigger Mode" is set to "Output", a trigger delay can be applied.
This is the delay between the rising edge of the trigger output and the beginning of exposure.

The available trigger delays are in microseconds and dependant from the 'Mode' parameter:
Gated Mode:
from -6.4 to 6.35 by steps of 0.05.
Default value 0.0.
Global Shutter Mode:
from -1280 to 1270 by steps of 10.0.
Default value 0.0.

Use the numeric form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Trigger Delay Output", 9.6 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Trigger Delay Output", "9.6"
);
```

Emit:

- NITConfigObserver.paramChanged 'Trigger Delay Output'.

Throw:

- NITException if the string value or numeric value

doesn't exist.

TriggerOut_1

TriggerOut_2

Set the behaviour of the GPO signals Out1 and Out2.
The following behaviours are available:

String value	Numeric value	Comment
Trigger Input	0	Output mirrors the external trigger signal
Trigger Ready	1	Output is high when camera is ready to receive an external trigger signal
Exposing	2	Output is high when camera exposes
ReadOut	3	Output is high when camera read pixels
Imaging	4	Output is high during exposition and readout
GPIO Low	5	Output is forced low
GPIO High	6	Output is forced high

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerOut_1", "Trigger Input" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerOut_1", 4 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerOut_1' or 'TriggerOut_2'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

Led

Turn the led on or off.

At startup, the led is allways on.
The following modes are available: { "Off", "On" }.
The behaviour is as follows:

Color	Description
Red	Blink when camera receive commands
Green	Blink when frames are outputted

Blue	TEC behaviour • Off(continuous):when TEC is off • Blink: when TEC on and target temperature not reached • On(continuous):when TEC on and target temperature reached

Use the unsigned int form of setParamValueOf to set these parameter.

```
Example: dev.setParamValueOf( "Led", 1 );
```

You can also set these values as string values:

```
Example: dev.setParamValueOf( "Led", "On" );
```

Emit:

- NITConfigObserver.paramChanged 'Led'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

Tec Mode

Permit to activate the regulation of the camera temperature.

This parameter allows setting pre-defined power limits for the camera cooling system to avoid thermal runaway depending on the heat removal capacity available in your system.
The “Temperature TEC” parameter sets the target temperature.
When setting a new operating temperature / power limit. The measured sensor temperature should be monitored to make sure the target temperature is reached.
Sufficient heatsinking might need to be provided depending on the ambient temperature and sensor setpoint to ensure a proper regulation.
The available Modes are: "No Current", "Low Current", "Medium Current" and "Strong Current".
Higher currents allow reaching the target temperature faster.
"No Current" means no regulation.

Tec Mode	Regulation Current	Comment
Low Current	250mA	camera can be powered by an USB2 connector

Medium Current	380mA	camera can be powered by an USB3 connector
Strong Current	830mA	camera **must** be powered by an external power supply

Use the string form of setParamValueOf to set this parameters.

```
Example: dev.setParamValueOf( "Tec Mode", "Low Current" );
```

You can also set this values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "Tec Mode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Mode'.
- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

Temperature Tec

The available target temperatures range from -15°C to 48°C by steps of 1°C.
Default values are 15°C in 'Global Shutter' Mode and 35°C in 'Gated Mode'.

Use the numeric(double) form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "Temperature Tec", -5.0 );
```

You can also set this values as string value:

```
Example: dev.setParamValueOf( "Temperature Tec", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'Tec Temperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

17. MC1003-1GB series parameters

NITLibrary 3.x series support NSC1003GigE cameras with firmware version above or equal to 2.7.0.

The following parameters are supported:
Where not otherwise indicated parameters are Read/Write.

Parameters

WidthMax

Expert **ReadOnly**
Integer value in the range [1, 1280] pixels.

Updated each time OffsetX, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Width' parameter.

HeightMax

Expert **ReadOnly**
Integer value in the range [1, 1024] pixels.

Updated each time OffsetY, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Height' parameter.

Width

Beginner
Integer value in the range [2, WidthMax] by steps of 2 pixels.
The value you can set is constrained by the frame rate and 'WidthMax'.

Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Width", 600 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Width", "600" );
```

Emit:

- NITConfigObserver.paramChanged 'Width'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

Height

Beginner
Integer value in the range [2, HeightMax] by steps of 2 pixels.
The value you can set is constrained by the frame rate and 'HeightMax'.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Height", 500 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Height", "500" );
```

Emit:

- NITConfigObserver.paramChanged 'Height'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

OffsetX

Beginner
Integer value in the range [0, 1270] by steps of 10 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetX", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetX", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetX'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

OffsetY

Beginner
Integer value in the range [0, 1010] by steps of 10 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetY", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetY", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetY'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

PixelFormat

Expert
While this parameter is writeable, its use is limited.
The pixel format is in fact changed by the 'OutputType' parameter.
The following pixel formats are available:

String value	Numeric value	Comment	NITFrame pixel type
Mono8	17301505	Set if 'OutputType' is set to 'Correction_Grey'	FLOAT
Mono14	17825829	Set if 'OutputType' is set to 'RAW'	FLOAT

Emit:

- NITConfigObserver.paramChanged 'PixelFormat'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

OutputType

Beginner
Permits to set the desired output type.
In all cases, if NUC is active it is applied before the OutputType.
The following output types are available:

String value	Numeric value	Comment
Correction_Grey	0	Activate gain control.

		Change 'PixelFormat' to Mono8
RAW	3	Deactivate gain control. Change 'PixelFormat' to Mono14

RAW is the output of the sensor with application of the NUC if it is active.

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "OutputType",
"Correction_Grey" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "OutputType", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'OutputType'.
- NITConfigObserver.paramChanged 'PixelFormat'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionMode

Beginner
This parameter is writable but can take only one value.

String value	Numeric value	Comment
Continuous	0	Frames are continuously outputted until AcquisitionStop is called

Emit:

- NITConfigObserver.paramChanged 'AcquisitionMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionStart

Beginner
This parameter is a Command. The only possible value is 1.
Start to output frames.

This command is a direct command to the camera. It bypasses the variables initialisations performed by the SDK.
Prefer NITDevice.start(), NITDevice.captureNFrames() or NITDevice.captureForDuration() to start the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStart", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStart", "1"
);
```

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the camera is already started.

AcquisitionStop

Beginner
This parameter is a Command. The only possible value is 1.
Stop to output frames.

This command is a direct command to the camera. It bypasses the variables initialisations performed by the SDK.
Prefer NITDevice.stop() to stop the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStop", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStop", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionFrameRate

Beginner
Set the frame rate.
Float value in the range [1.0,MaxFps] Hz
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionFrameRate", 100.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionFrameRate", "100.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AcquisitionFrameRate'.
- NITConfigObserver.paramChanged 'MaxExposure'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current ExposureTime or resolution.

MaxFPS

Beginner **ReadOnly**
Float value in the range [1,10000] Hz.
Updated each time the resolution or the ExposureTime are changed.
This is the maximum value which can be actually set for the 'AcquisitionFrameRate' parameter.

ReadOutMode

Beginner
Set the acquisition mode of the device.
This parameter can take these values:

String value	Numeric value	Comment
Global	0	The entire frame is exposed and captured at the same instant
Rolling	1	The frame is exposed and captured line by line
Differential	2	The output frame is the difference of two succesive

		frames in Global Shutter

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "ReadOutMode", "Rolling"
);
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "ReadOutMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'ReadOutMode'.
- NITConfigObserver.paramChanged 'MaxExposure'.

NITConfigObserver.paramChanged 'MaxFps'. **Throw:**

- NITException if the string value or the numeric value doesn't exist.

TriggerDirection

Beginner
Set the direction of the trigger signal.
The following directions are available:

String value	Numeric value	Comment
Internal	0	The trigger signal is generated and outputted by the camera each time a frame is outputted
External	1	The trigger signal is external and start acquisition of one frame

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerDirection",
"External" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerDirection", 1 );
```

Emit:

- NITConfigObserver.paramChanged

'TriggerDirection'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

TriggerOut1

Beginner

TriggerOut2

Beginner
Set the behaviour of the GPO signals Out1 and Out2.
The following behaviours are available:

String value	Numeric value	Comment
Exposing	0	Output is high when camera exposes
TriggerReady	1	Output is high when camera is ready to receive an external trigger signal
TriggerInput	2	Output mirrors the external trigger signal
ReadOut	3	Output is high when camera read pixels
Imaging	4	Output is high during exposition and readout
GPOLowLevel	5	Output is forced low
GPOHighLevel	6	Output is forced high

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerOut1",
"TriggerInput" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerOut1", 4 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerOut1' or 'TriggerOut2'.

Throw:

- NITException if the string value or the

numeric value doesn't exist.

ExposureTime

Beginner
Set the duration of the exposition.
Integer value in the range [100,MaxExposureTime] microseconds
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "ExposureTime", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "ExposureTime", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'ExposureTime'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current 'AcquisitionFrameRate'.

MaxExposureTime

Beginner ReadOnly
Integer value in the range [0,65535] microseconds.
Updated each time the resolution or the 'AcquisitionFrameRate' are changed.
This is the maximum value which can be actually set for the 'ExposureTime' parameter.

CamLoadConfiguration

Beginner
This parameter is a Command. The only possible value is 1.
Load the parameters stored in flash.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamLoadConfiguration", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamLoadConfiguration",
"1" );
```

Throw:

- NITException if the string value or the

numeric value doesn't exist.

CamSaveConfiguration

Beginner
This parameter is a Command. The only possible value is 1.
Save the current parameters in flash.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamSaveConfiguration", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamSaveConfiguration", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

CamResetParameter

Beginner
This parameter is a Command. The only possible value is 1.
Reset the current parameters to their default value.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamResetParameter", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamResetParameter", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NIOS_IS_BUSY

Beginner **ReadOnly**
Running some commands like eg. 'CamSaveConfiguration' access the flash and can take some time.
During this time new commands or parameter setting are not taken into account.
This parameter permits to know if it is safe to send commands and parameters to the camera.
If returned value is 1: The camera is busy, don't send commands or parameters.
If returned value is 0: The camera is not busy, you can safely send commands or parameters.

This information is for documentation only.
NITLibrary look after the busyness of the camera before sending parameters or commands.

NucActivate

Beginner
Activate or Deactivate the embedded Nuc processing.
If embedded Nuc is activated, library Nuc is deactived.
The Nuc applied is the Nuc in the current selected table. see nucBpr
This parameter can take these two values:

String value	Numeric value	Comment
Off	0	Embedded NUC is not active
On	1	Embedded NUC is active

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucActivate", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucActivate", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'NucActivate'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucCompute1Pts

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point for this device with the current configuration, the gain is untouched. The resulting Nuc is set in the current selected table. This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute1Pts", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute1Pts", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSelectTable

Beginner
Select the current Nuc table who will be applied to the captured frames.
At delivery time of the camera, NUC tables are empty, as NUC is not mandatory for CMOS sensors.
This parameter can take these values:

String value	Numeric value
NONE	0
NUC1	1
NUC2	2
NUC3	3
NUC4	4

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSelectTable", "NUC2"
);
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucSelectTable", 2 );
```

Emit:

- NITConfigObserver.paramChanged 'NucSelectTable'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucResetTable

Beginner
This parameter is a Command. The only possible value is 1.
Reset the content of the current selected Nuc table entry in memory(the Nuc table in flash is untouched).
The low point is set to all 0 and the gain matrix is set

to all ones.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucResetTable", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucResetTable", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSaveInFlash

Beginner
This parameter is a Command. The only possible value is 1.
Save the content of the current selected Nuc table entry in flash memory(the Nuc table in flash is overridden).
As we are writing to flash memory, this operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSaveInFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucSaveInFlash", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucLoadFromFlash

Beginner
This parameter is a Command. The only possible value is 1.
Load the content of the current selected Nuc table entry from flash to memory(the Nuc table in memory is overridden).
As we are reading from flash memory, this operation may take some time.
Use the numeric form of setParamValueOf to set

these parameters.

```
Example: dev.setParamValueOf( "NucLoadFromFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucLoadFromFlash", "1"
);
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcMode

Beginner
Set the method used to apply gain control.
The gain control is active if 'OutputType' is not 'RAW'.
This parameter can take these two values:

String value	Numeric value	Comment
Auto	0	The gain and offset are automatically calculated
Manuel	1	The gain and offset are preset

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcMode", "Auto" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcManualGain

Beginner
Set the gain used when gain control is set to manual.
Float value in the range [0.0, 1.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualGain", 0.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualGain", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualGain'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcManualOffset

Beginner
Set the offset used when gain control is set to manual.
Float value in the range [0.0, 16383.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualOffset", 10.2
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualOffset", "10.2"
);
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualOffset'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcAutoMeanStdThreshold

Beginner
Set the threshold used when gain control is set to auto.
Float value in the range [0.0, 16383.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"AgcAutoMeanStdThreshold", 10.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"AgcAutoMeanStdThreshold", "10.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcAutoMeanStdThreshold'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcSmoothingMethod

Beginner
Set the smoothing coefficient used to calculate gain control.
Smoothing is done with a temporal averaging on gain and offset.
'Slow' averages on more frames.
This parameter is only effective in 'Auto' 'AgcMode'.
This parameter can take these values:

String value	Numeric value
Slow	0
Medium	1
Fast	2

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcSmoothingMethod",
"Auto" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcSmoothingMethod", 1
);
```

Emit:

- NITConfigObserver.paramChanged 'AgcSmoothingMethod'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcAutoGain

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated gain for the last frame when gain control is set to auto.

AgcAutoOffset

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated offset for the last frame when gain control is set to auto.

AgcStatisticsFmean

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the mean pixel value of the last frame.

AgcStatisticsFstd

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the standard deviation value of the last frame.

AgcStatisticsWmin

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the minimum pixel value of the last frame.

AgcStatisticsWmax

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the maximum pixel value of the last frame.

AgcRoiMode

Beginner
Set the rectangular region of the frame on which the gain control is calculated.
This parameter can take these values:

String value	Numeric value	Comment
FULL_FRAME	0	Region of interest is the entire frame
HALF_FRAME	1	Region of interest is centered in the frame
QUARTER_FRAME	2	Region of interest is centered in the frame
CUSTOM_FRAME	3	Region of interest is anywhere in the frame
		The rectangle is set by the parameters:
		- AgcRoiCustomLeft
		- AgcRoiCustomTop
		- AgcRoiCustomRight
		- AgcRoiCustomBottom

Use the string form of setParamValueOf to set these

parameters.

```
Example: dev.setParamValueOf( "AgcRoiMode",
"HALF_FRAME" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcRoiCustomTop	**Beginner**	Integer in the range [0, UINT_MAX]
AgcRoiCustomBottom	**Beginner**	Integer in the range [0, UINT_MAX]
AgcRoiCustomLeft	**Beginner**	Integer in the range [0, UINT_MAX]
AgcRoiCustomRight	**Beginner**	Integer in the range [0, UINT_MAX]

Set the rectangular region of the frame on which the gain control is calculated when the 'RoiMode' parameter is set to 'CUSTOM_FRAME'.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiCustomTop", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcRoiCustomBottom",
"100" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiCustomTop', 'AgcRoiCustomBottom', 'AgcRoiCustomLeft' or 'AgcRoiCustomRight'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcRoiDraw **Beginner**

Draw a rectangular frame in the image representing

the rectangular region where the gain control is calculated.
This parameter can take this two values:

String value	Numeric value	Comment
Off	0	Rectangle not drawn
On	1	Rectangle drawn

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiDraw", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiDraw", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiDraw'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

Gamma

Beginner
Float in the range [0, 3]
Set gamma factor. A value of 1 imply no gamma correction
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "Gamma", 1.5 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Gamma", "1.4" );
```

Emit:

- NITConfigObserver.paramChanged 'Gamma'.

Throw:

- NITException if the string value or the numeric value is out of range.

GammaThd

Beginner
Integer in the range [0, 255]

Set the center of the gamma.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GammaThd", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GammaThd", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'GammaThd'.

Throw:

- NITException if the string value or the numeric value is out of range.

CustomReg_1 **Expert**

CustomReg_2 **Expert**

CustomReg_3 **Expert**

CustomReg_4 **Expert**

CustomReg_5 **Expert**

CustomReg_6 **Expert**

CustomReg_7 **Expert**

CustomReg_8 **Expert**

CustomReg_9 **Expert**

CustomReg_10 **Expert**

Integer in the range [0, UINT_MAX]
Registers provided to the user to store cutom values.
Writing to this registers may take some time.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CustomRegister_3", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CustomRegister_3", "10"
);
```

Emit:

- NITConfigObserver.paramChanged 'CustomRegister_N'.

Throw:

- NITException if the string value or the numeric value is out of range.

The following parameters are for debug purpose only and must not be modified by the users.

Parameters

CamSelectDataSrc **Guru**

The following parameters are the boostrap GigeVision parameters. They follow the GigE Vision® Specification version 2.0.
We invite you to consult the specification for more details.

Parameters

PayloadSize **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX] bytes
Maximum number of bytes transferred for each image on the stream channel, including any end-ofline, end-of-frame statistics or other stamp data. This is the maximum total size of data payload for a block.
UDP and GVSP headers are not considered.
Data leader and data trailer are not included.
This is mainly used by the application software to determine size of image buffers to allocate (largest buffer possible for current mode of operation).
For example, an image with no statistics or stamp data as PayloadSize equals to (width x height x pixelsize) in bytes.

GevVersion **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Version of the GigE Standard with which the device is compliant
The two most-significant bytes are allocated to the major number of the version,
the two least-significant bytes for the minor number.

GevDeviceMode **Guru ReadOnly**
Integer in the range [0, UINT_MAX]
Information about device mode of operation.
Returns Big Endian and UTF8.

GevDeviceMACAddressHigh **Beginner ReadOnly**
Integer in the range [0, UINT_MAX]
The two most-significant bytes of this area are reserved and will return 0.
Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
The MAC address of the camera can be read at the rear of the device.

GevDeviceMACAddressLow **Beginner ReadOnly**
Integer in the range [0, UINT_MAX]
Lower 4 bytes of the MAC address.
The MAC address of the camera can be read at the rear of the device.

GevNetworkCapability **Expert ReadOnly**
Integer in the range [0, UINT_MAX]
Supported IP Configuration and PAUSE schemes.
Returns LLA|DHCP|PersistentIp.

GevNetworkConfiguration **Expert**
Activated IP Configuration and PAUSE schemes.
This parameter can take these values:

String value	Numeric value	Comment
LLA	4	Local Link(APIPA) only
LLA_Persistent_IP	5	Local Link and Persistent IP
LLA_DHCP	6	Local Link and DHCP(default)
All	7	All modes

If set to LLA: on connection, the camera negociate an LLA Ip Address.
If set to LLA_DHCP: on connection, the camera try to contact a DHCP server to acquire an Ip address. If no DHCP is present, the device fallback to LLA negociation.
If set to LLA_Persistent_IP: on connection, the camera try to negociate an LLA Ip address. If the LLA negociation fail, the device fallback to Persistent IP.
If set to All: The device try to connect with DHCP,

the with LLA and finally takes the persistent IP. Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", "LLA" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", 5 );
```

Emit:

- NITConfigObserver.paramChanged 'GevNetworkConfiguration'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

GevCurrentIPAddress

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current IP address of this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentSubnetMask

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current subnet mask used by this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentDefaultGateway

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current default gateway used by this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevManufacturerName

Beginner **ReadOnly**
Provides the name of the manufacturer of the device.
Returns the string: "New-Imaging-Technologies"

GevModelName

Beginner **ReadOnly**
Provides the model name of the device.

Returns the string "NSC1003GigE".

GevDeviceVersion

Beginner ReadOnly
Provides the version of the device.
Returns a string whith the format "MM.mm"
Example: "02.07"

GevManufacturerInfo

Beginner ReadOnly
Provides extended manufacturer information about the device.
Returns the string "Camera NSC1003 from NIT"

GevSerialNumber

Beginner ReadOnly
Contains the serial number of the device.
It can be used to identify the device.
Returns a string with the format "NNNNNN"
Example: "200162"

GevUserDefinedName

Beginner
Null terminated string with maximum 16 characters(terminating null included).
This string contains a user-programmable name to assign to the device.
It can be used to identify the device.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevUserDefinedName",
"MyCamera" );
```

Emit:

- NITConfigObserver.paramChanged 'GevUserDefinedName'.

Throw:

- NITException if the string is to long.

GevFirstURL

Guru ReadOnly
This register stores the first URL to the XML device description file.
The first URL is used as the first choice by the application to retrieve the XML device description file.
Returns a string with the format <filename>;<address>;<file size>
Example: "local:adr_nios_1003.zip;A200;1938"

GevSecondURL

Guru ReadOnly
This register stores the second URL to the XML device description file.

This URL is an alternative if the application was unsuccessful to retrieve the device description file using the first URL.

GevNumberOfInterfaces **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
This register indicates the number of network interfaces supported by this device.
This is generally equivalent to the number of Ethernet connectors on the device, except when Link Aggregation is used. In this case, the physical interfaces regrouped by the Aggregator are counted as one virtual interface.
Returns allways 1.

GevLinkSpeed **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Value indicating current Ethernet link speed in Mbits per second. Gigabit ethernet returns 1000.

GevMessageChannelCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of message channels supported by this device.
Returns allways 0.

GevStreamChannelCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of stream channels supported by this device.
Returns allways 1.

GevActiveLinkCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of physical links that are currently active. A link is considered active as soon as it is connected to another Ethernet device. This happens after Ethernet link negotiation is completed.
Returns allways 1.

GevStreamChannelCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
First Stream Channel Capability register.
Returns SCSPx_supported|legacy_16bit_block_id_supported.

GevMessageChannelCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the capabilities of the message channel.

It lists which of the non-mandatory message channel features are supported.
Returns allways 0.

GevGVCPCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
This is a capability register indicating which one of the non-mandatory GVCP features are supported by this device. When supported, some of these features are enabled through the GVCP Configuration register.
Returns user_defined_name|serial_number|link_speed_register|WRITEMEM.

GevHeartbeatTimeout

Guru
Integer in the range [0, UINT_MAX] milliseconds.
Indicates the current heartbeat timeout in milliseconds.
Default is 3000 msec.
Internally, the heartbeat is rounded according to the clock used for heartbeat.
The heartbeat timeout shall have a minimum precision of 100 ms.
The minimal value is 500 ms.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout", 2000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout", "1500" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevGVCPPendingTimeout

Expert **ReadOnly**
Integer in the range [0, UINT_MAX] milliseconds.
Pending Timeout to report the longest GVCP command execution time before issuing a PENDING_ACK. If PENDING_ACK is not supported, then this is the worstcase execution time before command completion.
Returns 300000.

GevPhysicalLinkConfigurationCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the physical link configuration supported

by this device. Returns 0.

GevPhysicalLinkConfiguration

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the currently active physical link configuration.
Returns 0.

GevCCP

Guru
Integer in the range [0, UINT_MAX]
This register is used to grant privilege to an application.
Only one application is allowed to control the device. This application is able to write into device's registers. Other applications can read device's register only if the controlling application does not have the exclusive privilege.

On construction of the NITDevice the SDK set this parameter to exclusive access.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevCCP", 2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevCCP", "2" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCPHostPort

Guru
Integer in the range [0, UINT_MAX]
Stream Channel Host Port register.
The host port to which a stream channel must send data stream.
Setting this value to 0 closes the stream channel.
The frame number is reset to 1 when the stream channel is opened.

This parameter is managed by the SDK.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPHostPort", 12000
```

```
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPHostPort",
"14200" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCSP

Guru ReadOnly
Integer in the range [0, UINT_MAX]
Stream Channel Source Port register.
The port on which the stream channel emits packets.

GevSCDA

Guru
Integer in the range [0, UINT_MAX]
Stream Channel destination Ip address of the receiving host.
The Ip address to which a stream channel must send data stream.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

On construction of the NITDevice the SDK set this parameter to the IP address of the host.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCDA", 3232236047
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCDA", "3232236047"
);
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentIPAddress

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent IP address for the given network interface.
Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address.

Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentSubnetMask

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentDefaultGateway

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCPSPacketSize

Guru
Integer in the range [0, 9000] bytes
Size of the stream packets to transmit during acquisition.

On construction of the NITDevice the SDK negociate the highest possible packet size with the device.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPSPacketSize",
8000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPSPacketSize",
"8000" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

TimeStamp_frequency_high

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
High part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the most significant bytes.

TimeStamp_frequency_low

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Low part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the least significant bytes.

TimestampValueHigh

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (most significant bytes). The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current

frame.

TimestampValueLow

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (least significant bytes) The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current frame.

TimestampControl

Expert **WriteOnly**
Integer in the range [0, 3]
Write 1 to reset timestamp 64-bit counter to 0.
Write 2 to latch current timestamp counter into TimestampValue registers.
Write 3 to first latch current timestamp counter into TimeStampValue registers and then reset.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TimestampControl", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "TimestampControl", "3"
);
```

Throw:

- NITException if the string value or the numeric value is out of range.

18. MC1003-1Gx color series parameters

NITLibrary 3.x series support NSC1003GigE cameras with firmware version above or equal to 2.7.0.

The following parameters are supported:
Where not otherwise indicated parameters are Read/Write.

Parameters

WidthMax

Expert **ReadOnly**
Integer value in the range [1, 1280] pixels.

Updated each time OffsetX, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Width' parameter.

HeightMax

Expert **ReadOnly**
Integer value in the range [1, 1024] pixels.

Updated each time OffsetY, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Height' parameter.

Width

Beginner
Integer value in the range [2, WidthMax] by steps of 2 pixels.
The value you can set is constrained by the frame rate and 'WidthMax'.

Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Width", 600 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Width", "600" );
```

Emit:

- NITConfigObserver.paramChanged 'Width'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

Height

Beginner
Integer value in the range [2, HeightMax] by steps of 2 pixels.
The value you can set is constrained by the frame rate and 'HeightMax'.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Height", 500 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Height", "500" );
```

Emit:

- NITConfigObserver.paramChanged 'Height'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

OffsetX

Beginner
Integer value in the range [0, 1270] by steps of 10 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetX", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetX", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetX'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

OffsetY

Beginner
Integer value in the range [0, 1010] by steps of 10 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetY", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetY", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetY'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

PixelFormat

Expert
While this parameter is writeable, its use is limited.
The pixel format is in fact changed by the 'OutputType' parameter.
The following pixel formats are available:

String value	Numeric value	Comment	NITFrame pixel type
Mono8	17301505	Set if 'OutputType' is set to 'Correction_Grey'	FLOAT
Mono14	17825829	Set if 'OutputType' is set to 'RAW'	FLOAT
RGB8	35127316	Set if 'OutputType' is set to 'Correction_Color'	RGBA
BayerGR8	17301512	Set if 'OutputType' is set to 'Correction_Bayer'	FLOAT
BayerGR16	17825838	Set if 'OutputType' is set to 'RAW_Bayer'	FLOAT

Emit:

- NITConfigObserver.paramChanged 'PixelFormat'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

OutputType

Beginner
Permits to set the desired output type.
In all cases, if NUC is active it is applied before the OutputType.
The following output types are available:

String value	Numeric value	Comment
Correction_Grey	0	Activate gain control. Change 'PixelFormat' to Mono8
Correction_Bayer	1	Change 'PixelFormat' to BayerGR8
Correction_Color	2	Activate debayering. Change 'PixelFormat' to RGB8
RAW	3	Activate gain control. Change 'PixelFormat' to Mono14
RAW_Bayer	4	Deactivate gain control and debayering. Change 'PixelFormat' to BayerGR16

RAW is the output of the sensor with application of the NUC if it is active.
RAW_Bayer is identical to RAW. It is only an hint. Some viewers use this information to apply a custom debayering before display.
Correction_Bayer is identical to RAW clamped to 8bits per pixel. It is only an hint. Some viewers use this information to apply a custom debayering before display.
Correction_Color applies debayering.
Correction_Grey applies gain control to RAW data.

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "OutputType",
"Correction_Grey" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "OutputType", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'OutputType'.
- NITConfigObserver.paramChanged 'PixelFormat'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionMode

Beginner
This parameter is writable but can take only one

value.

String value	Numeric value	Comment
Continuous	0	Frames are continuously outputted until AcquisitionStop is called

Emit:

- NITConfigObserver.paramChanged 'AcquisitionMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionStart

Beginner
This parameter is a Command. The only possible value is 1.
Start to output frames.

This command is a direct command to the camera. It bypasses the variables initialisations performed by the SDK.
Prefer NITDevice.start(), NITDevice.captureNFrames() or NITDevice.captureForDuration() to start the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStart", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStart", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the camera is already started.

AcquisitionStop

Beginner
This parameter is a Command. The only possible value is 1.
Stop to output frames.

This command is a direct command to the camera. It bypass the variables initialisations performed by the SDK.
Prefer NITDevice.stop() to stop the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStop", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStop", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionFrameRate

Beginner
Set the frame rate.
Float value in the range [1.0,MaxFps] Hz
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionFrameRate",
100.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionFrameRate",
"100.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AcquisitionFrameRate'.
- NITConfigObserver.paramChanged 'MaxExposure'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current ExposureTime or resolution.

MaxFPS

Beginner ReadOnly
Float value in the range [1,10000] Hz.
Updated each time the resolution or the ExposureTime are changed.
This is the maximum value which can be actually set for the 'AcquisitionFrameRate' parameter.

ReadOutMode

Beginner
Set the acquisition mode of the device.
This parameter can take these values:

String value	Numeric value	Comment
Global	0	The entire frame is exposed and captured at the same instant
Rolling	1	The frame is exposed and captured line by line
Differential	2	The output frame is the difference of two succesive frames in Global Shutter

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "ReadOutMode", "Rolling" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "ReadOutMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'ReadOutMode'.
- NITConfigObserver.paramChanged 'MaxExposure'.

NITConfigObserver.paramChanged 'MaxFps'. **Throw:**

- NITException if the string value or the numeric value doesn't exist.

TriggerDirection

Beginner
Set the direction of the trigger signal.
The following directions are available:

String value	Numeric value	Comment
Internal	0	The trigger signal is generated and outputted by the camera each time a frame is outputted
External	1	The trigger signal is external and start acquisition of one

		frame

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerDirection",
"External" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerDirection", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerDirection'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

TriggerOut1 **Beginner**

TriggerOut2 **Beginner**

Set the behaviour of the GPO signals Out1 and Out2.
The following behaviours are available:

String value	Numeric value	Comment
Exposing	0	Output is high when camera exposes
TriggerReady	1	Output is high when camera is ready to receive an external trigger signal
TriggerInput	2	Output mirrors the external trigger signal
ReadOut	3	Output is high when camera read pixels
Imaging	4	Output is high during exposition and readout
GPOLowLevel	5	Output is forced low
GPOHighLevel	6	Output is forced high

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerOut1",
```

```
"TriggerInput" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerOut1", 4 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerOut1' or 'TriggerOut2'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

ExposureTime

Beginner
Set the duration of the exposition.
Integer value in the range [100,MaxExposureTime] microseconds
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "ExposureTime", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "ExposureTime", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'ExposureTime'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current 'AcquisitionFrameRate'.

MaxExposureTime

Beginner ReadOnly
Integer value in the range [0,65535] microseconds.
Updated each time the resolution or the 'AcquisitionFrameRate' are changed.
This is the maximum value which can be actually set for the 'ExposureTime' parameter.

CamLoadConfiguration

Beginner

This parameter is a Command. The only possible value is 1.
Load the parameters stored in flash.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamLoadConfiguration", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamLoadConfiguration", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

CamSaveConfiguration

Beginner
This parameter is a Command. The only possible value is 1.
Save the current parameters in flash.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamSaveConfiguration", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamSaveConfiguration", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

CamResetParameter

Beginner
This parameter is a Command. The only possible value is 1.
Reset the current parameters to their default value.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamResetParameter", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamResetParameter", "1" );
```

Throw:

- NITException if the string value or the

numeric value doesn't exist.

NIOS_IS_BUSY

Beginner **ReadOnly**
Running some commands like eg. 'CamSaveConfiguration' access the flash and can take some time.
During this time new commands or parameter setting are not taken into account.
This parameter permits to know if it is safe to send commands and parameters to the camera.
If returned value is 1: The camera is busy, don't send commands or parameters.
If returned value is 0: The camera is not busy, you can safely send commands or parameters.

This information is for documentation only.
NITLibrary look after the busyness of the camera before sending parameters or commands.

NucActivate

Beginner
Activate or Deactivate the embedded Nuc processing.
If embedded Nuc is activated, library Nuc is deactived.
The Nuc applied is the Nuc in the current selected table. see nucBpr
This parameter can take these two values:

String value	Numeric value	Comment
Off	0	Embedded NUC is not active
On	1	Embedded NUC is active

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucActivate", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucActivate", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'NucActivate'.

Throw:

- NITException if the string value or the

numeric value doesn't exist.

NucCompute1Pts

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point for this device with the current configuration, the gain is untouched. The resulting Nuc is set in the current selected table. This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute1Pts", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute1Pts", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSelectTable

Beginner
Select the current Nuc table who will be applied to the captured frames.
At delivery time of the camera, NUC tables are empty, as NUC is not mandatory for CMOS sensors.
This parameter can take these values:

String value	Numeric value
NONE	0
NUC1	1
NUC2	2
NUC3	3
NUC4	4

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSelectTable", "NUC2"
);
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucSelectTable", 2 );
```

Emit:

- NITConfigObserver.paramChanged

'NucSelectTable'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucResetTable

Beginner
This parameter is a Command. The only possible value is 1.
Reset the content of the current selected Nuc table entry in memory(the Nuc table in flash is untouched).
The low point is set to all 0 and the gain matrix is set to all ones.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucResetTable", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucResetTable", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSaveInFlash

Beginner
This parameter is a Command. The only possible value is 1.
Save the content of the current selected Nuc table entry in flash memory(the Nuc table in flash is overridden).
As we are writing to flash memory, this operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSaveInFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucSaveInFlash", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucLoadFromFlash

Beginner
This parameter is a Command. The only possible value is 1.
Load the content of the current selected Nuc table entry from flash to memory(the Nuc table in memory is overridden).
As we are reading from flash memory, this operation may take some time.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucLoadFromFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucLoadFromFlash", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcMode

Beginner
Set the method used to apply gain control.
The gain control is active if 'OutputType' is not 'RAW'.
This parameter can take these two values:

String value	Numeric value	Comment
Auto	0	The gain and offset are automatically calculated
Manuel	1	The gain and offset are preset

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcMode", "Auto" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged

'AgcMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcManualGain

Beginner
Set the gain used when gain control is set to manual.
Float value in the range [0.0, 1.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualGain", 0.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualGain", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualGain'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcManualOffset

Beginner
Set the offset used when gain control is set to manual.
Float value in the range [0.0, 16383.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualOffset", 10.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualOffset", "10.2"
);
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualOffset'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcAutoMeanStdThreshold

Beginner
Set the threshold used when gain control is set to auto.
Float value in the range [0.0, 16383.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcAutoMeanStdThreshold", 10.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcAutoMeanStdThreshold", "10.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcAutoMeanStdThreshold'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcSmoothingMethod

Beginner
Set the smoothing coefficient used to calculate gain control.
Smoothing is done with a temporal averaging on gain and offset.
'Slow' averages on more frames.
This parameter is only effective in 'Auto' 'AgcMode'.
This parameter can take these values:

String value	Numeric value
Slow	0
Medium	1
Fast	2

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcSmoothingMethod", "Auto" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcSmoothingMethod", 1 );
```

Emit:

- NITConfigObserver.paramChanged

'AgcSmoothingMethod'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcAutoGain

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated gain for the last frame when gain control is set to auto.

AgcAutoOffset

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated offset for the last frame when gain control is set to auto.

AgcStatisticsFmean

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the mean pixel value of the last frame.

AgcStatisticsFstd

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the standard deviation value of the last frame.

AgcStatisticsWmin

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the minimum pixel value of the last frame.

AgcStatisticsWmax

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the maximum pixel value of the last frame.

AgcRoiMode

Beginner
Set the rectangular region of the frame on which the gain control is calculated.
This parameter can take these values:

String value	Numeric value	Comment
FULL_FRAME	0	Region of interest is the entire frame
HALF_FRAME	1	Region of interest is centered in the frame
QUARTER_FRAME	2	Region of interest is centered in the frame
CUSTOM_FRAME	3	Region of interest is anywhere in the

		frame
		The rectangle is set by the parameters:
		- AgcRoiCustomLeft
		- AgcRoiCustomTop
		- AgcRoiCustomRight
		- AgcRoiCustomBottom

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiMode", "HALF_FRAME" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcRoiCustomTop **Beginner** Integer in the range [0, UINT_MAX]

AgcRoiCustomBottom **Beginner** Integer in the range [0, UINT_MAX]

AgcRoiCustomLeft **Beginner** Integer in the range [0, UINT_MAX]

AgcRoiCustomRight **Beginner** Integer in the range [0, UINT_MAX]

Set the rectangular region of the frame on which the gain control is calculated when the 'RoiMode' parameter is set to 'CUSTOM_FRAME'.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiCustomTop", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcRoiCustomBottom", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiCustomTop', 'AgcRoiCustomBottom', 'AgcRoiCustomLeft' or 'AgcRoiCustomRight'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcRoiDraw

Beginner
Draw a rectangular frame in the image representing the rectangular region where the gain control is calculated.
This parameter can take this two values:

String value	Numeric value	Comment
Off	0	Rectangle not drawn
On	1	Rectangle drawn

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiDraw", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiDraw", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiDraw'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

GainChroma

Beginner
Integer in the range [0, 255]
Set the gain factor for the debayering process.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GainChroma", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GainChroma", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'GainChroma'.

Throw:

- NITException if the string value or the numeric value is out of range.

Gamma

Beginner
Float in the range [0, 3]
Set gamma factor. A value of 1 imply no gamma correction
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "Gamma", 1.5 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Gamma", "1.4" );
```

Emit:

- NITConfigObserver.paramChanged 'Gamma'.

Throw:

- NITException if the string value or the numeric value is out of range.

GammaThd

Beginner
Integer in the range [0, 255]
Set the center of the gamma.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GammaThd", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GammaThd", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'GammaThd'.

Throw:

- NITException if the string value or the numeric value is out of range.

CustomReg_1 **Expert**

CustomReg_2 **Expert**

CustomReg_3 **Expert**

CustomReg_4 **Expert**

CustomReg_5 **Expert**

CustomReg_6 **Expert**

CustomReg_7 **Expert**

CustomReg_8 **Expert**

CustomReg_9 **Expert**

CustomReg_10 **Expert**

Integer in the range [0, UINT_MAX]
Registers provided to the user to store cutom values.
Writing to this registers may take some time.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CustomRegister_3", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CustomRegister_3", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'CustomRegister_N'.

Throw:

- NITException if the string value or the numeric value is out of range.

The following parameters are for debug purpose only and must not be modified by the users.

Parameters

CamSelectDataSrc **Guru**

The following parameters are the boostrap GigeVision parameters. They follow the GigE Vision® Specification version 2.0.
We invite you to consult the specification for more details.

Parameters

PayloadSize **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX] bytes
Maximum number of bytes transferred for each image on the stream channel, including any end-ofline, end-of-frame statistics or other stamp data. This is the maximum total size of data payload for a block.
UDP and GVSP headers are not considered.
Data leader and data trailer are not included.
This is mainly used by the application software to determine size of image buffers to allocate (largest buffer possible for current mode of operation).
For example, an image with no statistics or stamp data as PayloadSize equals to (width x height x pixelsize) in bytes.

GevVersion **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Version of the GigE Standard with which the device is compliant
The two most-significant bytes are allocated to the major number of the version,
the two least-significant bytes for the minor number.

GevDeviceMode **Guru** **ReadOnly**
Integer in the range [0, UINT_MAX]
Information about device mode of operation.
Returns Big Endian and UTF8.

GevDeviceMACAddressHigh **Beginner** **ReadOnly**
Integer in the range [0, UINT_MAX]
The two most-significant bytes of this area are reserved and will return 0.
Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
The MAC address of the camera can be read at the rear of the device.

GevDeviceMACAddressLow **Beginner** **ReadOnly**
Integer in the range [0, UINT_MAX]
Lower 4 bytes of the MAC address.
The MAC address of the camera can be read at the rear of the device.

GevNetworkCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Supported IP Configuration and PAUSE schemes.
Returns LLA|DHCP|PersistentIp.

GevNetworkConfiguration **Expert**
Activated IP Configuration and PAUSE schemes.
This parameter can take these values:

String value	Numeric value	Comment
LLA	4	Local Link(APIPA) only
LLA_Persistent_IP	5	Local Link and Persistent IP
LLA_DHCP	6	Local Link and DHCP(default)
All	7	All modes

If set to LLA: on connection, the camera negociate an LLA Ip Address.
If set to LLA_DHCP: on connection, the camera try to contact a DHCP server to acquire an Ip address. If no DHCP is present, the device fallback to LLA negociation.
If set to LLA_Persistent_IP: on connection, the camera try to negociate an LLA Ip address. If the LLA negociation fail, the device fallback to Persistent IP.
If set to All: The device try to connect with DHCP, the with LLA and finally takes the persistent IP.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", "LLA" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", 5 );
```

Emit:

- NITConfigObserver.paramChanged 'GevNetworkConfiguration'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

GevCurrentIPAddress

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current IP address of this device on its first interface.
Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentSubnetMask

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current subnet mask used by this device on its first interface.
Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentDefaultGateway

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current default gateway used by this device on its first interface.
Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevManufacturerName

Beginner **ReadOnly**
Provides the name of the manufacturer of the device.
Returns the string: "New-Imaging-Technologies"

GevModelName

Beginner **ReadOnly**
Provides the model name of the device.
Returns the string "NSC1003cGigE".

GevDeviceVersion

Beginner **ReadOnly**
Provides the version of the device.
Returns a string whith the format "MM.mm"
Example: "02.07"

GevManufacturerInfo

Beginner **ReadOnly**
Provides extended manufacturer information about the device.
Returns the string "Camera NSC1003c from NIT"

GevSerialNumber

Beginner **ReadOnly**
Contains the serial number of the device.

It can be used to identify the device.
Returns a string with the format "NNNNNN"
Example: "200162"

GevUserDefinedName

Beginner
Null terminated string with maximum 16 characters(terminating null included).
This string contains a user-programmable name to assign to the device.
It can be used to identify the device.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevUserDefinedName", "MyCamera" );
```

Emit:

- NITConfigObserver.paramChanged 'GevUserDefinedName'.

Throw:

- NITException if the string is to long.

GevFirstURL

Guru ReadOnly
This register stores the first URL to the XML device description file.
The first URL is used as the first choice by the application to retrieve the XML device description file.
Returns a string with the format <filename>;<address>;<file size>
Example: "local:adr_nios_1003.zip;A200;1938"

GevSecondURL

Guru ReadOnly
This register stores the second URL to the XML device description file.
This URL is an alternative if the application was unsuccessful to retrieve the device description file using the first URL.

GevNumberOfInterfaces

Expert ReadOnly
Integer in the range [0, UINT_MAX]
This register indicates the number of network interfaces supported by this device.
This is generally equivalent to the number of Ethernet connectors on the device, except when Link Aggregation is used. In this case, the physical interfaces regrouped by the Aggregator are counted as one virtual interface.

Returns allways 1.

GevLinkSpeed **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Value indicating current Ethernet link speed in Mbits per second. Gigabit ethernet returns 1000.

GevMessageChannelCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of message channels supported by this device.
Returns allways 0.

GevStreamChannelCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of stream channels supported by this device.
Returns allways 1.

GevActiveLinkCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of physical links that are currently active. A link is considered active as soon as it is connected to another Ethernet device. This happens after Ethernet link negotiation is completed.
Returns allways 1.

GevStreamChannelCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
First Stream Channel Capability register.
Returns SCSPx_supported|legacy_16bit_block_id_supported.

GevMessageChannelCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the capabilities of the message channel. It lists which of the non-mandatory message channel features are supported.
Returns allways 0.

GevGVCPCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
This is a capability register indicating which one of the non-mandatory GVCP features are supported by this device. When supported, some of these features are enabled through the GVCP Configuration register.
Returns user_defined_name|serial_number|link_speed_re

gister|WRITEMEM.

GevHeartbeatTimeout

Guru
Integer in the range [0, UINT_MAX] milliseconds.
Indicates the current heartbeat timeout in milliseconds.
Default is 3000 msec.
Internally, the heartbeat is rounded according to the clock used for heartbeat.
The heartbeat timeout shall have a minimum precision of 100 ms.
The minimal value is 500 ms.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout",
2000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout",
"1500" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevGVCPPendingTimeout

Expert **ReadOnly**
Integer in the range [0, UINT_MAX] milliseconds.
Pending Timeout to report the longest GVCP command execution time before issuing a PENDING_ACK. If PENDING_ACK is not supported, then this is the worstcase execution time before command completion.
Returns 300000.

GevPhysicalLinkConfigurationCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the physical link configuration supported by this device. Returns 0.

GevPhysicalLinkConfiguration

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the currently active physical link configuration.
Returns 0.

GevCCP

Guru
Integer in the range [0, UINT_MAX]
This register is used to grant privilege to an application.
Only one application is allowed to control the

device. This application is able to write into device's registers. Other applications can read device's register only if the controlling application does not have the exclusive privilege.

On construction of the NITDevice the SDK set this parameter to exclusive access.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevCCP", 2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevCCP", "2" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCPHostPort

Guru
Integer in the range [0, UINT_MAX]
Stream Channel Host Port register.
The host port to which a stream channel must send data stream.
Setting this value to 0 closes the stream channel.
The frame number is reset to 1 when the stream channel is opened.

This parameter is managed by the SDK.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPHostPort", 12000
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPHostPort",
"14200" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCSP

Guru ReadOnly
Integer in the range [0, UINT_MAX]
Stream Channel Source Port register.

The port on which the stream channel emits packets.

GevSCDA

Guru
Integer in the range [0, UINT_MAX]
Stream Channel destination Ip address of the receiving host.
The Ip address to which a stream channel must send data stream.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

On construction of the NITDevice the SDK set this parameter to the IP address of the host.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCDA", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCDA", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentIPAddress

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent IP address for the given network interface.
Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", "3232236047" );
```

Throw:

- NITException if the string value or the

numeric value is out of range.

GevPersistentSubnetMask

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address. Example: 192.168.2.15 is coded 0xC0A8020F. Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentDefaultGateway

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address. Example: 192.168.2.15 is coded 0xC0A8020F. Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCPSPacketSize

Guru
Integer in the range [0, 9000] bytes
Size of the stream packets to transmit during acquisition.

On construction of the NITDevice the SDK negociate the highest possible packet size with the device.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPSPacketSize", 8000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPSPacketSize", "8000" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

TimeStamp_frequency_high **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
High part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the most significant bytes.

TimeStamp_frequency_low **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Low part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the least significant bytes.

TimestampValueHigh **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (most significant bytes). The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current frame.

TimestampValueLow **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (least significant bytes) The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current frame.

TimestampControl **Expert** **WriteOnly**
Integer in the range [0, 3]
Write 1 to reset timestamp 64-bit counter to 0.
Write 2 to latch current timestamp counter into TimestampValue registers.

Write 3 to first latch current timestamp counter into TimeStampValue registers and then reset. Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TimestampControl", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "TimestampControl", "3" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

19. WiDySWIR 640G-SE series parameters

NITLibrary 3.x series support NSC1201GigE cameras with firmware version above or equal to 2.7.0.

The following parameters are supported:
Where not otherwise indicated parameters are Read/Write.

Parameters

WidthMax

Expert **ReadOnly**
Integer value in the range [8, 640] pixels.
Updated each time OffsetX, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Width' parameter.

HeightMax

Expert **ReadOnly**
Integer value in the range [8, 512] pixels.
Updated each time OffsetY, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Height' parameter.

Width

Beginner
Integer value in the range [16, WidthMax] by steps of 8 pixels.
The value you can set is constrained by the frame rate and 'WidthMax'.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Width", 600 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Width", "600" );
```

Emit:

- NITConfigObserver.paramChanged 'Width'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

Height

Beginner
Integer value in the range [16, HeightMax] by steps

of 8 pixels.
The value you can set is constrained by the frame rate and 'HeightMax'.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Height", 500 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Height", "500" );
```

Emit:

- NITConfigObserver.paramChanged 'Height'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

OffsetX

Beginner
Integer value in the range [0, 640] by steps of 4 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetX", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetX", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetX'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

OffsetY

Beginner
Integer value in the range [0, 512] by steps of 4 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetY", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetY", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetY'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

PixelFormat

Beginner
While this parameter is writeable, its use is limited. The pixel format is in fact changed by the 'OutputType' parameter.
The following pixel formats are available:

String value	Numeric value	Comment	NITFrame pixel type
Mono8	17301505	Set if 'OutputType' is set to 'GREY'	FLOAT
Mono14	17825829	Set if 'OutputType' is set to 'RAW'	FLOAT
RGB8	35127316	Set if 'OutputType' is not 'GREY' and not 'RAW'	RGBA

Emit:

- NITConfigObserver.paramChanged 'PixelFormat'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

OutputType

Beginner
Permits to set the desired output type.
In all cases, if NUC and/or BPR are active they are applied before the OutputType.

The following output types are available:

String value	Numeric value	Comment
GREY	0	Activate gain control. Change 'PixelFormat' to Mono8
RAINBOW	1	Apply a color map. Activate gain control. Change 'PixelFormat' to RGB8
STEEL	2	
LOCKIN	3	
THRESHOLD	4	
HOT	5	
RAW	7	Deactivate gain control. Change 'PixelFormat' to Mono14

RAW is the output of the sensor with application of the NUC and BPR if they are active.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "OutputType", "GREY" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "OutputType", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'OutputType'.
- NITConfigObserver.paramChanged 'PixelFormat'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

Flip_Image

Beginner
Change the orientation of the frame.
The following orientations are available:

String value	Numeric value	Comment
NoFlip	0	No flip

FlipH	1	Flip frame horizontally
FlipV	2	Flip frame vertically
FlipHV	3	Flip frame horizontally and vertically

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "Flip_Image", "FlipH" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "Flip_Image", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Flip_Image'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionMode

Beginner
This parameter is writable but can take only one value.

String value	Numeric value	Comment
Continuous	0	Frames are continuously outputted until AcquisitionStop is called

Emit:

- NITConfigObserver.paramChanged 'AcquisitionMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionStart

Beginner
This parameter is a Command. The only possible value is 1.
Start to output frames.

This command is a direct command to the camera. It bypass the variables initialisations performed by the

SDK.
Prefer NITDevice.start(), NITDevice.captureNFrames() or NITDevice.captureForDuration() to start the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStart", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStart", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the camera is already started.

AcquisitionStop

Beginner
This parameter is a Command. The only possible value is 1.
Stop to output frames.

This command is a direct command to the camera. It bypass the variables initialisations performed by the SDK.
Prefer NITDevice.stop() to stop the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStop", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStop", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionFrameRate

Beginner
Set the frame rate.
Float value in the range [1.0,MaxFps] Hz
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionFrameRate",
```

```
100.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionFrameRate",
"100.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AcquisitionFrameRate'.
- NITConfigObserver.paramChanged 'MaxExposure'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current ExposureTime or resolution.

MaxFPS

Beginner ReadOnly
Float value in the range [1,10000] Hz.
Updated each time the resolution or the ExposureTime are changed.
This is the maximum value which can be actually set for the 'AcquisitionFrameRate' parameter.

TriggerDirection

Beginner
Set the direction of the trigger signal.
The following directions are available:

String value	Numeric value	Comment
Internal	0	The trigger signal is generated and outputted by the camera each time a frame is outputted
External	1	The trigger signal is external and start acquisition of one frame

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerDirection",
"External" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerDirection", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerDirection'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

TriggerExpo

Beginner
Set how the exposure duration is managed when 'TriggerDirection' is set to 'External'.
The following behaviours are available:

String value	Numeric value	Comment
TTLWidth	0	Exposure duration is the trigger width(signal high)
ExposureTime	1	Exposure duration is the value of 'ExposureTime' parameter

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerExpo", "TTLWidth" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerExpo", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerExpo'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

TriggerOut1

Beginner

TriggerOut2

Beginner
Set the behaviour of the GPO signals Out1 and Out2.
The following behaviours are available:

String value	Numeric value	Comment
Exposing	0	Output is high when camera exposes
TriggerReady	1	Output is high when camera is ready to receive an external trigger signal
TriggerInput	2	Output mirrors the external trigger signal
ReadOut	3	Output is high when camera read pixels
Imaging	4	Output is high during exposition and readout
GPOLowLevel	5	Output is forced low
GPOHighLevel	6	Output is forced high

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerOut1",
"TriggerInput" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerOut1", 4 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerOut1' or 'TriggerOut2'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

ExposureTime

Beginner
Set the duration of the exposition.
Float value in the range [1.0,MaxExposureTime] microseconds
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "ExposureTime", 100.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "ExposureTime", "100.2" );
         dev.setParamValueOf( "ExposureTime", "100.2us"
```

```
);
```

Emit:

- NITConfigObserver.paramChanged 'ExposureTime'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current 'AcquisitionFrameRate'.

MaxExposureTime

Beginner ReadOnly
Float value in the range [1,65535] microseconds.
Updated each time the resolution or the 'AcquisitionFrameRate' are changed.
This is the maximum value which can be actually set for the 'ExposureTime' parameter.

CamBoardTemperature

Beginner ReadOnly
Float value in the range [-200.0,200.0] °C.
Temperature of the sensor.

CamLoadConfiguration

Beginner
This parameter is a Command. The only possible value is 1.
Load the parameters stored in flash.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamLoadConfiguration", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamLoadConfiguration",
"1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

CamSaveConfiguration

Beginner
This parameter is a Command. The only possible value is 1.
Save the current parameters in flash.
Use the numeric form of setParamValueOf to set

these parameters.

```
Example: dev.setParamValueOf( "CamSaveConfiguration", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamSaveConfiguration",
"1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

CamResetParameter

Beginner
This parameter is a Command. The only possible value is 1.
Reset the current parameters to their default value.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamResetParameter", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamResetParameter", "1"
);
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NIOS_IS_BUSY

Beginner **ReadOnly**
Running some commands like eg. 'CamSaveConfiguration' access the flash and can take some time.
During this time new commands or parameter setting are not taken into account.
This parameter permits to know if it is safe to send commands and parameters to the camera.
If returned value is 1: The camera is busy, don't send commands or parameters.
If returned value is 0: The camera is not busy, you can safely send commands or parameters.

This information is for documentation only.
NITLibrary look after the busyness of the camera before sending parameters or commands.

NucActivate

Beginner
Activate or Deactivate the embedded Nuc processing.
If embedded Nuc is activated, library Nuc is

deactived.
The Nuc applied is the Nuc in the current selected table. see nucBpr
This parameter can take these two values:

String value	Numeric value	Comment
Off	0	Embedded NUC is not active
On	1	Embedded NUC is active

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucActivate", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucActivate", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'NucActivate'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucCompute1Pts

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point for this device with the current configuration, the gain is untouched. The resulting Nuc is set in the current selected table. This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute1Pts", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute1Pts", "1" );
```

Throw:

- NITException if the string value or the

numeric value doesn't exist.

NucCompute2Pts1

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point for this device with the current configuration, the gain matrix is set to all ones.
The resulting Nuc is set in the current selected table.
This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute2Pts1", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute2Pts1", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucCompute2Pts2

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point and gain matrix for this device with the current configuration.
The resulting Nuc is set in the current selected table.
This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute2Pts2", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute2Pts2", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSelectTable

Beginner
Select the current Nuc table who will be applied to the captured frames.
At delivery time of the camera, 4 tables are provided for different exposure times.
This parameter can take these values:

String value	Numeric value	Exposure time at delivery
NONE	0	
NUC1	1	500 µs
NUC2	2	2000 µs
NUC3	3	5000 µs
NUC4	4	10000 µs

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSelectTable", "NUC2" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucSelectTable", 2 );
```

Emit:

- NITConfigObserver.paramChanged 'NucSelectTable'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucResetTable

Beginner
This parameter is a Command. The only possible value is 1.
Reset the content of the current selected Nuc table entry in memory(the Nuc table in flash is untouched).
The low point is set to all 0 and the gain matrix is set to all ones.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucResetTable", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucResetTable", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSaveInFlash

Beginner
This parameter is a Command. The only possible value is 1.
Save the content of the current selected Nuc table entry in flash memory(the Nuc table in flash is overridden).
As we are writing to flash memory, this operation may take some time.

Use this function only if you understand what you are doing.
'NucSaveInFlash' will overwrite factory calibrations.
This is a permanent modification and should be handled with care.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSaveInFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucSaveInFlash", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucLoadFromFlash

Beginner
This parameter is a Command. The only possible value is 1.
Load the content of the current selected Nuc table entry from flash to memory(the Nuc table in memory is overridden).
As we are reading from flash memory, this operation may take some time.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucLoadFromFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucLoadFromFlash", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprRun

Beginner
This parameter is a Command. The only possible value is 1.
Start the process of bad pixels detection using the detection method defined by 'BprDetectMethod'.
At the end of the operation, the current in memory bad pixel list is updated.
To make this new set of bad pixels persistent you have to call 'BprSave'.
This operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprRun", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprRun", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprActivate

Beginner
Activate or Deactivate the embedded Bpr processing.
The Bpr applied is the Bpr actually in memory.
This parameter can take these two values:

String value	Numeric value	Comment
Off	0	Embedded BPR is not active
On	1	Embedded BPR is active

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprActivate", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "BprActivate", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'BprActivate'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprDetectMethod

Beginner
Set the method used to detect bad pixels.
To effectively start the bad pixel detection process, you have to set a detection threshold and to call 'BprRun'.
This parameter can take these two values:

String value	Numeric value	Comment	Threshold to set
Gain	0	Detection using the gain response level	BprGainThreshold
Offset	1	Detection using the dark offset level	BprOffsetThreshold

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprDetectMethod", "Gain" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "BprDetectionMethod", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'BprDetectionMethod'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprBadPixelNumber

Beginner **ReadOnly**
Integer value in the range [0,UINT_MAX].
Return the number of current bad pixels.

BprGainThreshold

Beginner
Integer value in the range [0, 100].
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "BprGainThreshold", 6 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprGainThreshold", "6" );
```

Emit:

- NITConfigObserver.paramChanged 'BprGainThreshold'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprOffsetThreshold

Beginner
Integer value in the range [0, 100].
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "BprOffsetThreshold", 6 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprOffsetThreshold", "6"
);
```

Emit:

- NITConfigObserver.paramChanged 'BprOffsetThreshold'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprResetList

Beginner
This parameter is a Command. The only possible value is 1.
Reset the list of bad pixels in memory(bad pixel list in flash is untouched).
This operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set

these parameters.

```
Example: dev.setParamValueOf( "BprResetList", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprResetList", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprSave

Beginner
This parameter is a Command. The only possible value is 1.
Save the current in memory list of bad pixels to flash(bad pixel list in flash is overriden).
This operation may take some time.

Use this function only if you understand what you are doing.
'BprSave' will overwrite factory bad pixels map. This is a permanent modification and should be handled with care.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprSave", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprSave", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprLoad

Beginner
This parameter is a Command. The only possible value is 1.
Load the bad pixel list from flash to memory(the bad pixel list in memory is overridden).
As we are reading from flash memory, this operation may take some time.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprLoad", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprLoad", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcMode

Beginner
Set the method used to apply gain control.
The gain control is active if 'OutputType' is not 'RAW'.
This parameter can take these two values:

String value	Numeric value	Comment
Auto	0	The gain and offset are automatically calculated
Manuel	1	The gain and offset are preset

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcMode", "Auto" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcManualGain

Beginner
Set the gain used when gain control is set to manual.
Float value in the range [0.0, 1.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualGain", 0.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualGain", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualGain'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcManualOffset

Beginner
Set the offset used when gain control is set to manual.
Float value in the range [0.0, 16383.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualOffset", 10.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualOffset", "10.2"
);
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualOffset'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcAutoMeanStdThreshold

Beginner
Set the threshold used when gain control is set to auto.
Float value in the range [0.0, 16383.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcAutoMeanStdThreshold",
10.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcAutoMeanStdThreshold",
"10.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcAutoMeanStdThreshold'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcAutoGain

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated gain for the last frame when gain control is set to auto.

AgcAutoOffset

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated offset for the last frame when gain control is set to auto.

AgcStatisticsFmean

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the mean pixel value of the last frame.

AgcStatisticsFstd

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the standard deviation value of the last frame.

AgcStatisticsWmin

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the minimum pixel value of the last frame.

AgcStatisticsWmax

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the maximum pixel value of the last frame.

AgcRoiMode

Beginner
Set the rectangular region of the frame on which the gain control is calculated.
This parameter can take these values:

String value	Numeric value	Comment
FULL_FRAME	0	Region of interest is the entire frame
HALF_FRAME	1	Region of interest is centered in the frame
QUARTER_FRAME	2	Region of interest is centered in the frame
CUSTOM_FRAME	3	Region of interest is anywhere in the frame
		The rectangle is set

		by the parameters:
		- AgcRoiCustomLeft
		- AgcRoiCustomTop
		- AgcRoiCustomRight
		- AgcRoiCustomBottom

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiMode", "HALF_FRAME" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcRoiCustomTop **Beginner** Integer in the range [0, 510]

AgcRoiCustomBottom **Beginner** Integer in the range [1, 511]

AgcRoiCustomLeft **Beginner** Integer in the range [0, 638]

AgcRoiCustomRight **Beginner** Integer in the range [1, 639]
Set the rectangular region of the frame on which the gain control is calculated when the 'RoiMode' parameter is set to 'CUSTOM_FRAME'.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiCustomTop", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcRoiCustomBottom", "100" );
```

Emit:

- NITConfigObserver.paramChanged

'AgcRoiCustomTop', 'AgcRoiCustomBottom', 'AgcRoiCustomLeft' or 'AgcRoiCustomRight'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcRoiDraw

Beginner
Draw a rectangular frame in the image representing the rectangular region where the gain control is calculated.
This parameter can take this two values:

String value	Numeric value	Comment
Off	0	Rectangle not drawn
On	1	Rectangle drawn

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiDraw", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiDraw", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiDraw'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

ImgColorPolarity

Beginner
Permit to invert the pixel values of the frame.
This parameter can take these two values:

String value	Numeric value	Comment
Positif	0	pixel value 0 is black
Negatif	1	pixel value 0 is white

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "ImgColorPolarity",
"Negatif" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "ImgColorPolarity", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'ImgColorPolarity'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

RetOn

Beginner
Draw a reticule(crosshair) in the image.
This parameter can take these two values:

String value	Numeric value	Comment
Off	0	Reticule not drawn
On	1	Reticule drawn

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "RetOn", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "RetOn", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'RetOn'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

RetThickV

Beginner Integer in the range [0, 512]

RetThickH

Beginner Integer in the range [0, 640]
Set the thickness of the branches of the reticule.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "RetThickH", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "RetThickV", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'RetThickV' or 'RetThickH'.

Throw:

- NITException if the string value or the numeric value is out of range.

RetLengthV

Beginner Integer in the range [0, 512]

RetLengthH

Beginner Integer in the range [0, 640]
Set the length of the branches of the reticule.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "RetLengthH", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "RetLengthV", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'RetLengthV' or 'RetLengthH'.

Throw:

- NITException if the string value or the numeric value is out of range.

RetYcolor

Beginner
Integer in the range [0, 255]
Set the grey level of the reticule.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "RetYcolor", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "RetYcolor", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'RetYcolor'.

Throw:

- NITException if the string value or the numeric value is out of range.

RetCenterPosX

Beginner Integer in the range [0, 640]

RetCenterPosY

Beginner Integer in the range [0, 512]
Set the position of the center of the reticule.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "RetCenterPosX", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "RetCenterPosY", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'RetCenterPosX' or 'RetCenterPosY'.

Throw:

- NITException if the string value or the numeric value is out of range.

RetResetProperties

Beginner
This parameter is a Command. The only possible value is 1.
Set the reticule to its default values. That is color white centered in the image.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "RetResetProperties", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "RetResetProperties", "1"
);
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

SharpMode

Beginner
Apply a sharpen filter to the image.
This parameter can take these values:

String value	Numeric value	Comment

Off	0	Filtering off
SLOW	1	Light sharpning
MEDIUM	2	Medium sharpning
STRONG	3	Strong sharpning

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "SharpMode", "MEDIUM" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "SharpMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'SharpMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

EqualActivate

Beginner
Apply an equalization histogram filter to the image. This parameter can take these values:

String value	Numeric value	Comment
Off	0	Filtering off
On	1	Filtering on

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "EqualActivate", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "EqualActivate", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'EqualActivate'.

Throw:

- NITException if the string value or the

numeric value doesn't exist.

EqualThreshold

Beginner
Integer in the range [0, 100]
Threshold used to process equalization histogram.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "EqualThreshold", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "EqualThreshold", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'EqualThreshold'.

Throw:

- NITException if the string value or the numeric value is out of range.

Gamma

Beginner
Float in the range [0, 3]
Set gamma factor. A value of 1 imply no gamma correction
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "Gamma", 1.5 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Gamma", "1.4" );
```

Emit:

- NITConfigObserver.paramChanged 'Gamma'.

Throw:

- NITException if the string value or the numeric value is out of range.

GammaThd

Beginner
Integer in the range [0, 255]
Set the center of the gamma.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GammaThd", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GammaThd", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'GammaThd'.

Throw:

- NITException if the string value or the numeric value is out of range.

killpix_x

Expert Integer in the range [0, 639]

killpix_y

Expert Integer in the range [0, 511]
Permits to set the coordinates of a pixel to be added to the bad pixel list.
The bad pixel is added to the list when you call 'killpix_go'
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "killpix_x", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "killpix_x", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'killpix_x' or 'killpix_y.

Throw:

- NITException if the string value or the numeric value is out of range.

killpix_go

Expert
This parameter is a Command. The only possible value is 1.
Add the bad pixel defined by the coordinates given by 'killpix_x' and 'killpix_y' to the bad pixel list.
Only the in memory list is modified. The list in flash is untouched.
This operation may take some time.

```
Use this function only if you understand what you
```

are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "killpix_go", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "killpix_go", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

zone_killpix_start_x **Expert** Integer in the range [0, 639]

zone_killpix_end_x **Expert** Integer in the range [0, 639]

zone_killpix_start_y **Expert** Integer in the range [0, 511]

zone_killpix_end_y **Expert** Integer in the range [0, 511]
Permits to set the coordinates of an enclosing rectangle of a cluster of pixels to be added to the bad pixel list.
The bad pixel is added to the list when you call 'zone_killpix_go'
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "zone_killpix_start_x", 10
);
         dev.setParamValueOf( "zone_killpix_end_y", 100
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "zone_killpix_end_x", "10"
);
```

Emit:

- NITConfigObserver.paramChanged 'zone_killpix_start_x', 'zone_killpix_start_y', 'zone_killpix_end_x' or 'zone_killpix_end_y'.

Throw:

- NITException if the string value or the numeric value is out of range.

zone_killpix_go **Expert**
This parameter is a Command. The only possible value is 1.

Add the bad pixels included in the rectangle defined by the coordinates given by 'zone_killpix_start_x', 'zone_killpix_start_y', 'zone_killpix_end_x' and 'zone_killpix_end_y' to the bad pixel list.
Only the in memory list is modified. The list in flash is untouched.
This operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "zone_killpix_go", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "zone_killpix_go", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

The following parameters are for debug purpose only and must not be modified by the users.

Parameters

CamSelectDataSrc	**Guru**
NucSetCustom	**Guru**
Cbias	**Guru**
Dbias	**Guru**

The following parameters are the boostrap GigeVision parameters. They follow the GigE Vision® Specification version 2.0.
We invite you to consult the specification for more details.

Parameters

PayloadSize	**Expert** **ReadOnly** Integer in the range [0, UINT_MAX] bytes Maximum number of bytes transferred for each image on the stream channel, including any end-ofline, end-of-frame statistics or other stamp data. This is the maximum total size of data payload for a block.

UDP and GVSP headers are not considered.
Data leader and data trailer are not included.
This is mainly used by the application software to determine size of image buffers to allocate (largest buffer possible for current mode of operation).
For example, an image with no statistics or stamp data as PayloadSize equals to (width x height x pixelsize) in bytes.

GevVersion

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Version of the GigE Standard with which the device is compliant
The two most-significant bytes are allocated to the major number of the version,
the two least-significant bytes for the minor number.

GevDeviceMode

Guru **ReadOnly**
Integer in the range [0, UINT_MAX]
Information about device mode of operation.
Returns Big Endian and UTF8.

GevDeviceMACAddressHigh

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
The two most-significant bytes of this area are reserved and will return 0.
Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
The MAC address of the camera can be read at the rear of the device.

GevDeviceMACAddressLow

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Lower 4 bytes of the MAC address.
The MAC address of the camera can be read at the rear of the device.

GevNetworkCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Supported IP Configuration and PAUSE schemes.
Returns LLA|DHCP|PersistentIp.

GevNetworkConfiguration

Expert
Activated IP Configuration and PAUSE schemes.
This parameter can take these values:

String value	Numeric value	Comment
LLA	4	Local

		Link(APIPA) only
LLA_Persistent_IP	5	Local Link and Persistent IP
LLA_DHCP	6	Local Link and DHCP(default)
All	7	All modes

If set to LLA: on connection, the camera negociate an LLA Ip Address.
If set to LLA_DHCP: on connection, the camera tries to contact a DHCP server to acquire an Ip address. If no DHCP is present, the device fallback to LLA negociation.
If set to LLA_Persistent_IP: on connection, the camera tries to negociate an LLA Ip address. If the LLA negociation fail, the device fallback to Persistent IP.
If set to All: The device tries to connect with DHCP, with LLA and finally takes the persistent IP.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", "LLA" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", 5 );
```

Emit:

- NITConfigObserver.paramChanged 'GevNetworkConfiguration'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

GevCurrentIPAddress

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current IP address of this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentSubnetMask

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current subnet mask used by this device on its

first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentDefaultGateway

Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current default gateway used by this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevManufacturerName

Beginner **ReadOnly**
Provides the name of the manufacturer of the device.
Returns the string: "New-Imaging-Technologies"

GevModelName

Beginner **ReadOnly**
Provides the model name of the device.
Returns the string "NSC1201GigE".

GevDeviceVersion

Beginner **ReadOnly**
Provides the version of the device.
Returns a string whith the format "MM.mm"
Example: "02.07"

GevManufacturerInfo

Beginner **ReadOnly**
Provides extended manufacturer information about the device.
Returns the string "Camera NSC1201 from NIT"

GevSerialNumber

Beginner **ReadOnly**
Contains the serial number of the device.
It can be used to identify the device.
Returns a string with the format "NNNNNN"
Example: "200162"

GevUserDefinedName

Beginner
Null terminated string with maximum 16 characters(terminating null included).
This string contains a user-programmable name to assign to the device.
It can be used to identify the device.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevUserDefinedName",
"MyCamera" );
```

Emit:

- NITConfigObserver.paramChanged

'GevUserDefinedName'.

Throw:

- NITException if the string is to long.

GevFirstURL

Guru ReadOnly
This register stores the first URL to the XML device description file.
The first URL is used as the first choice by the application to retrieve the XML device description file.
Returns a string with the format <filename>;<address>;<file size>
Example: "local:adr_nios_1201.zip;A200;1938"

GevSecondURL

Guru ReadOnly
This register stores the second URL to the XML device description file.
This URL is an alternative if the application was unsuccessful to retrieve the device description file using the first URL.

GevNumberOfInterfaces

Expert ReadOnly
Integer in the range [0, UINT_MAX]
This register indicates the number of network interfaces supported by this device.
This is generally equivalent to the number of Ethernet connectors on the device, except when Link Aggregation is used. In this case, the physical interfaces regrouped by the Aggregator are counted as one virtual interface.
Returns allways 1.

GevLinkSpeed

Expert ReadOnly
Integer in the range [0, UINT_MAX]
Value indicating current Ethernet link speed in Mbits per second. Gigabit ethernet returns 1000.

GevMessageChannelCount

Expert ReadOnly
Integer in the range [0, UINT_MAX]
Indicates the number of message channels supported by this device.
Returns allways 0.

GevStreamChannelCount

Expert ReadOnly
Integer in the range [0, UINT_MAX]
Indicates the number of stream channels supported by this device.
Returns allways 1.

GevActiveLinkCount

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of physical links that are currently active. A link is considered active as soon as it is connected to another Ethernet device. This happens after Ethernet link negotiation is completed.
Returns allways 1.

GevStreamChannelCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
First Stream Channel Capability register.
Returns SCSPx_supported|legacy_16bit_block_id_supported.

GevMessageChannelCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the capabilities of the message channel. It lists which of the non-mandatory message channel features are supported.
Returns allways 0.

GevGVCPCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
This is a capability register indicating which one of the non-mandatory GVCP features are supported by this device. When supported, some of these features are enabled through the GVCP Configuration register.
Returns user_defined_name|serial_number|link_speed_register|WRITEMEM.

GevHeartbeatTimeout

Guru
Integer in the range [0, UINT_MAX] milliseconds.
Indicates the current heartbeat timeout in milliseconds.
Default is 3000 msec.
Internally, the heartbeat is rounded according to the clock used for heartbeat.
The heartbeat timeout shall have a minimum precision of 100 ms.
The minimal value is 500 ms.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout",
2000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout",
```

```
"1500" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevGVCPPendingTimeout

Expert **ReadOnly**
Integer in the range [0, UINT_MAX] milliseconds.
Pending Timeout to report the longest GVCP command execution time before issuing a PENDING_ACK. If PENDING_ACK is not supported, then this is the worstcase execution time before command completion.
Returns 300000.

GevPhysicalLinkConfigurationCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the physical link configuration supported by this device. Returns 0.

GevPhysicalLinkConfiguration

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the currently active physical link configuration.
Returns 0.

GevCCP

Guru
Integer in the range [0, UINT_MAX]
This register is used to grant privilege to an application.
Only one application is allowed to control the device. This application is able to write into device's registers. Other applications can read device's register only if the controlling application does not have the exclusive privilege.

On construction of the NITDevice the SDK set this parameter to exclusive access.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevCCP", 2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevCCP", "2" );
```

Throw:

- NITException if the string value or

the numeric value is out of range.

GevSCPHostPort

Guru
Integer in the range [0, UINT_MAX]
Stream Channel Host Port register.
The host port to which a stream channel must send data stream.
Setting this value to 0 closes the stream channel.
The frame number is reset to 1 when the stream channel is opened.

This parameter is managed by the SDK.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPHostPort", 12000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPHostPort", "14200" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCSP

Guru ReadOnly
Integer in the range [0, UINT_MAX]
Stream Channel Source Port register.
The port on which the stream channel emits packets.

GevSCDA

Guru
Integer in the range [0, UINT_MAX]
Stream Channel destination Ip address of the receiving host.
The Ip address to which a stream channel must send data stream.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

On construction of the NITDevice the SDK set this parameter to the IP address of the host.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCDA", 3232236047
```

```
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCDA", "3232236047"
);
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentIPAddress

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent IP address for the given network interface.
Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentSubnetMask

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme. Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentDefaultGateway

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme. Each byte represents a part of the Ip address. Example: 192.168.2.15 is coded 0xC0A8020F. Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCPSPacketSize

Expert
Integer in the range [29, 8100] bytes
Size of the stream packets to transmit during acquisition.

On construction of the NITDevice the SDK negociate the highest possible packet size with the device.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPSPacketSize",
8000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPSPacketSize",
"8000" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

TimeStamp_frequency_high **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
High part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the most significant bytes.

TimeStamp_frequency_low **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Low part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the least significant bytes.

TimestampValueHigh **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (most significant bytes). The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current frame.

TimestampValueLow **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (least significant bytes) The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current frame.

TimestampControl **Expert** **WriteOnly**
Integer in the range [0, 3]
Write 1 to reset timestamp 64-bit counter to 0.
Write 2 to latch current timestamp counter into TimestampValue registers.
Write 3 to first latch current timestamp counter into TimeStampValue registers and then reset.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TimestampControl", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "TimestampControl", "3"
);
```

Throw:

- NITException if the string value or the numeric value is out of range.

20. WiDySenS 640G-STE series parameters

NITLibrary 3.x series support NSC1601GigE cameras with firmware version above or equal to 1.5.0.

The following parameters are supported:
Where not otherwise indicated parameters are Read/Write.

Parameters

WidthMax

Expert **ReadOnly**
Integer value in the range [8, 640] pixels.
Updated each time OffsetX, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Width' parameter.

HeightMax

Expert **ReadOnly**
Integer value in the range [8, 512] pixels.
Updated each time OffsetY, ExposureTime or Framerate is changed.
This is the maximum value which can be set actually for the 'Height' parameter.

Width

Beginner
Integer value in the range [16, WidthMax] by steps of 8 pixels.
The value you can set is constrained by the frame rate and 'WidthMax'.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Width", 600 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Width", "600" );
```

Emit:

- NITConfigObserver.paramChanged 'Width'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

Height

Beginner
Integer value in the range [16, HeightMax] by steps of

8 pixels.
The value you can set is constrained by the frame rate and 'HeightMax'.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "Height", 500 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Height", "500" );
```

Emit:

- NITConfigObserver.paramChanged 'Height'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

OffsetX

Beginner
Integer value in the range [0, 640] by steps of 4 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetX", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetX", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetX'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Width' + 'OffsetX' is above 640.

OffsetY

Beginner
Integer value in the range [0, 512] by steps of 4 pixels.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "OffsetY", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "OffsetY", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'OffsetY'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the sum 'Height' + 'OffsetY' is above 512.

PixelFormat

Beginner
While this parameter is writeable, its use is limited.
The pixel format is in fact changed by the 'OutputType' parameter.
The following pixel formats are available:

String value	Numeric value	Comment	NITFrame pixel type
Mono8	17301505	Set if 'OutputType' is set to 'GREY'	FLOAT
Mono14	17825829	Set if 'OutputType' is set to 'RAW'	FLOAT

Emit:

- NITConfigObserver.paramChanged 'PixelFormat'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

OutputType

Beginner
Permits to set the desired output type.
In all cases, if NUC and/or BPR are active they are applied before the OutputType.
The following output types are available:

String value	Numeric value	Comment
GREY	0	Activate gain control. Change

		'PixelFormat' to Mono8
RAW	7	Deactivate gain control. Change 'PixelFormat' to Mono14

RAW is the output of the sensor with application of the NUC and BPR if they are active.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "OutputType", "GREY" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "OutputType", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'OutputType'.
- NITConfigObserver.paramChanged 'PixelFormat'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

Flip_Image

Beginner
Change the orientation of the frame.
The following orientations are available:

String value	Numeric value	Comment
NoFlip	0	No flip
FlipH	1	Flip frame horizontally
FlipV	2	Flip frame vertically
FlipHV	3	Flip frame horizontally and vertically

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "Flip_Image", "FlipH" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "Flip_Image", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'Flip_Image'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionMode

Beginner
This parameter is writable but can take only one value.

String value	Numeric value	Comment
Continuous	0	Frames are continuously outputted until AcquisitionStop is called
SingleFrame	1	Only one frame is acquired when AcquisitionStart is called
MultiFrame	2	N Frames are acquired when AcquisitionStart is called N is set by 'AcquisitionFrameCount'

When you set SingleFrame or MultiFrame, call NITDevice.captureNFrames(N) to start acquisition, and call NITDevice.WaitEndOfFrame() to wait for the end of capture.
Calling captureNFrames will automatically set the number of frames to capture and call 'AcquisitionStart'.

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionMode",
"SingleFrame" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AcquisitionMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AcquisitionMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionFrameCount

Beginner
Integer value in the range [1, 65535].
Number of frames to acquire when in 'Multiframe' 'AcquisitionMode'.
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "AcquisitionFrameCount",
100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionFrameCount",
"100" );
```

Emit:

- NITConfigObserver.paramChanged 'AcquisitionFrameCount'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionStart

Beginner
This parameter is a Command. The only possible value is 1.
Start to output frames.

This command is a direct command to the camera. It bypasses the variables initialisations performed by the SDK.
Prefer NITDevice.start(), NITDevice.captureNFrames() or NITDevice.captureForDuration() to start the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStart", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStart", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the camera is already started.

AcquisitionStop

Beginner
This parameter is a Command. The only possible value is 1.
Stop to output frames.

This command is a direct command to the camera. It bypass the variables initialisations performed by the SDK.
Prefer NITDevice.stop() to stop the capture under control of the SDK.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionStop", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionStop", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AcquisitionFrameRate

Beginner
Set the frame rate.
Float value in the range [1.0,MaxFps] Hz
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AcquisitionFrameRate",
100.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AcquisitionFrameRate",
"100.2" );
```

Emit:

- NITConfigObserver.paramChanged

'AcquisitionFrameRate'.
- NITConfigObserver.paramChanged 'MaxExposure'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current ExposureTime or resolution.

MaxFPS

Beginner **ReadOnly**
Float value in the range [1,10000] Hz.
Updated each time the resolution or the ExposureTime are changed.
This is the maximum value which can be actually set for the 'AcquisitionFrameRate' parameter.

CameraMode

Beginner
Set sensor response of the camera.

String value	Numeric value	Comment
Log	0	Logatithmic response
LinLow	1	Linear response with low gain
LinHigh	2	Linear response with high gain
LinHighCds	3	Linear response with high gain and Cds

These values are a combination of the following features:

- Log : Logarithmic response. The output of the sensor is not proportional to the amount of light reaching the pixels. This allow wide dynamic range images(120db).
- Lin : Linear response. The output of the sensor is proportional to the amount of light reaching the pixels and the integration time.
- Low gain: Higher noise but allows higher dynamic range.
- High gain: High sensitivity.
- Cds: Correlated Double Sampling is a in-sensor noise reduction method in which the signal is sampled twice: once with the sensor in a reset state and once at the end of the sensor exposure.

Use the string form of setParamValueOf to set these

parameters.

```
Example: dev.setParamValueOf( "CameraMode", "LinHigh" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "CameraMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'CameraMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

IntegrationMode

Beginner

Set integration mode of the camera.

String value	Numeric value	Comment
ITR	0	Integrate Then ReadOut
IWR	1	Integrate While ReadOut

The meaning of these values are detailed below:

- ITR : Integrate Then Read. The image is being read from the sensor once the exposure is finished and the next exposure can only start once the readout of the current image has been finished.
- IWR : Integrate While Read. The image is being read from the sensor once the exposure is finished but the next exposure can start before the readout of the current image has been finished.
  This mode allows reaching higher frame rates. IWR cannot work with exposure times below 100µs. To use shorter exposures, ITR mode should be selected.

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "IntegrationMode", "IWR" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "IntegrationMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'IntegrationMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

TECMode

Beginner

Permit to activate the regulation of the camera temperature.
The parameter "TECTemperature" sets the target temperature.
The available modes are:

String value	Numeric value	Comment
Off	0	TEC is not active
LowCurrent	1	
MediumCurrent	2	
LargeCurrent	3	

Higher currents allow reaching the target temperature faster.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TECMode", "LowCurrent" );
```

You can also set these values as unsigned int value(in this case the value is the index of the enumeration):

```
Example: dev.setParamValueOf( "TECMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'TECMode'.

Throw:

- NITException if the string value or numeric value doesn't exist.

TECTemperature

Beginner

Float value in the range [-15.0,48.0] °C
Use the numeric(double) form of setParamValueOf to set this parameter.

```
Example: dev.setParamValueOf( "TECTemperature", -5.0 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "TECTemperature", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'TECTemperature'.

Throw:

- NITException if the string value or numeric value doesn't exist.

TriggerDirection

Beginner
Set the direction of the trigger signal.
The following directions are available:

String value	Numeric value	Comment
Internal	0	The trigger signal is generated and outputted by the camera each time a frame is outputted
External	1	The trigger signal is external and start acquisition of one frame

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerDirection",
"External" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerDirection", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerDirection'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

TriggerExpo

Beginner
Set how the exposure duration is managed when 'TriggerDirection' is set to 'External'.
The following behaviours are available:

String value	Numeric value	Comment
TTLWidth	0	Exposure duration is the trigger width(signal high)
ExposureTime	1	Exposure duration is the value of 'ExposureTime' parameter

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerExpo", "TTLWidth" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerExpo", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerExpo'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

TriggerDelay

Beginner
Integer value in the range [-1280000, 1780000] nanoseconds.
Set the delay between the rising edge of the trigger and the beginning of exposure.

'TriggerDelay' can be negative. To make this possible,

the camera uses the first trigger to synchronize. This implies the first frame is never transmitted out of the camera. Furthermore the camera supposes the trigger period is constant. Dont use 'TriggerDelay' below +130 microseconds if the trigger period is not constant or you use the 'CaptureNFrames' function(you will never have the number of frames expected).

Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.etParamValueOf( "TriggerDelay", 500 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "TriggerDelay", "500" );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerDelay'.

Throw:

- NITException if the string value or the numeric value doesn't exist.
- NITException if the value is out of range.

TriggerOut1 **Beginner**

TriggerOut2 **Beginner**

Set the behaviour of the GPO signals Out1 and Out2. The following behaviours are available:

String value	Numeric value	Comment
Exposing	0	Output is high when camera exposes
TriggerReady	1	Output is high when camera is ready to receive an external trigger signal
TriggerInput	2	Output mirrors the external trigger signal
ReadOut	3	Output is high when camera read pixels
Imaging	4	Output is high during exposition and readout
GPOLowLevel	5	Output is forced low

GPOHighLevel	6	Output is forced high

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TriggerOut1",
"TriggerInput" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "TriggerOut1", 4 );
```

Emit:

- NITConfigObserver.paramChanged 'TriggerOut1' or 'TriggerOut2'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

ExposureTime

Beginner
Set the duration of the exposition.
Float value in the range [1.0,MaxExposureTime] microseconds
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "ExposureTime", 100.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "ExposureTime", "100.2" );
         dev.setParamValueOf( "ExposureTime", "100.2us"
);
```

Emit:

- NITConfigObserver.paramChanged 'ExposureTime'.
- NITConfigObserver.paramChanged 'MaxFps'.

Throw:

- NITException if the string value or the numeric value is out of range.
- NITException if the value cannot be set due to the current 'AcquisitionFrameRate'.

MaxExposureTime

Beginner **ReadOnly**
Float value in the range [1,65535] microseconds.

Updated each time the resolution or the 'AcquisitionFrameRate' are changed.
This is the maximum value which can be actually set for the 'ExposureTime' parameter.

AutomaticExposureTime **Beginner**

Activate/Deactivate AutomaticExposure mode.
When enabled, the exposure time is adapted so that the images have a mean luminosity.
The adaptation is done by calculating the exposure time to set for next image taking into account the mean value and exposure time of the current image.
The following values are available:

String value	Numeric value	Comment
Off	0	Automatic exposure is disabled
On	1	Automatic exposure is enabled

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AutomaticExposureTime",
"On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AutomaticExposureTime", 1
);
```

Emit:

- NITConfigObserver.paramChanged 'AutomaticExposureTime'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

SetUpAET **Beginner**

Integer value in the range [0,100]
Converging speed factor of the 'AutomaticExposureTime'.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "SetUpAET", 50 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "SetUpAET", "100" );
```

Emit:

- NITConfigObserver.paramChanged 'SetUpAET'.

Throw:

- NITException if the string value or the numeric value is out of range.

CamBoardTemperature

Beginner **ReadOnly**
Float value in the range [-200.0,200.0] °C.
Temperature of the sensor.

CamLoadConfiguration

Beginner
This parameter is a Command. The only possible value is 1.
Load the parameters stored in flash.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamLoadConfiguration", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamLoadConfiguration", "1"
);
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

CamSaveConfiguration

Beginner
This parameter is a Command. The only possible value is 1.
Save the current parameters in flash.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamSaveConfiguration", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamSaveConfiguration", "1"
);
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

CamResetParameter

Beginner
This parameter is a Command. The only possible value is 1.
Reset the current parameters to their default value.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CamResetParameter", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CamResetParameter", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NIOS_IS_BUSY

Beginner **ReadOnly**
Running some commands like eg. 'CamSaveConfiguration' access the flash and can take some time.
During this time new commands or parameter setting are not taken into account.
This parameter permits to know if it is safe to send commands and parameters to the camera.
If returned value is 1: The camera is busy, don't send commands or parameters.
If returned value is 0: The camera is not busy, you can safely send commands or parameters.

This information is for documentation only.
NITLibrary look after the busyness of the camera before sending parameters or commands.

NucActivate

Beginner
Activate or Deactivate the embedded Nuc processing.
If embedded Nuc is activated, library Nuc is deactived.
The Nuc applied is the Nuc in the current selected table. see nucBpr
This parameter can take these two values:

String value	Numeric value	Comment
Off	0	Embedded NUC is not active

On	1	Embedded NUC is active

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucActivate", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucActivate", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'NucActivate'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSelectTable

Beginner

Select the current Nuc table who will be applied to the captured frames.
At delivery time of the camera, a set of 4 tables per 'CameraMode' are provided for different exposure times.
This parameter can take these values:

String value	Numeric value	Exposure time at delivery
NONE	0	
NUC1	1	10 µs
NUC2	2	100 µs
NUC3	3	1000 µs
NUC4	4	10000 µs

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSelectTable", "NUC2" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucSelectTable", 2 );
```

Emit:

- NITConfigObserver.paramChanged 'NucSelectTable'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucResetTable

Beginner

This parameter is a Command. The only possible value is 1.
Reset the content of the current selected Nuc table entry in memory(the Nuc table in flash is untouched). The low point is set to all 0 and the gain matrix is set to all ones.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucResetTable", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucResetTable", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucMode

Beginner

Set the mode of operation of NUC.
The Nuc applied is the Nuc in the current selected table. see nucBpr
This parameter can take these two values:

String value	Numeric value	Comment
Automatic	0	See below
Manual	1	applied NUC is the one in the current selected table

In 'Automatic' 'NucMode', the NUC applied is the one corrsponding to the current 'ExposureTime', for the current 'CameraMode'.

If no such NUC exists in the tables, an interpolation is done with the two nearest NUC tables.
Example:

- If 'CameraMode' is 'Log' and current 'ExposureTime' is 1000µs, the table NUC3 for 'Log' 'CameraMode' is applied. (see above 'NucSelectTable').
- If 'CameraMode' is 'Log' and current 'ExposureTime' is 800µs, the table NUC2 and NUC3 for 'Log' 'CameraMode' are interpolated.

String value	Numeric value	Exposure time at delivery
NONE	0	
NUC1	1	10 µs
NUC2	2	100 µs
NUC3	3	1000 µs
NUC4	4	10000 µs

When setting a new exposure a new NUC table will be loaded if the new exposure time is out of the range allowed by the NUC tables currently in memory.
Examples:

- Switching ‘CameraMode’ parameter will cause the camera to load new NUC tables from flash to memory.
- If current exposure time is 50µs, NUC1 & NUC2 are in memory. When switching to 5000µs NUC3 & NUC4 will need to be loaded from flash to memory.

NUC loading will cause a few images to appear partially uncorrected, switching to Manual will allow quick exposure time variations without the risk of getting partially corrected images.
Image quality might be slightly degraded when using a NUC table calibrated for an exposure setting which differs a lot from the current setpoint.

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucMode", "Manual" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "NucMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'NucMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucCompute1Pts

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point for this device with the current configuration, the gain is untouched. The resulting Nuc is set in the current selected table. This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute1Pts", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute1Pts", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucCompute2Pts1

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point for this device with the current configuration, the gain matrix is set to all ones.
The resulting Nuc is set in the current selected table. This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute2Pts1", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute2Pts1", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucCompute2Pts2

Beginner
This parameter is a Command. The only possible value is 1.
Run the computation of the low point and gain matrix for this device with the current configuration.
The resulting Nuc is set in the current selected table.
This operation may take some time. The process need to capture 50 frames to compute the Nuc.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucCompute2Pts2", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucCompute2Pts2", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucSaveInFlash

Beginner
This parameter is a Command. The only possible value is 1.
Save the content of the current selected Nuc table entry in flash memory(the Nuc table in flash is overridden).
As we are writing to flash memory, this operation may take some time.

Use this function only if you understand what you are doing.
'NucSaveInFlash' will overwrite factory calibrations.
This is a permanent modification and should be handled with care.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucSaveInFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucSaveInFlash", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

NucLoadFromFlash

Beginner
This parameter is a Command. The only possible value is 1.
Load the content of the current selected Nuc table entry from flash to memory(the Nuc table in memory is overridden).
As we are reading from flash memory, this operation may take some time.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "NucLoadFromFlash", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NucLoadFromFlash", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprRun

Beginner
This parameter is a Command. The only possible value is 1.
Start the process of bad pixels detection using the detection method defined by 'BprDetectMethod'.
At the end of the operation, the current in memory bad pixel list is updated.
To make this new set of bad pixels persistent you have to call 'BprSave'.
This operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprRun", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprRun", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprActivate

Beginner
Activate or Deactivate the embedded Bpr processing.
The Bpr applied is the Bpr actually in memory.
This parameter can take these two values:

String value	Numeric value	Comment
Off	0	Embedded BPR is not active
On	1	Embedded BPR is active

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprActivate", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "BprActivate", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'BprActivate'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprDetectMethod

Beginner
Set the method used to detect bad pixels.
To effectively start the bad pixel detection process, you have to set a detection threshold and to call 'BprRun'.
This parameter can take these two values:

String value	Numeric value	Comment	Threshold to set
Gain	0	Detection using the gain response level	BprGainThreshold
Offset	1	Detection using the dark offset level	BprOffsetThreshold

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprDetectMethod", "Gain"
);
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "BprDetectionMethod", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'BprDetectionMethod'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprBadPixelNumber

Beginner ReadOnly
Integer value in the range [0,UINT_MAX].
Return the number of current bad pixels.

BprGainThreshold

Beginner
Integer value in the range [0, 100].
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "BprGainThreshold", 6 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprGainThreshold", "6" );
```

Emit:

- NITConfigObserver.paramChanged 'BprGainThreshold'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprOffsetThreshold

Beginner
Integer value in the range [0, 100].
Use the unsigned int form of setParamValueOf to set these parameters in numeric format.

```
Example: dev.setParamValueOf( "BprOffsetThreshold", 6 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprOffsetThreshold", "6"
```

```
);
```

Emit:

- NITConfigObserver.paramChanged 'BprOffsetThreshold'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprResetList

Beginner
This parameter is a Command. The only possible value is 1.
Reset the list of bad pixels in memory(bad pixel list in flash is untouched).
This operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprResetList", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprResetList", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprSave

Beginner
This parameter is a Command. The only possible value is 1.
Save the current in memory list of bad pixels to flash(bad pixel list in flash is overriden).
This operation may take some time.

Use this function only if you understand what you are doing.
'BprSave' will overwrite factory bad pixels map. This is a permanent modification and should be handled with care.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprSave", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprSave", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

BprLoad

Beginner
This parameter is a Command. The only possible value is 1.
Load the bad pixel list from flash to memory(the bad pixel list in memory is overridden).
As we are reading from flash memory, this operation may take some time.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "BprLoad", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "BprLoad", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

killpix_x

Expert Integer in the range [0, 639]

killpix_y

Expert Integer in the range [0, 511]
Permits to set the coordinates of a pixel to be added to the bad pixel list.
The bad pixel is added to the list when you call 'killpix_go'
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "killpix_x", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "killpix_x", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'killpix_x' or 'killpix_y.

Throw:

- NITException if the string value or the numeric value is out of range.

killpix_go

Expert
This parameter is a Command. The only possible value is 1.
Add the bad pixel defined by the coordinates given by 'killpix_x' and 'killpix_y' to the bad pixel list.
Only the in memory list is modified. The list in flash is untouched.
This operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "killpix_go", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "killpix_go", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

zone_killpix_start_x **Expert** Integer in the range [0, 639]

zone_killpix_end_x **Expert** Integer in the range [0, 639]

zone_killpix_start_y **Expert** Integer in the range [0, 511]

zone_killpix_end_y **Expert** Integer in the range [0, 511]
Permits to set the coordinates of an enclosing rectangle of a cluster of pixels to be added to the bad pixel list.
The bad pixel is added to the list when you call 'zone_killpix_go'
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "zone_killpix_start_x", 10 );
         dev.setParamValueOf( "zone_killpix_end_y", 100 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "zone_killpix_end_x", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'zone_killpix_start_x', 'zone_killpix_start_y', 'zone_killpix_end_x' or 'zone_killpix_end_y'.

Throw:

- NITException if the string value or the numeric value is out of range.

zone_killpix_go

Expert
This parameter is a Command. The only possible value is 1.
Add the bad pixels included in the rectangle defined by the coordinates given by 'zone_killpix_start_x', 'zone_killpix_start_y', 'zone_killpix_end_x' and 'zone_killpix_end_y' to the bad pixel list.
Only the in memory list is modified. The list in flash is untouched.
This operation may take some time.

Use this function only if you understand what you are doing.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "zone_killpix_go", 1 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "zone_killpix_go", "1" );
```

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcMode

Beginner
Set the method used to apply gain control.
The gain control is active if 'OutputType' is not 'RAW'.
This parameter can take these two values:

String value	Numeric value	Comment
Auto	0	The gain and offset are automatically calculated
Manuel	1	The gain and offset are preset

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcMode", "Auto" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcManualGain

Beginner
Set the gain used when gain control is set to manual.
Float value in the range [0.0, 1.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualGain", 0.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualGain", "0.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualGain'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcManualOffset

Beginner
Set the offset used when gain control is set to manual.
Float value in the range [0.0, 16383.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcManualOffset", 10.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcManualOffset", "10.2"
);
```

Emit:

- NITConfigObserver.paramChanged 'AgcManualOffset'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcAutoGain

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated gain for the last frame when gain control is set to auto.

AgcAutoOffset

Beginner **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the calculated offset for the last frame when gain control is set to auto.

AgcStatisticsFmean

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the mean pixel value of the last frame.

AgcStatisticsFstd

Expert **ReadOnly**
Float value in the range [0, 4294967295.0].
Return the standard deviation value of the last frame.

AgcStatisticsWmin

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the minimum pixel value of the last frame.

AgcStatisticsWmax

Expert **ReadOnly**
Integer value in the range [0, UINT_MAX].
Return the maximum pixel value of the last frame.

AgcRoiMode

Beginner
Set the rectangular region of the frame on which the gain control is calculated.
This parameter can take these values:

String value	Numeric value	Comment
FULL_FRAME	0	Region of interest is the entire frame
HALF_FRAME	1	Region of interest is centered in the frame
QUARTER_FRAME	2	Region of interest is centered in the frame

CUSTOM_FRAME	3	Region of interest is anywhere in the frame
		The rectangle is set by the parameters:
		- AgcRoiCustomLeft
		- AgcRoiCustomTop
		- AgcRoiCustomRight
		- AgcRoiCustomBottom

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiMode", "HALF_FRAME" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiMode", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiMode'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

AgcRoiCustomTop **Beginner** Integer in the range [0, 510]

AgcRoiCustomBottom **Beginner** Integer in the range [1, 511]

AgcRoiCustomLeft **Beginner** Integer in the range [0, 638]

AgcRoiCustomRight **Beginner** Integer in the range [1, 639]
Set the rectangular region of the frame on which the gain control is calculated when the 'RoiMode' parameter is set to 'CUSTOM_FRAME'.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiCustomTop", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcRoiCustomBottom", "100"
```

```
);
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiCustomTop', 'AgcRoiCustomBottom', 'AgcRoiCustomLeft' or 'AgcRoiCustomRight'.

Throw:

- NITException if the string value or the numeric value is out of range.

AgcRoiDraw

Beginner
Draw a rectangular frame in the image representing the rectangular region where the gain control is calculated.
This parameter can take this two values:

String value	Numeric value	Comment
Off	0	Rectangle not drawn
On	1	Rectangle drawn

Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcRoiDraw", "On" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf( "AgcRoiDraw", 1 );
```

Emit:

- NITConfigObserver.paramChanged 'AgcRoiDraw'.

Throw:

- NITException if the string value or the numeric value doesn't exist.

NbIgnorePixelsLowHisto

Beginner

NbIgnorePixelsHighHisto

Beginner
Float value in the range [0.0, 10.0] pixels.
Count of first/last pixels not taken into account when determining the min/max value of the gain control when in 'Auto' 'AgcMode'.
Use the numeric form of setParamValueOf to set

these parameters.

```
Example: dev.setParamValueOf( "NbIgnorePixelsLowHisto", 5.0 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "NbIgnorePixelsLowHisto", "5" );
```

Emit:

- NITConfigObserver.paramChanged 'NbIgnorePixelsLowHisto' or NbIgnorePixelsHighHisto.

Throw:

- NITException if the value is out of range.

AgcSmoothingCoefficient

Beginner

Set the smoothing coefficient used to calculate gain control.
Smoothing is done with a temporal averaging on gain and offset.
This parameter is only effective in 'Auto' 'AgcMode'.
Float value in the range [0.0, 100.0]
Use the double form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "AgcSmoothingCoefficient", 10.2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "AgcSmoothingCoefficient", "10.2" );
```

Emit:

- NITConfigObserver.paramChanged 'AgcSmoothingCoefficient'.

Throw:

- NITException if the string value or the numeric value is out of range.

Gamma

Beginner

Float in the range [0, 3]
Set gamma factor. A value of 1 imply no gamma correction
Use the double form of setParamValueOf to set these

parameters.

```
Example: dev.setParamValueOf( "Gamma", 1.5 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "Gamma", "1.4" );
```

Emit:

- NITConfigObserver.paramChanged 'Gamma'.

Throw:

- NITException if the string value or the numeric value is out of range.

GammaThd

Beginner
Integer in the range [0, 255]
Set the center of the gamma.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GammaThd", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GammaThd", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'GammaThd'.

Throw:

- NITException if the string value or the numeric value is out of range.

CustomReg_1 **Expert**

CustomReg_2 **Expert**

CustomReg_3 **Expert**

CustomReg_4 **Expert**

CustomReg_5 **Expert**

CustomReg_6 **Expert**

CustomReg_7 **Expert**

CustomReg_8 **Expert**

CustomReg_9 **Expert**

CustomReg_10 **Expert**

Integer in the range [0, UINT_MAX]
Registers provided to the user to store cutom values.
Writing to this registers may take some time.
Use the integer form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "CustomRegister_3", 10 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "CustomRegister_3", "10" );
```

Emit:

- NITConfigObserver.paramChanged 'CustomRegister_N'.

Throw:

- NITException if the string value or the numeric value is out of range.

The following parameters are for debug purpose only and must not be modified by the users.

Parameters

pattern **Guru**

ADCTest **Guru**

tableddr1number **Guru**

tableddr1coefficient **Guru**

tableddr1ExposureTime **Guru**

tableddr2number **Guru**

tableddr2coefficient **Guru**

tableddr2ExposureTime **Guru**

The following parameters are the boostrap GigeVision parameters. They follow the GigE Vision® Specification version 2.0.
We invite you to consult the specification for more details.

Parameters

PayloadSize **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX] bytes
Maximum number of bytes transferred for each image on the stream channel, including any end-ofline, end-of-frame statistics or other stamp data. This is the maximum total size of data payload for a block.
UDP and GVSP headers are not considered.
Data leader and data trailer are not included.
This is mainly used by the application software to determine size of image buffers to allocate (largest buffer possible for current mode of operation).
For example, an image with no statistics or stamp data as PayloadSize equals to (width x height x pixelsize) in bytes.

GevVersion **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Version of the GigE Standard with which the device is compliant
The two most-significant bytes are allocated to the major number of the version,
the two least-significant bytes for the minor number.

GevDeviceMode **Guru** **ReadOnly**
Integer in the range [0, UINT_MAX]
Information about device mode of operation.
Returns Big Endian and UTF8.

GevDeviceMACAddressHigh **Beginner** **ReadOnly**
Integer in the range [0, UINT_MAX]
The two most-significant bytes of this area are reserved and will return 0.
Upper 2 bytes of the MAC address are stored in the lower 2 significant bytes of this register.
The MAC address of the camera can be read at the rear of the device.

GevDeviceMACAddressLow **Beginner** **ReadOnly**
Integer in the range [0, UINT_MAX]
Lower 4 bytes of the MAC address.
The MAC address of the camera can be read at the rear of the device.

GevNetworkCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Supported IP Configuration and PAUSE schemes.
Returns LLA|DHCP|PersistentIp.

GevNetworkConfiguration

Expert
Activated IP Configuration and PAUSE schemes.
This parameter can take these values:

String value	Numeric value	Comment
LLA	4	Local Link(APIPA) only
LLA_Persistent_IP	5	Local Link and Persistent IP
LLA_DHCP	6	Local Link and DHCP(default)
All	7	All modes

If set to LLA: on connection, the camera negociate an LLA Ip Address.
If set to LLA_DHCP: on connection, the camera tries to contact a DHCP server to acquire an Ip address. If no DHCP is present, the device fallback to LLA negociation.
If set to LLA_Persistent_IP: on connection, the camera tries to negociate an LLA Ip address. If the LLA negociation fail, the device fallback to Persistent IP.
If set to All: The device tries to connect with DHCP, with LLA and finally takes the persistent IP.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", "LLA" );
```

You can also set these values as unsigned int value(in this case the value is the numeric value of the enumeration):

```
Example: dev.setParamValueOf(
"GevNetworkConfiguration", 5 );
```

Emit:

- NITConfigObserver.paramChanged 'GevNetworkConfiguration'.

Throw:

- NITException if the string value or

the numeric value doesn't exist.

GevCurrentIPAddress
Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current IP address of this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentSubnetMask
Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current subnet mask used by this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevCurrentDefaultGateway
Beginner **ReadOnly**
Integer in the range [0, UINT_MAX]
Current default gateway used by this device on its first interface.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

GevManufacturerName
Beginner **ReadOnly**
Provides the name of the manufacturer of the device.
Returns the string: "New-Imaging-Technologies"

GevModelName
Beginner **ReadOnly**
Provides the model name of the device.
Returns the string "NSC1601GigE".

GevDeviceVersion
Beginner **ReadOnly**
Provides the version of the device.
Returns a string whith the format "MM.mm"
Example: "01.05"

GevManufacturerInfo
Beginner **ReadOnly**
Provides extended manufacturer information about the device.
Returns the string "Camera NSC1601 from NIT"

GevSerialNumber
Beginner **ReadOnly**
Contains the serial number of the device.
It can be used to identify the device.
Returns a string with the format "NNNNNN"
Example: "200162"

GevUserDefinedName
Beginner
Null terminated string with maximum 16 characters(terminating null included).

This string contains a user-programmable name to assign to the device.
It can be used to identify the device.
Use the string form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevUserDefinedName", "MyCamera" );
```

Emit:

- NITConfigObserver.paramChanged 'GevUserDefinedName'.

Throw:

- NITException if the string is to long.

GevFirstURL

Guru ReadOnly
This register stores the first URL to the XML device description file.
The first URL is used as the first choice by the application to retrieve the XML device description file.
Returns a string with the format <filename>;<address>;<file size>
Example: "local:adr_nios_1201.zip;A200;1938"

GevSecondURL

Guru ReadOnly
This register stores the second URL to the XML device description file.
This URL is an alternative if the application was unsuccessful to retrieve the device description file using the first URL.

GevNumberOfInterfaces

Expert ReadOnly
Integer in the range [0, UINT_MAX]
This register indicates the number of network interfaces supported by this device.
This is generally equivalent to the number of Ethernet connectors on the device, except when Link Aggregation is used. In this case, the physical interfaces regrouped by the Aggregator are counted as one virtual interface.
Returns allways 1.

GevLinkSpeed

Expert ReadOnly
Integer in the range [0, UINT_MAX]
Value indicating current Ethernet link speed in Mbits per second. Gigabit ethernet returns 1000.

GevMessageChannelCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of message channels supported by this device.
Returns allways 0.

GevStreamChannelCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of stream channels supported by this device.
Returns allways 1.

GevActiveLinkCount **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the number of physical links that are currently active. A link is considered active as soon as it is connected to another Ethernet device. This happens after Ethernet link negotiation is completed.
Returns allways 1.

GevStreamChannelCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
First Stream Channel Capability register.
Returns SCSPx_supported|legacy_16bit_block_id_supported.

GevMessageChannelCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the capabilities of the message channel. It lists which of the non-mandatory message channel features are supported.
Returns allways 0.

GevGVCPCapability **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
This is a capability register indicating which one of the non-mandatory GVCP features are supported by this device. When supported, some of these features are enabled through the GVCP Configuration register.
Returns user_defined_name|serial_number|link_speed_register|WRITEMEM.

GevHeartbeatTimeout **Guru**
Integer in the range [0, UINT_MAX] milliseconds.
Indicates the current heartbeat timeout in milliseconds.
Default is 3000 msec.

Internally, the heartbeat is rounded according to the clock used for heartbeat.
The heartbeat timeout shall have a minimum precision of 100 ms.
The minimal value is 500 ms.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout", 2000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevHeartbeatTimeout", "1500" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevGVCPPendingTimeout

Expert **ReadOnly**
Integer in the range [0, UINT_MAX] milliseconds.
Pending Timeout to report the longest GVCP command execution time before issuing a PENDING_ACK. If PENDING_ACK is not supported, then this is the worstcase execution time before command completion.
Returns 300000.

GevPhysicalLinkConfigurationCapability

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the physical link configuration supported by this device. Returns 0.

GevPhysicalLinkConfiguration

Expert **ReadOnly**
Integer in the range [0, UINT_MAX]
Indicates the currently active physical link configuration.
Returns 0.

GevCCP

Guru
Integer in the range [0, UINT_MAX]
This register is used to grant privilege to an application.
Only one application is allowed to control the device. This application is able to write into device's registers. Other applications can read device's register only if the controlling application does not have the exclusive privilege.

On construction of the NITDevice the SDK set this parameter to exclusive access.

It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevCCP", 2 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevCCP", "2" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCPHostPort

Guru
Integer in the range [0, UINT_MAX]
Stream Channel Host Port register.
The host port to which a stream channel must send data stream.
Setting this value to 0 closes the stream channel.
The frame number is reset to 1 when the stream channel is opened.

This parameter is managed by the SDK.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPHostPort", 12000
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPHostPort",
"14200" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCSP

Guru ReadOnly
Integer in the range [0, UINT_MAX]
Stream Channel Source Port register.
The port on which the stream channel emits packets.

GevSCDA

Guru
Integer in the range [0, UINT_MAX]
Stream Channel destination Ip address of the receiving host.

The Ip address to which a stream channel must send data stream.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.

On construction of the NITDevice the SDK set this parameter to the IP address of the host.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCDA", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCDA", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentIPAddress

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent IP address for the given network interface.
Only used when the device is set to use the Persistent IP configuration scheme. Each byte represent a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentIPAddress", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentSubnetMask

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme.

Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentSubnetMask", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevPersistentDefaultGateway

Beginner
Integer in the range [0, UINT_MAX]
Indicate the persistent subnet mask associated with the persistent IP address on the given network interface. Only used when the device is set to use the Persistent IP configuration scheme.
Each byte represents a part of the Ip address.
Example: 192.168.2.15 is coded 0xC0A8020F.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", 3232236047 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf(
"GevPersistentDefaultGateway", "3232236047" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

GevSCPSPacketSize

Expert
Integer in the range [29, 8100] bytes
Size of the stream packets to transmit during acquisition.

On construction of the NITDevice the SDK negociate the highest possible packet size with the device.
It is not recommended to modifiy this parameter.

Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "GevSCPSPacketSize",
8000 );
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "GevSCPSPacketSize",
"8000" );
```

Throw:

- NITException if the string value or the numeric value is out of range.

TimeStamp_frequency_high **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
High part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the most significant bytes.

TimeStamp_frequency_low **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Low part of a 64-bit value indicating the number of timestamp clock tick in 1 second.
This register holds the least significant bytes.

TimestampValueHigh **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (most significant bytes). The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current frame.

TimestampValueLow **Expert** **ReadOnly**
Integer in the range [0, UINT_MAX]
Latched value of the timestamp (least significant bytes) The timestamp value can be accessed by calling NITFrame.gigeTimestamp() on the current frame.

TimestampControl **Expert** **WriteOnly**
Integer in the range [0, 3]
Write 1 to reset timestamp 64-bit counter to 0.
Write 2 to latch current timestamp counter into TimestampValue registers.
Write 3 to first latch current timestamp counter into TimeStampValue registers and then reset.
Use the numeric form of setParamValueOf to set these parameters.

```
Example: dev.setParamValueOf( "TimestampControl", 1
);
```

You can also set these values as string value:

```
Example: dev.setParamValueOf( "TimestampControl", "3"
);
```

Throw:

- NITException if the string value or the numeric value is out of range.

21. Steps to write a script (Sample Codes)

To illustrate how to write a script, we provide two sample codes:

- **NITBeginner**: demonstrates how to connect to a camera and run the streaming to capture frames and display them.
- **NITAdvanced**: explains the basic usage of pipelines.

A sample script using the NITLibrary (organization of the sample codes)

To implement a program dealing with NIT cameras whe have to follow this steps:

- Instantiate a **NITConfigObserver** object to be informed of changes in the camera(this is not mandatory).
- Create a camera object.
- Configure the camera(send parameters).
- Connect a pipeline to the camera to capture and manage frames.
- Run the capture.

Some of those steps are common to all cameras(eg. build a pipeline, run the capture).
Some steps depends of the camera model(eg. configure the camera).
And finally some steps depends of the connector type(Gige,Usb)(eg. create the camera).

As a result, a minimal sample script should look something like this:

```
import NITLibrary_x64_320_py36 as NITLibrary
# The line above import the NITLibrary
#( you must adapt 320 to the version of the NITLibrary and 36 to the version of Python )
import numpy                                                    # This import numpy

# Step 1 : Open Connected Device
# NITManager manage the connections to the cameras.
# It's the first object we deal with
# NITManager is a singleton instantiated on the first call to getInstance()

nm = NITLibrary.NITManager.getInstance() #Get unique instance of NITManager

print( nm.listDevices() )  # List the discovered devices

# Open one of connected device (if it exists)

 dev = nm.openDevice(0)  # if you have only one device you can also call dev =
nm.openOneDevice()
#Other open functions exists.
if( dev == None ):
    print("No Device Connected")
    return

#Step 2 : Device Configuration
#You can get the current parameter values in two ways

# By asking for a string value
print("Exposure:" + dev.paramStrValueOf( "Exposure Time" ) )
# By asking for a numerice value
print("PixelClock:" + str( dev.paramValueOf( "Pixel Clock" ) ) )


#You can set parameters values in two ways

dev.setParamValueOf( "ExposureTime", 20000 ) #By numeric value or
```

```
dev.setParamValueOf( "ExposureTime", "20ms" ) #By string value
#Data is not sent to the device until updateConfig is called
dev.updateConfig()

#You can also write
#dev.setParamValueOf( param1, value1 ).setParamValueOf( param2, value2 ).setParamValueOf(
paramN, valueN ).updateConfig()


#Some parameters are dependant on the connector model of the camera (Usb, Gige, …)
if (dev.connectorType() == NITLibrary.GIGE):#To use myAGC camera must provide a RAW data; as
Gige cameras can output different data types,we force RAW output type
    print("GIGE CAM")
    dev.setParamValueOf("OutputType", "RAW").updateconfig()

#Fps configuration: set fps to a mean value
min_fps = dev.minFps()
max_fps = dev.maxFps()
print("Frame rate: " + str(dev.minFps())+" <= " + str(dev.fps()) + " <= " + str(dev.maxFps())
)

dev.setFps( (min fps + max fps)/2 )    #Data is not sent to the device
dev.updateConfig()                     #Data is sent to the device

print("New Frame Rate: " + str(dev.fps()) )

#Step 3: Connect pipeline
# Once our device is created and configured; we have to connect it to a pipeline to process
the frames outputted by the camera.
# As our samples have for purpose to display frames in a viewer, we always need a NITPlayer
object to display the frames.
# Some cameras like Gige cameras can output frames directly in a displayable format. On the
other hand Usb cameras output RAW data who as to be processed before display.

myAGC = NITLibrary.NITToolBox.NITAutomaticGainControl()   #A NITFilter which perform Automatic
Gain Control on Frames
myPlayer = NITLibrary.NITToolBox.NITPlayer( "My Player" ) #An NITObserver which display Frames
in a window named "My Player"

dev << myAGC << myPlayer # The Agc and Player objects are connected to the NITDevice

#Step 4: Run the capture
#Finally we have to actually capture frames.
#NITLibrary provide 3 capture modes.
# -- Continuous capture.
# -- Capture for an amount of time.
# -- Capture a number of frames.
# In our minimal sample, we choose to play with this 3 modes.

#Start Capture!!!
#Continuous
print("First Way")
dev.start()                                #Start Capture
input("Press Enter to stop capture...")  # Wait for user to push enter key
dev.stop()                                 #Stop Capture


# Number of frames
print("Second Way")
dev.captureNFrames( 100 )     # Capture 100 Frames
dev.waitEndCapture()          # Wait end of capture


#For duration
print("Third Way")
dev.captureForDuration( 5000 )     #Capture for 5 seconds (time in ms)
dev.waitEndCapture()               #Wait end of capture
```

The last code snippet above is of the **NITBeginner** sample code.
The **NITAdvanced** sample code introduces the use of a user filter and NITConfigObserver. In addition to the work done in **NITBeginner**:

- a MyUserFilter class is declared and defined.

- a userFilter object is instantiated andconnected to the pipeline.

For **NITAdvanced**, the code above becomes:

```
import NITLibrary_x64_320_py39 as NITLibrary
# Adapt line above to your version of NITLibrary and Python
import numpy

#Creation of our custom filters and observers
class myPyFilter(NITLibrary.NITUserFilter):  # A filter who invert the pixel values
    def onNewFrame(self, frame):   #You MUST define this function - It will be called on each
frames
        if( frame.pixelType() == NITLibrary.ePixType.FLOAT ) :      #if we have a grayscale
image
            new_array = 1.0 - frame.data() #frame.data() gives access to the first pixel
        else : #we have a RGBA image
            new_array = numpy.invert(frame.data())
        return new_array        #Don't forget to return the resulting array


class myPyObserver(NITLibrary.NITUserObserver):  # An observer who display a string for each
frame
    def onNewFrame(self, frame):  #You MUST define this function
        "Your code goes here"
        print( "frame used " + str(frame.id()) )


#We report here only the step 3 steps 1 2 and 4 are identical with the code above.
#Step 3: Connect pipeline
# Once our device is created and configured; we have to connect it to a pipeline to process
the frames outputted by the camera.
# As our samples have for purpose to display frames in a viewer, we always need a NITPlayer
object to display the frames.
# Some cameras like Gige cameras can output frames directly in a displayable format. On the
other hand Usb cameras output RAW data who as to be processed before display.

mpf = myPyFilter()
myAGC = NITLibrary.NITToolBox.NITAutomaticGainControl()
myAGCPlayer = NITLibrary.NITToolBox.NITPlayer( "(AGC) Player" )
myFilterPlayer = NITLibrary.NITToolBox.NITPlayer( "(AGC + MyFilter) Player")

dev << myAGC << myAGCPlayer
             myAGC << mpf << myFilterPlayer

#myAGC is connected in one branch to a player and on another branch to our user filter and to
another player.
```

22. Release Notes

Version numbers follow the version number of the C++ NITLibrary on which this Python library is based.

- 2021-10-15 Version 3.2.0
  - First official release.

23. Document Revision

Document revision follows the library versioning followed by a letter.
The letter is incremented each time the documentation is revised without a modification of the library.
When major or minor version of the library is incremented the document revision is reset.

- *2021-10-15 Version 3.2.0-A*
  - *First official release.*

NIT
1 Impasse de la noisette, Bat D- 1er étage
BP426 91370 Verrières Le Buisson Cedex
France

Phone: +33(0) 1 64 47 88 58
www.new-imaging-technologies.com
support@new-imaging-technologies.com

NIT, New Imaging Technologies and Native WDR, are trademarks of New Imaging Technologies