

**ConoscopeSDK API python**

|  |  |
| --- | --- |
| Abstract | The aim of this document is to list the API of conoscope SDK |
| Version | 0.5 |
| Status | Draft |
| Date | 2020/06/29 |

Revision history

|  |  |  |
| --- | --- | --- |
| Version | Date | Content |
| 0.1 | 2020/03/11 | Initial version |
| 0.2 | 2020/04/14 | Conoscope.Iris enum has changed  Add CfgFile APIs  Add capture sequence APIs |
| 0.3 | 2020/06/04 | Update SetConfig options Add MeasureAE API |
| 0.4 | 2020/06/29 | Update SetConfig Update CaptureSequence parameters |
| 0.5 | 2020/11/17 | Update Set/Get config parameters |
|  |  |  |

|  |  |
| --- | --- |
|  | ELDIM 1185 Rue d’Epron (Ancienne) 14200 Hérouville Saint-Clair France |
|  | Copyright © 2020 All rights reserved. Printed in France. |
|  | ELDIM, the ELDIM logo and other product names referenced herein are trademarks of ELDIM.  Other product names, designations, logos, and symbols are trademarks or registered trademarks of their respective owners. |
|  | NO WARRANTY. The technical documentation is being delivered to you AS-IS and ELDIM makes no warranty as to its accuracy or use. Any use of the technical documentation or the information contained therein is at the risk of the user. Documentation may include technical or other inaccuracies or typographical errors. ELDIM reserves the right to make changes without prior notice. |
|  | ELDIM considers information included in this documentation as Confidential Information. The access and use of this confidential information are subject to the terms and conditions of the Software license agreement, with which you agree to comply.  This documentation cannot be reproduced in any way without the prior agreement and written permission of ELDIM. |

Table of contents

[1 Introduction 7](#_Toc56507490)

[2 Instantiation 7](#_Toc56507491)

[2.1 Description 7](#_Toc56507492)

[2.2 Declaration 7](#_Toc56507493)

[2.3 Remark 7](#_Toc56507494)

[3 GetVersion 7](#_Toc56507495)

[3.1 Description 7](#_Toc56507496)

[3.2 Declaration 7](#_Toc56507497)

[3.3 Return dict 7](#_Toc56507498)

[4 GetConfig 8](#_Toc56507499)

[4.1 Description 8](#_Toc56507500)

[4.2 Declaration 8](#_Toc56507501)

[4.3 Return dict 8](#_Toc56507502)

[5 SetConfig 9](#_Toc56507503)

[5.1 Description 9](#_Toc56507504)

[5.2 Declaration 9](#_Toc56507505)

[5.3 Parameter dict 9](#_Toc56507506)

[5.4 Return dict 10](#_Toc56507507)

[5.5 Usage 10](#_Toc56507508)

[6 GetDebugConfig 11](#_Toc56507509)

[6.1 Description 11](#_Toc56507510)

[6.2 Declaration 11](#_Toc56507511)

[6.3 Return dict 11](#_Toc56507512)

[7 SetDebugConfig 12](#_Toc56507513)

[7.1 Description 12](#_Toc56507514)

[7.2 Declaration 12](#_Toc56507515)

[7.3 Parameter dict 12](#_Toc56507516)

[7.4 Return dict 12](#_Toc56507517)

[7.5 Usage 12](#_Toc56507518)

[8 Open 13](#_Toc56507519)

[8.1 Description 13](#_Toc56507520)

[8.2 Declaration 13](#_Toc56507521)

[8.3 Return dict 13](#_Toc56507522)

[9 Setup 14](#_Toc56507523)

[9.1 Description 14](#_Toc56507524)

[9.2 Parameter dict 14](#_Toc56507525)

[9.3 Remark 14](#_Toc56507526)

[9.4 Return dict 14](#_Toc56507527)

[10 SetupStatus 15](#_Toc56507528)

[10.1 Description 15](#_Toc56507529)

[10.2 Declaration 15](#_Toc56507530)

[10.3 Return dict 15](#_Toc56507531)

[11 CmdMeasure 16](#_Toc56507532)

[11.1 Description 16](#_Toc56507533)

[11.2 Declaration 16](#_Toc56507534)

[11.3 Parameter dict 16](#_Toc56507535)

[11.4 Return dict 16](#_Toc56507536)

[12 MeasureAE 17](#_Toc56507537)

[12.1 Description 17](#_Toc56507538)

[12.2 Declaration 17](#_Toc56507539)

[12.3 Parameter dict 17](#_Toc56507540)

[12.4 Return dict 17](#_Toc56507541)

[13 MeasureAEStatus 17](#_Toc56507542)

[13.1 Description 17](#_Toc56507543)

[13.2 Declaration 17](#_Toc56507544)

[13.3 Return dict 17](#_Toc56507545)

[14 MeasureAECancel 18](#_Toc56507546)

[14.1 Description 18](#_Toc56507547)

[14.2 Declaration 18](#_Toc56507548)

[14.3 Return dict 18](#_Toc56507549)

[15 ExportRaw 18](#_Toc56507550)

[15.1 Description 18](#_Toc56507551)

[15.2 Declaration 18](#_Toc56507552)

[15.3 Return dict 18](#_Toc56507553)

[16 ExportProcessed 19](#_Toc56507554)

[16.1 Description 19](#_Toc56507555)

[16.2 Declaration 19](#_Toc56507556)

[16.3 Return dict 19](#_Toc56507557)

[17 CfgFileRead 20](#_Toc56507558)

[17.1 Description 20](#_Toc56507559)

[17.2 Declaration 20](#_Toc56507560)

[17.3 Return dict 20](#_Toc56507561)

[18 CfgFileStatus 20](#_Toc56507562)

[18.1 Description 20](#_Toc56507563)

[18.2 Declaration 20](#_Toc56507564)

[18.3 Return dict 20](#_Toc56507565)

[19 CaptureSequence 21](#_Toc56507566)

[19.1 Description 21](#_Toc56507567)

[19.2 Parameter dict 21](#_Toc56507568)

[19.3 Declaration 21](#_Toc56507569)

[19.4 Return dict 21](#_Toc56507570)

[20 CaptureSequenceCancel 22](#_Toc56507571)

[20.1 Description 22](#_Toc56507572)

[20.2 Declaration 22](#_Toc56507573)

[20.3 Return dict 22](#_Toc56507574)

[21 CaptureSequenceStatus 22](#_Toc56507575)

[21.1 Description 22](#_Toc56507576)

[21.2 Declaration 22](#_Toc56507577)

[21.3 Return dict 22](#_Toc56507578)

[22 GetCaptureSequence 23](#_Toc56507579)

[22.1 Description 23](#_Toc56507580)

[22.2 Declaration 23](#_Toc56507581)

[22.3 Return dict 23](#_Toc56507582)

[23 Close 23](#_Toc56507583)

[23.1 Descrition 23](#_Toc56507584)

[23.2 eclaration: 23](#_Toc56507585)

[23.3 Return dict 23](#_Toc56507586)

[24 Reset 24](#_Toc56507587)

[24.1 Description 24](#_Toc56507588)

[24.2 Declaration: 24](#_Toc56507589)

[24.3 Return dict 24](#_Toc56507590)

[25 QuitApplication 24](#_Toc56507591)

[25.1 Description 24](#_Toc56507592)

[25.2 Declaration 24](#_Toc56507593)

[26 Example: 25](#_Toc56507594)

# Introduction

This interface is defined in **Conoscope.py** file.

It should be imported:

from Conoscope import Conoscope

# Instantiation

## Description

Use this class to get an instance of the conoscope device

## Declaration

Conoscope()

## Remark

A new thread will be created inside Conoscope class.  
It is handled internally  
Function *QuitApplication* must be called to terminate it correctly.

# GetVersion

## Description

This function returns the version of DLL used by the module

## Declaration

CmdGetVersion()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “Lib\_Name” | String  i.e. “CONOSCOPE\_LIB” |
| “Lib\_Date” | String  i.e. “2020/03/10” |
| “Lib\_Version” | String  i.e. “0.8.18” |
| “Pipeline\_Name” | String  i.e. “PIPELINE\_LIB” |
| “Pipeline\_Date” | String  i.e. “2020/03/05” |
| Pipeline\_Version | String  i.e. “0.2.5” |

# GetConfig

## Description

Returns current configuration of the conoscope device

## Declaration

CmdGetConfig()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “CfgPath” | String  Folder where configuration data are  i.e. “./Cfg” |
| “CapturePath” | String  Folder where capture are stored  i.e. “./CaptureFolder” |
| “FileNamePrepend” | String  String to be added at the begging of the exported file name  Default value is “” |
| “FileNameAppend” | String  String to be added at the end of the exported file name  Default value is “” |
| “ExportFileNameFormat” | String  FileName format (Not implemented) Default value is “” |
| “ExportFormat” | Conoscope.ExportFomat type |
| "AEMinExpoTimeUs" | Integer  Minimum exposure time set by Auto Exposure |
| "AEMaxExpoTimeUs" | Integer  Maximum exposure time set by Auto Exposure |
| "AEExpoTimeGranularityUs" | Interger  Exposure time granularity for Auto Exposure |
| "AELevelPercent" | Float |
| "AEMeasAreaHeight" | Int  Auto Exposure ROI |
| "AEMeasAreaWidth" | Int  Auto Exposure ROI |
| "AEMeasAreaX" | Int  Auto Exposure ROI |
| "AEMeasAreaY" | Int  Auto Exposure ROI |
| "bUseRoi" | Boolean  Use of ROI for export commands |
| "RoiXLeft" | Int  Export ROI |
| "RoiXRight" | Int  Export ROI |
| "RoiYTop" | Int  Export ROI |
| "RoiYBottom" | Int  Export ROI |

# SetConfig

## Description

Change the configuration of the conoscope.

## Declaration

CmdSetConfig(parameterDict)

## Parameter dict

|  |  |
| --- | --- |
| Key | Value |
| “CfgPath” | String  Folder where configuration data are  i.e. “./Cfg” |
| “CapturePath” | String  Folder where capture are stored  i.e. “./CaptureFolder” |
| “FileNamePrepend” | String  String to be added at the begging of the exported file name  Default value is “” |
| “FileNameAppend” | String  String to be added at the end of the exported file name  Default value is “” |
| “ExportFileNameFormat” | String  FileName format (Not implemented) Default value is “” |
| “ExportFormat” | Conoscope.ExportFomat type  Defines the type of the exported file:  - Binary  - Binary + JPG |
| "AEMinExpoTimeUs" | Integer  Minimum exposure time set by Auto Exposure |
| "AEMaxExpoTimeUs" | Integer  Maximum exposure time set by Auto Exposure |
| "AEExpoTimeGranularityUs" | Interger  Exposure time granularity for Auto Exposure |
| "AELevelPercent" | Float |
| "AEMeasAreaHeight" | Int  Auto Exposure ROI |
| "AEMeasAreaWidth" | Int  Auto Exposure ROI |
| "AEMeasAreaX" | Int  Auto Exposure ROI |
| "AEMeasAreaY" | Int  Auto Exposure ROI |
| "bUseRoi" | Boolean  Use of ROI for export commands |
| "RoiXLeft" | Int  Export ROI |
| "RoiXRight" | Int  Export ROI |
| "RoiYTop" | Int  Export ROI |
| "RoiYBottom" | Int  Export ROI |

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

## Usage

CmdSetConfig({“capturePath”: “./CaptureFolder”, “cfgPath”: “./Cfg”}  
  
**cfgPath** can be changed **ONLY** before CmdOpen.  
**capturePath** can be changed at any time  
to change only capturePath:  
CmdSetConfig({“capturePath”: “./CaptureFolder”}

# GetDebugConfig

## Description

Returns the debug configuration

## Declaration

CmdGetDebugConfig()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
|  |  |
| “debugMode” | Should be set to false |
| “emulatedCamera” | Indicate the configuration of the camera |

# SetDebugConfig

## Description

Configure the debug option.

**This function must be called before CmdOpen to configure the camera in emulated mode.**

## Declaration

CmdSetDebugConfig(parameterDict)

## Parameter dict

|  |  |
| --- | --- |
| Key | Value |
| “CfgPath” | String  Folder where configuration data are  i.e. “./Cfg” |
| “CapturePath” | String  Folder where capture are stored  i.e. “./CaptureFolder” |
|  |  |
| “debugMode” | Boolean  Should be set to false |
| “emulatedCamera” | Boolean  Indicate the configuration of the camera |
| “dummyRawImagePath” | String  This is the path of the raw data to be processed when the camera is in emulated mode |
| “emulatedWheel” | Boolean Indicate if the wheel failure is taken into account. For debug purpose only |

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

## Usage

It is possible to configure only the raw image:  
CmdSetDebugConfig({“dummyRawImagePath”: “.\\CaptureImage\\20200310\_181207\_filt\_Yb\_nd\_4\_iris\_2\_proc\_1.bin”})

# Open

## Description

Open the conoscope instance. Connect to the device.

## Declaration

CmdOpen()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “CfgPath” | String  Folder where the configuration data will be accessed. Folder name is matching the folder set with CmdSetConfig + serial number of the device |

# Setup

## Description

Configure the Conoscope: set the temperature target and configure filters  
Declaration:  
CmdSetup(param)

## Parameter dict

|  |  |
| --- | --- |
| Key | Value |
| “sensorTemperature” | Float  Target of the regulated temperature (precision is around 2%)  i.e. 25.0 |
| “eFilter” | Conoscope.Filter type  Filter to be used  (The command will turn the wheel)  i.e. Conoscope.Filter.X.value |
| “eNd” | Conoscope.Nd type  Nd filter to be used  (The command will turn the wheel)  i.e. Conoscope.Nd.Nd\_0.value |
| “eIris” | Conoscope.Iris type  Iris installed on the device  i.e. Conoscope.Iris.aperture\_2mm.value |

## Remark

An enum is available for filter/nd/iris (Conoscope.Filter, Conoscope.Nd, Conoscope.Iris)

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# SetupStatus

## Description

Return current status of the conoscope.  
Setting the temperature with CmdSetup is not instantaneous:  
it may take some time to be stable depending on the expected temperature and the temperature at the time it is set

## Declaration

CmdSetupStatus()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “eTemperatureMonitoringState” | Conoscope.TemperatureMonitoringState  Indicate the status of the temperature regulation  **TemperatureMonitoringState\_Processing** indicates the regulation is on going and the target is not reached yet  **TemperatureMonitoringState\_Locked** indicates the temperature is locked  **TemperatureMonitoringState\_NotStarted** indicates setup has not been done  **TemperatureMonitoringState\_Aborted** indicates processing has been stopped  **TemperatureMonitoringState\_Error** indicates an error happened |
| “sensorTemperature” | Float  Current temperature |
| “eWheelState” | Conoscope.WheelState  Indicate the status of the wheel  **WheelState\_Idle** indicates wheel has not been configured  **WheelState\_Success** indicates last setup was ok  **WheelState\_Operating** indicates setup was on going  **WheelState\_Error** indicates an error happen |
| “eFilter” | Current filter |
| “eNd” | Nd filter |
| “eIris” | Iris |

# CmdMeasure

## Description

Perform the measurement.  
At this point data are stored internally.

## Declaration

CmdMeasure()

## Parameter dict

|  |  |
| --- | --- |
| Key | Value |
| “exposureTimeUs” | Int  Exposure time in microseconds |
| “nbAcquisition” | Int  Number of acquisitions  Exported image will be the average of those captures |

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# MeasureAE

## Description

Perform the measurement. An auto exposure is used.

This command is asynchronous. Once the command is launched, a polling must be done with MeasureAEStatus command to know when the processing is finished.

At This point data are stored internally.

## Declaration

CmdMeasure()

## Parameter dict

|  |  |
| --- | --- |
| Key | Value |
| “exposureTimeUs” | Int  Exposure time in microseconds. It will be the start value of the algorythm |
| “nbAcquisition” | Int  Number of acquisitions  Exported image will be the average of those captures |

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# MeasureAEStatus

## Description

This command returns the status of MeasureAE. Use this command to check the completion of MeasureAE.

## Declaration

CmdMeasureAEStatus()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “ExposureTimeUs” | Current exposure time |
| “NbAcquisition” | Configured nb acquisition |
| “State” | Conoscope.MeasureAEState type  Is the state of the processing.  MeasureAEState\_Process indicate that the measurement is on going |

# MeasureAECancel

## Description

This command stops pending MeasureAE.

## Declaration

CmdMeasureAECancel()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# ExportRaw

## Description

Export raw data of last measurement done.

The file will be store in the folder configured with CmdSetConfig

## Declaration

CmdExportRaw()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “CaptureFile” | String  Capture File name CaptureFolder1/19041841/20200311\_153127\_filt\_X\_nd\_0\_iris\_2\_raw\_1.bin |
| « ExposureUs » | Integer  Exposure time in micro second |
| « NbAcquisition » | Integer |
| « Filter » | Conoscope.Filter type |
| « Nd » | Conoscope.Nd type |
| « Iris » | Conoscope.Iris type |
| « SensorTemperature » | Float  Temperature at the capture time |
| “Height” | Interger  i.e. 6004 |
| “Width” | Integer  i.e. 7920 |

# ExportProcessed

## Description

Export processed data of last measurement done.

The file will be store in the folder configured with CmdSetConfig

## Declaration

CmdExportProcessed()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “CaptureFile” | String  Capture File name CaptureFolder1/19041841/20200311\_153127\_filt\_X\_nd\_0\_iris\_2\_proc\_1.bin |
| « ExposureUs » | Integer  Exposure time in micro second |
| « NbAcquisition » | Integer |
| « Filter » | Conoscope.Filter type |
| « Nd » | Conoscope.Nd type |
| « Iris » | Conoscope.Iris type |
| « SensorTemperature » | Float  Temperature at the capture time |
| “Height” | Interger  i.e. 6004 |
| “Width” | Integer  i.e. 7920 |
| “ConversionFactor” | Float  ConversionFactor  i.e. 0.23040654296875002 |
|  |  |
| “CameraCfgFile” | String  For information, indicate calibration file used  “CameraCfgFile : Cfg/19041841/AIRSHIP\_19041841.cfg” |
| “CameraCfgValid” | Boolean  Indicate if there is a problem with this file |
| “OpticalColumnCfgFile“ | String  For information, indicate calibration file used  “OpticalColumnCfgFile: Cfg/19041841/OpticalColumn.xml” |
| “OpticalColumnCfgValid” | Boolean  Indicate if there is a problem with this file |
| “FlatFieldFile” | String  For information, indicate calibration file used  “Cfg/19041841/FlatField\_iris\_2\_filter\_X.bin” |
| “FlatFieldValid” | Boolean  Indicate if there is a problem with this file |

# CfgFileRead

## Description

Save configuration data from the conoscope in the folder return by open API (“CfgPath”).

This function is asynchronous because it takes about 30 minutes to complete.

Use CfgFileStatus to know the status of the upload.

## Declaration

CmdCfgFileRead()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# CfgFileStatus

## Description

Get the status of CfgFileRead

## Declaration

CmdCfgFileStatus()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| “eState” | Conoscope. CfgFileState |
| “progress” | In percent |
| “elapsedTime” | Time spent in sec |
| “filename” |  |

# CaptureSequence

## Description

The goal of this feature is to capture 5 images with filters X, Xz, Ya, Yb, Z  
And create 3 images X, Y, Z by composing them together, according to the configuration data.

## Parameter dict

|  |  |
| --- | --- |
| Key | Value |
| “sensorTemperature” | Sensor temperature configuration |
| “bWaitForSensorTemperature” | Indicate whether the capture must wait for temperature to be set (may take 2 minutes) |
| “eNd” | Conoscope.Nd type |
| “eIris“ | Conoscope.Iris type |
| “exposureTimeUs“ | Exposure time in micro seconds |
| “nbAcquisition“ | Nb of acquisitions |
| “bAutoExposure” | Use autoexposure (instead of exposure time parameter) |
| “bUseExpoFile“ | Use a json file with predefined exposure time for each capture |
| “bSaveCapture“ | Captures required to generate composite images are stored |

Note that:

- “exposureTimeUs” is valid only if “bAutoExposure” and “bUseExpoFile” are False.

- “bAutoExposure” and “bUseExpoFile” cannot be True at the same time

## Declaration

CmdCaptureSequence(param)

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# CaptureSequenceCancel

## Description

Cancel on going Capture Sequence

## Declaration

CmdCaptureSequenceCancel()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

Remark

Please check Capture Sequence state with **CaptureSequenceStatus** API to be sure the processing is finished.

(This function can take some time)

# CaptureSequenceStatus

## Description

Returns the status of Capture Sequence feature

## Declaration

CmdCaptureSequenceStatus()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| "nbSteps" | Number of steps of the complete capture sequence |
| "currentSteps", | Current step |
| "eFilter" | Current value of the filter |
| "state” | Current capture state (Conoscope. CaptureSequenceState type) |

# GetCaptureSequence

## Description

Get the settings of capture sequence (this API may not be necessary)

## Declaration

CmdGetCaptureSequence()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |
| "sensorTemperature" | Sensor temperature configuration (not the actual value) |
| "bWaitForSensorTemperature" | Indicate whether the capture must wait for temperature to be set (may take 2 minutes) |
| "eNd" | Conoscope.Nd type |
| "eIris" | Conoscope.Iris type |
| "exposureTimeUs" | Exposure time in microseconds |
| "nbAcquisition" | Nb of acquisitions |
| "bAutoExposure" | Use autoexposure (instead of exposure time parameter) |
| "bUseExpoFile" | Use a json file with predefined exposure time for each capture |

# Close

## Descrition

Close the conoscope.

CmdOpen will be necessary to use the conoscope again

## eclaration:

CmdClose()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# Reset

## Description

Reset the Conoscope and open the connection automatically.

Note this function take several time

## Declaration:

CmdReset()

## Return dict

|  |  |
| --- | --- |
| Key | Value |
| “Error” | Integer 0 means ok |
| “Message” | String  i.e. “Ok” |

# QuitApplication

## Description

Terminate the instance.

To be called before deleting conoscope instance

## Declaration

QuitAplication()

# Example:

conoscope = Conoscope()

ret = conoscope.CmdGetVersion()

LogFunction(ret, "CmdGetVersion")

ret = conoscope.CmdGetConfig()

LogFunction(ret, "CmdGetConfig")

config = dict()

config["capturePath"] = "./CaptureTest"

config["cfgPath"] = "./Cfg"

ret = conoscope.CmdSetConfig(config)

LogFunction(ret, "CmdSetConfig")

#executeFunction(CmdGetInfo(), "CmdGetInfo")

ret = conoscope.CmdOpen()

LogFunction(ret, "CmdOpen")

setupConfig = dict()

setupConfig["sensorTemperature"] = 25.0

setupConfig["eFilter"] = Conoscope.Filter.X.value

setupConfig["eNd"] = Conoscope.Nd.Nd\_0.value

setupConfig["eIris"] = Conoscope.Iris.aperture\_2mm.value

ret = conoscope.CmdSetup(setupConfig)

LogFunction(ret, "CmdSetup")

ret = conoscope.CmdSetupStatus()

LogFunction(ret, "CmdSetupStatus")

time.sleep(5)

measureConfig = dict()

measureConfig["exposureTimeUs"] = 100000

measureConfig["nbAcquisition"] = 1

ret = conoscope.CmdMeasure(measureConfig)

LogFunction(ret, "CmdMeasure")

ret = conoscope.CmdExportRaw()

LogFunction(ret, "CmdExportRaw")

ret = conoscope.CmdExportProcessed()

LogFunction(ret, "CmdExportProcessed")

ret = conoscope.CmdClose()

LogFunction(ret, "CmdClose")

# end the application

conoscope.QuitApplication()