# CONEX-AGAP

## Agilis-D Controller
## with Strain Gages Feedback


Newport®

# Command Interface Manual

V2.0.x

# Table of Contents


# Agilis-D Controller with Strain Gages Feedback CONEX-AGAP

## 1.0 Introduction

### 1.1 Purpose

The purpose of this document is to provide the method syntax of each command to communicate with the CONEX-AGAP device.

### 1.2 Overview

The Command Interface is the wrapper class that maintains a list of CONEX-AGAP instruments. It exposes methods to communicate with any CONEX-AGAP device.

These commands work in both synchronous and asynchronous mode in the NStruct environment or not. The communication is based on the NStruct server.

**NOTE**

**Each function name is defined with the command code "AA".**

**For each command function, refer to the CONEX-AGAP programmer's manual.**

# 2.0 Command Interface

## 2.1 Constructor

CONEXAGAP()

The constructor is used to create an instance of the CONEX-AGAP device.

## 2.2 Functions

### 2.2.1 General Functions

#### 2.2.1.1 RegisterComponent

**Syntax**

int RegisterComponent(string intrumentKey)

intrumentKey: Instrument key

return: componentID

**Description**

This function allows registering the device to the server. A component ID is returned. If the registering failed, the returned component ID is zero.

**NOTE**

**The component ID is mandatory to use all commands from the CommandInterface.**

#### 2.2.1.2 UnregisterComponent

**Syntax**

void UnregisterComponent(int componentID)

**Description**

This function allows unregistering the device to the server with the component ID.

#### 2.2.1.3 LockInstrument

**Syntax**

int LockInstrument(int componentID, int stage, ref string response)

**Description**

This function allows locking the device communication to not share the communication.

#### 2.2.1.4 UnlockInstrument

**Syntax**

int UnlockInstrument(int componentID, int stage, ref string response)

**Description**

This function allows unlocking the device communication to share the communication.

#### 2.2.1.5 GetDevices

**Description**

This function is used to get the list of the connected devices

**Syntax**

string[] GetDevices()

Return: string array to get the list of devices

#### 2.2.1.6 WriteToInstrument

**Syntax**

int WriteToInstrument(int componentID, string command, ref string response, int stage)

componentID: Instrument ID

command: Instrument command

response: Response of the command

stage: Instrument Stage

Return: Communication error code

**Description**

This Overridden function Queries or writes the command given by the user to the instrument.

### 2.2.2 Controller Command Functions

#### 2.2.2.1 ID_Get

**Syntax**

int ID_Get(int componentID, int controllerAddress, out string outStageIdentifier, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outStageIdentifier: outStageIdentifier

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous ID Get command which is used to Get stage identifier.

#### 2.2.2.2 ID_Set

**Syntax**

int ID_Set(int componentID, int controllerAddress, string inStageIdentifier, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inStageIdentifier: inStageIdentifier.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous ID Set command which is used to Set stage identifier.

#### 2.2.2.3 JA_Get

**Syntax**

int JA_Get(int componentID, int controllerAddress, string Axis, out double outJogVelocity, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outJogVelocity: outJogVelocity

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous JA Get command which is used to Move jogging.

#### 2.2.2.4 JA_Set

**Syntax**

int JA_Set(int componentID, int controllerAddress, string Axis, double inJogVelocity, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inJogVelocity: inJogVelocity.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous JA Set command which is used to Move jogging.

#### 2.2.2.5 MM_Get

**Syntax**

int MM_Get(int componentID, int controllerAddress, out string outState, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outState: outState

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous MM Get command which is used to Leave DISABLE state.

#### 2.2.2.6 MM_Set

**Syntax**

int MM_Set(int componentID, int controllerAddress, int inState, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inState: inState.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous MM Set command which is used to Leave DISABLE state.

#### 2.2.2.7 PA_Get

**Syntax**

int PA_Get(int componentID, int controllerAddress, string Axis, out double outTarget, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outTarget: outTarget

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous PA Get command which is used to Move absolute.

#### 2.2.2.8 PA_Set

**Syntax**

int PA_Set(int componentID, int controllerAddress, string Axis, double inTarget, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inTarget: inTarget.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous PA Set command which is used to Move absolute.

#### 2.2.2.9 PR_Get

**Syntax**

int PR_Get(int componentID, int controllerAddress, string Axis, out double outStep, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outStep: outStep

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous PR Get command which is used to Move relative.

#### 2.2.2.10 PR_Set

**Syntax**

int PR_Set(int componentID, int controllerAddress, string Axis, double inStep, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inStep: inStep.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous PR Set command which is used to Move relative.

#### 2.2.2.11 PW_Get

**Syntax**

int PW_Get(int componentID, int controllerAddress, out int outState, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outState: outState

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous PW Get command which is used to Enter/Leave CONFIGURATION state.

#### 2.2.2.12 PW_Set

**Syntax**

int PW_Set(int componentID, int controllerAddress, int inState, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inState: inState.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous PW Set command which is used to Enter/Leave CONFIGURATION state.

---

**NOTE**

**The PW command is limited to 100 writes. Unit failure due to excessive use of the PW command is not covered by warranty.**

**The PW command is used to change the configuration parameters that are stored in memory, and not parameters that are needed to be changed on the fly.**

---

#### 2.2.2.13 RS

**Syntax**

int RS(int componentID, int controllerAddress, out string errString)

clientID: Instrument ID

controllerAddress: controllerAddress identifying the Address of Controller

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous RS Set command which is used to Reset controller.

#### 2.2.2.14 RS485

**Syntax**

int RS485(int componentID, int controllerAddress, out string errString)

clientID: Instrument ID

controllerAddress: controllerAddress identifying the Address of Controller

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous RS## Set command which is used to Reset controller's address to 1.

#### 2.2.2.15 SA_Get

**Syntax**

int SA_Get(int componentID, int controllerAddress, out int outAdress, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outAdress: outAdress

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous SA Get command which is used to Get controller's RS-485 address.

#### 2.2.2.16 SA_Set

**Syntax**

int SA_Set(int componentID, int controllerAddress, int inAdress, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inAdress: inAdress.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous SA Set command which is used to Set controller's RS-485 address.

#### 2.2.2.17 SL_Get

**Syntax**

int SL_Get(int componentID, int controllerAddress, string Axis, out double outLimit, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outLimit: outLimit

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous SL Get command which is used to Get negative software limit.

#### 2.2.2.18 SL_Set

**Syntax**

int SL_Set(int componentID, int controllerAddress, string Axis, double inLimit, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inLimit: inLimit.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous SL Set command which is used to Set negative software limit.

#### 2.2.2.19 SR_Get

**Syntax**

int SR_Get(int componentID, int controllerAddress, string Axis, out double outLimit, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outLimit: outLimit

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous SR Get command which is used to Get positive software limit.

#### 2.2.2.20 SR_Set

**Syntax**

int SR_Set(int componentID, int controllerAddress, string Axis, double inLimit, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inLimit: inLimit.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous SR Set command which is used to Set positive software limit.

#### 2.2.2.21 ST

**Syntax**

int ST(int componentID, int controllerAddress, out string errString)

clientID: Instrument ID

controllerAddress: controllerAddress identifying the Address of Controller

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous ST Set command which is used to Stop motion

#### 2.2.2.22 TB

**Syntax**

int TB(int componentID, int controllerAddress, string inErrorCode, out string outErrorCode, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inErrorCode: inErrorCode.

outErrorCode: outErrorCode

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous TB Get command which is used to Get command error string.

#### 2.2.2.23 TE

**Syntax**

int TE(int componentID, int controllerAddress, out string outLastCommandError, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outLastCommandError: outLastCommandError

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous TE Get command which is used to Get last command error.

#### 2.2.2.24 TH

**Syntax**

int TH(int componentID, int controllerAddress, string Axis, out double outPosition, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outPosition: outPosition

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous TH Get command which is used to Get target position.

#### 2.2.2.25 TP

**Syntax**

int TP(int componentID, int controllerAddress, string Axis, out double outPosition, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outPosition: outPosition

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous TP Get command which is used to Get current position.

#### 2.2.2.26 TS

**Syntax**

int TS(int componentID, int controllerAddress, out string errorCode, out string controllerState, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

errorCode: errorCode

controllerState: controllerState

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous TS Get command which is used to Get positioner error and controller state.

#### 2.2.2.27 VE

**Syntax**

int VE(int componentID, int controllerAddress, out string outInformation, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outInformation: outInformation

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous VE Get command which is used to Get controller revision information.

#### 2.2.2.28 XU_Get

**Syntax**

int XU_Get(int componentID, int controllerAddress, string Axis, out double outStepSize, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outStepSize: outStepSize

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous XU Get command which is used to Get step size for STEPPING OL state.

#### 2.2.2.29 XU_Set

**Syntax**

int XU_Set(int componentID, int controllerAddress, string Axis, double inStepSize, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inStepSize: inStepSize.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous XU Set command which is used to Set step size for STEPPING OL state.

#### 2.2.2.30 XR_Get

**Syntax**

int XR_Get(int componentID, int controllerAddress, string Axis, out double outStep, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

outStep: outStep

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous XR Get command

which is used to Move stepping.

#### 2.2.2.31 XR_Set

**Syntax**

int XR_Set(int componentID, int controllerAddress, string Axis, double inStep, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

inStep: inStep.

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous XR Set command which is used to Move stepping.

#### 2.2.2.32 ZT

**Syntax**

int ZT(int componentID, int controllerAddress, out List<string> Parameters, out string errString)

componentID: Instrument ID

controllerAddress: Address of Controller

Parameters: Parameters

errString: The failure reason

Return: 0 in success and -1 on failure

**Description**

This function is used to process synchrounous ZT Get command which is used to Get all controller parameters.

## 3.0 Python Example

```
#==========================================================
#Initialization Start
#The script within Initialization Start and Initialization End is needed for properly
# initializing IOPortClientLib and Command Interface for CONEX-AGAP
instrument.
#The user should copy this code as is and specify correct paths here.
import sys
import time

#IOPortClientLib and Command Interface DLL can be found here.
print "Adding location of IOPortClientLib.dll &
Newport.CONEXAGAP.CommandInterface.dll to sys.path"
sys.path.append(r'C:\Program Files\Newport\Instrument
Manager\NStruct\Instruments\CONEX-AGAP\Bin')

# The CLR module provide functions for interacting with the underlying
# .NET runtime
import clr

# Add reference to assembly and import names from namespace
clr.AddReferenceToFile("Newport.CONEXAGAP.CommandInterface.dll")
from CommandInterface import *

import System
#==========================================================

# Instrument Initialization
# The key should have double slashes since
# (one of them is escape character)
instrument="CONEX-AGAP (A6TOO38G)"
print 'Instrument Key=>', instrument

# create AGAP instance
AGAP = CONEXAGAP()

# register to server, componentID needs to be used in all commands
componentID = AGAP.RegisterComponent(instrument);
print 'componentID=>', componentID

# variable setting
address = 1
axis = "U"

# Get controller revision information
result, response, errString = AGAP.VE(componentID,address)
if result == 0 :
  print 'controller revision=>', response
else:
  print 'Error=>',errString

# Get positive software limit
result, response, errString = AGAP.SR_Get(componentID,address,axis)
if result == 0 :
  print 'positive software limit=>', response
else:
  print 'Error=>',errString
```

```
# Get negative software limit
result, response, errString = AGAP.SL_Get(componentID,address,axis)
if result == 0 :
  print 'negative software limit=>', response
else:
  print 'Error=>',errString

# Get current position
result, response, errString = AGAP.TP(componentID,address,axis)
if result == 0 :
  print 'position=>', response
else:
  print 'Error=>',errString

print 'End of script'

# unregister server
AGAP.UnregisterComponent(componentID);
```

# Service Form

**Your Local Representative**

Tel.: _____

Fax: _____

Name: _____

Company: _____

Address: _____

Country: _____

P.O. Number: _____

Item(s) Being Returned: _____

Model#: _____

Return authorization #: _____
*(Please obtain prior to return of item)*

Date: _____

Phone Number: _____

Fax Number: _____

Serial #: _____

Description: _____

Reasons of return of goods (please list any specific problems): _____

Newport®

Visit Newport Online at:
www.newport.com


**North America & Asia**
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

**Sales**
Tel.: (800) 222-6440
e-mail: sales@newport.com

**Technical Support**
Tel.: (800) 222-6440
e-mail: tech@newport.com

**Service, RMAs & Returns**
Tel.: (800) 222-6440
e-mail: service@newport.com

**Europe**
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

**Sales**
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport.com

**Technical Support**
e-mail: tech_europe@newport.com

**Service & Returns**
Tel.: +33 (0)2.38.40.51.55


mks

Newport® Ophir® Spectra-Physics®