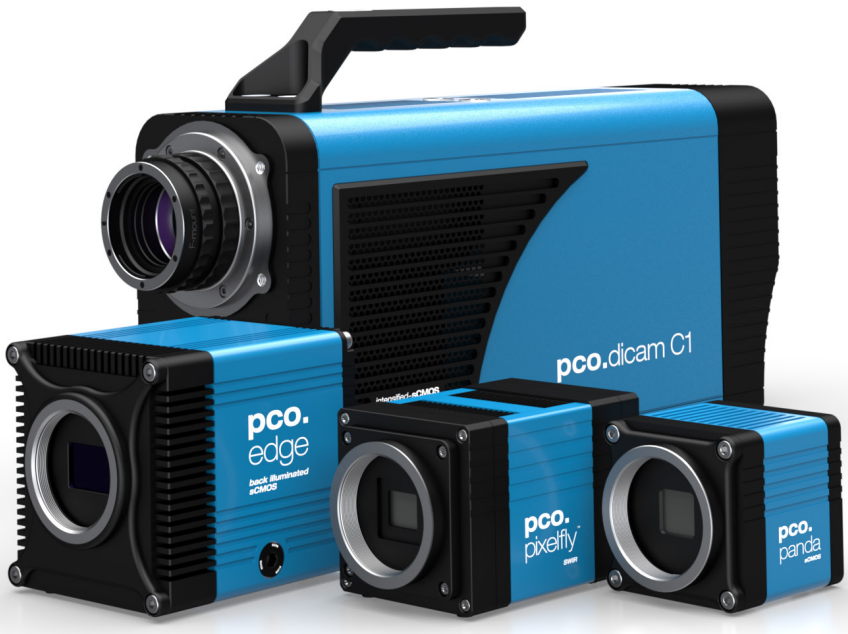
pco.®

pco.server

FOR WINDOWS AND LINUX



excelitas®

Excelitas PCO GmbH asks you to carefully read and follow the instructions in this document.
For any questions or comments, please feel free to contact us at any time.

address:	Excelitas PCO GmbH Donaupark 11 93309 Kelheim, Germany
phone:	(+49) 9441-2005-0 (+1) 866-662-6653 (+86) 400-080-2255
mail:	pco@excelitas.com
web:	www.excelitas.com/pco

pco.server user manual 1.0.1

Released August 2026

Contents

1 Introduction

The **pco.server** is designed to provide remote access to PCO cameras and X-Cite® light sources. It is intended for use with the PCO SDKs, which provide convenient access to camera functionality and handle communication with the server internally. In most applications, direct interaction with the REST API is therefore not required.

However, if direct API access is required, this document provides a concise overview of the available HTTP endpoints and their request and response formats. Most endpoints correspond directly to functions and data structures in the pco.cpp SDK. For complete definitions of functions, parameters, and data structures, refer to the pco.cpp manual.

The latest version of pco.server and the required drivers are available on our website.

1.1 Installation

1.1.1 Windows

Download the ***pco.server Windows*** package, unzip it, and execute it. Follow the installation wizard:

1. Select *"Install for anyone using this computer"* to install to the Program Files folder, otherwise it will be installed only to your user folder.
2. In the ***Choose components*** window, select additional drivers if needed: ***Silicon Software DLL MeIV*** for Camera Link interface or ***Camera Link HS DLL CLHS*** for Camera Link HS interface (for USB and USB3 the driver must be installed separately).
3. Choose install directory.
4. Click *"Install"* and after the next two screens installation is complete.

After a successful installation, you will find the program folder PCO Digital Camera Toolbox in the selected install directory (by default, Program Files) and a PCO software icon on your desktop.

To uninstall pco.server, use the standard software uninstallation function provided by the Windows operating system.

1.1.2 Linux

This software application supports multiple Linux distributions. Download the ***pco.server Linux*** package that is compatible with your system from our website.

Ubuntu/Debian Download the *.deb package for **pco.server**. Install the file using:

```
sudo apt-get install ./pco.server_*.*.*_*.deb
```

Rocky/Red Hat Download the *.rpm package for **pco.server**. Install the file using:

```
sudo dnf install ./pco-server-*.*.*-*.rpm
```

1.2 Supported Platforms

This section lists the currently supported operating systems, runtimes, and tools as well as planned support changes.

Operating systems

OS	Minimum supported	Latest supported	Planned
Debian LTS	11 (until 2026)	12	13 (2026)
Ubuntu LTS	20 (until 2026)	22	24 (2026)
Redhat	8 (until 2026)	9	10 (2026)
Windows	10 (until 2027)	11	Not planned

Runtimes/Tools

Name	Minimum supported	Latest supported	Planned
Python	3.8 (until 2026)	3.12	3.13 (2027)
.Net Framework	4.6 (until 2026)	4.8.1	Not planned
cMake	3.20	—	Not planned
GPU CC	5.0 (until 2026)	—	7.0 (2026)

1.3 Heartbeat Model

To ensure exclusive device usage the pco.server uses a heartbeat mechanism.

Behavior:

- The client sends `/heartbeat` periodically.
- While `/heartbeat` requests are received, the device remains reserved.
- The client releases the device using `/setFree` when finished.

If the heartbeat times out, the device is reset and becomes available.

1.4 General Description

The pco.server interface uses HTTP GET, PUT and POST requests to exchange data with the client.

Item	Description
Device type	camera, xcite
Protocol	HTTP
Port	camera: 8080, 8081, ...; xcite: 9090, 9091, ...
Base Path	/v1
Control Data	JSON
Image Data	Binary (application/octet-stream)
Error Handling	HTTP status codes + JSON error object

1.5 Status Codes

The pco.server will respond with standard HTTP status codes. The status codes can be interpreted using the following table.

Code	Meaning
200 OK	Success
400 Bad Request	Missing or invalid parameters
404 Not Found	Resource or file not found
409 Conflict	Camera already recording
500 Internal Server Error	Server failure
503 Service Unavailable	Device failure

In addition to the HTTP status codes, the server may provide additional information in the form of a numeric code and text description:

```
{
 "code": camera exception error code or -1,
 "where": "description of occurred camera exception or server ↩
    endpoint"
}
```

2 Endpoints

2.1 Server & Heartbeat Endpoints

Endpoint	Method	Description
/info	GET	Get server and device information
/heartbeat	POST	Reserve device/Keep device reserved (updates heartbeat timestamp, sets state to busy)
/setFree	POST	Manually release device

2.2 Camera Endpoints

This section describes the REST API of the Camera HTTP Server. The API provides HTTP-based access to a PCO camera, including description, configuration, recording, image acquisition, conversion, and memory management.

2.2.1 Camera Properties

Endpoint	Method	Description
/getName	GET	Camera name
/getSerial	GET	Camera serial number
/getInterface	GET	Camera interface
/getRawFormat	GET	Raw sensor data format
/isRecording	GET	Recording state (True if camera is recording, False if camera is idle)
/isColored	GET	Color sensor availability (True if camera has a color sensor)

2.2.2 Camera Descriptions

Endpoint	Method	Description
/getDescription	GET	High level camera description
/getPCODescription	GET	SDK description for more information

2.2.3 Camera Configuration Management

Endpoint	Method	Description
/defaultConfiguration	PUT	Reset configuration to default
/getConfiguration	GET	Get current camera configuration
/setConfiguration	PUT	Set a new configuration to the camera

/setConfiguration

Required parameter

Name	Type	Description
config	Configuration	Configuration that should be set

Required body

The request body must conform to the pco::Configuration structure.
See the pco.cpp manual for the complete definition.

```
Example (excerpt)

{
    "exposure_time_s": 0.01,
    "delay_time_s": 0.0,

    "roi": {
        "x0": 0,
        "y0": 0,
        "x1": 1023,
        "y1": 1023
    },
}
```

2.2.4 Exposure & Delay Control

All values are specified in seconds.

Endpoint	Method	Description
/getExposureTime	GET	Get current exposure time
/setExposureTime	PUT	Set new exposure time
/getDelayTime	GET	Get current delay time
/setDelayTime	PUT	Set new delay time

Required parameter

Name	Type	Description
exposure_s	double	Exposure time
delay_s	double	Delay time

2.2.5 Recording Control

Endpoint	Method	Description
/record	POST	Start the recording of images
/stop	POST	Stop the current recording
/getRecordedImageCount	GET	Recorded image count (actual amount)
/trigger	POST	Software trigger

/record

Optional parameter

Name	Type	Description
imgCount	int	Number of images
recMode	pco::RecordMode	Recording mode
file	std::filepath	Path or pco_vram
flags	WORD	Recorder flags

file
pco_vram writes images into a temporary VRAM directory which is cleared before recording starts. If a path is provided it must be valid on the host system's filesystem.

2.2.6 Image waiting

Endpoint	Method	Description
/waitForFirstImage	POST	Wait until the first image has been recorded
/waitForNewImage	POST	Wait until a new image has been recorded

Optional parameter

Name	Type	Description
timeout	double	Timeout in seconds
delay	bool	Delay for lower CPU usage

2.2.7 Auto Exposure

Endpoint	Method	Description
/autoExposureOn	POST	Switch auto exposure on
/autoExposureOff	POST	Switch auto exposure off
/configureAutoExposure	PUT	Set the parameters for auto-exposure calculations

/configureAutoExposure

Required body

```
{
    "region": "...",
    "min_exposure_s": 0.001,
    "max_exposure_s": 0.1
}
```

See pco.cpp manual for a complete description of the region parameter used in the auto-exposure settings.

2.2.8 Image Conversion & LUT Handling

Endpoint	Method	Description
/uploadLut	POST	Upload LUT file to host system
/loadLut	POST	Set the LUT file for the convert control settings
/getConvertControl	GET	Get current color convert settings
/setConvertControl	PUT	Set new color convert settings
/adaptWhiteBalance	POST	Do a white-balance using a transferred image

/uploadLut

A valid call to /uploadLut is required before calling `/loadLut` or using pseudocolor conversion with `/setConvertControl`, as both rely on the uploaded LUT file. Missing files return 404.

2.2.9 Hardware IO Configuration

Endpoint	Method	Description
/configureHWIO_1_exposureTrigger	PUT	Configure the exposure trigger signal input connector
/configureHWIO_2_acquireEnable	PUT	Configure the acquire enable signal input connector
/configureHWIO_3_statusBusy	PUT	Configure the Status Busy signal output connector
/configureHWIO_4_statusExpos	PUT	Configure the Status Exposure signal output connector

Signal polarity, signal type and signal timing can be set by optional parameters. See pco.cpp manual for details.

2.2.10 Image Retrieval

Image data is returned as binary payload. Metadata and timestamps are returned via HTTP headers.

Endpoint	Method	Description
/image	GET	Read a recorded image
/images	GET	Read the wanted amount of images and store them in server memory for fast access.
/getImageFromImages	GET	Use to retrieve stored images.

/image

Required parameter

Name	Type	Description
index	DWORD	Recorder image index
format	pco::DataFormat	Output data format

Optional parameter

Name	Type	Description
crop	pco::Roi	Region of interest
comp_params	PCO_Recorder_CompressionParams	Compression parameters

2.2.11 Image Averaging

Endpoint	Method	Description
/imageAverage	GET	Read an averaged image (averaged over all recorded images)

Required parameter

Name	Type	Description
format	pco::DataFormat	Output data format

Optional parameter

Name	Type	Description
crop	pco::Roi	Region of interest

2.2.12 Camera RAM (CamRAM)

Endpoint	Method	Description
/hasRam	GET	Check if camera has internal memory for recording with camram
/switchToCamRam	POST	Set the camram segment where the images should be written to/read from
/getCamRamSegment	GET	Get segment number of the active segment
/getCamRamMaxImages	GET	Get number of images that can be stored in the active segment
/getCamRamNumImages	GET	Get number of images that are available in the active segment
/getCamRamAllocation	GET	Get current allocation distribution of camram segments
/setCamRamAllocation	PUT	Set allocation distribution of camram segments.

2.3 XCite Endpoints

This section describes the REST API for controlling an X-Cite® light source. The XCite server follows the same architectural concepts as the camera server.

2.3.1 XCite Properties

Endpoint	Method	Description
/getComPort	GET	Active COM Protocol
/getType	GET	Device type

Continued on next page

Continued from previous page

Endpoint	Method	Description
/getDescription	GET	Detailed device description
/getConfiguration	GET	Current configuration

2.3.2 XCite Configuration Management

Endpoint	Method	Description
/setConfiguration	PUT	Apply configuration
/defaultConfiguration	PUT	Reset the default configuration

2.3.3 Device Control

Endpoint	Method	Description
/switchOn	POST	Turn light source on
/switchOff	POST	Turn light source off

2.3.4 Advanced Command Interface

Endpoint	Method	Description
/executeCommand	POST	Execute raw XCite command. The command list for X-Cite® is available by request. To obtain the latest update, contact Excelitas support.

Required parameter

Name	Type	Description
command	std::string	Command string

Optional parameter

Name	Type	Description
in_val	std::string	Optional input value for command

Behavior

- Sends raw command to device
- Returns device response as JSON
- Returns 400 if no command provided
- Returns 503 if device communication fails

2.3.5 Version Information

Endpoint	Method	Description
/getVersions	GET	Get firmware/software versions

Optional parameter

Name	Type	Description
library	std::string	Optional filter (specific library/component)

About Excelitas®

Excelitas is a leading provider of advanced, life-enriching technologies that make a difference, serving global market leaders in the life sciences, advanced industrial, next-generation semiconductor and avionics end markets. Headquartered in Pittsburgh, PA, USA, Excelitas is an essential partner in the design, development and manufacture of advanced technologies, offering leading-edge innovation in sensing, detection, imaging, optics and specialty illumination for customers worldwide.

Excelitas is at the forefront of addressing many of the relevant megatrends impacting the world today, including precision medicine, industrial automation, artificial intelligence and connected devices (IoT).

excelitas Enabling the future

+49 9441 2005 0
Europe

+1 866 662 6653
North America

+86 400 080 2255
Asia-Pacific

Donaupark 11
93309 Kelheim
Germany

excelitas.com
pco@excelitas.com

For a complete listing of our global offices, visit www.excelitas.com/locations