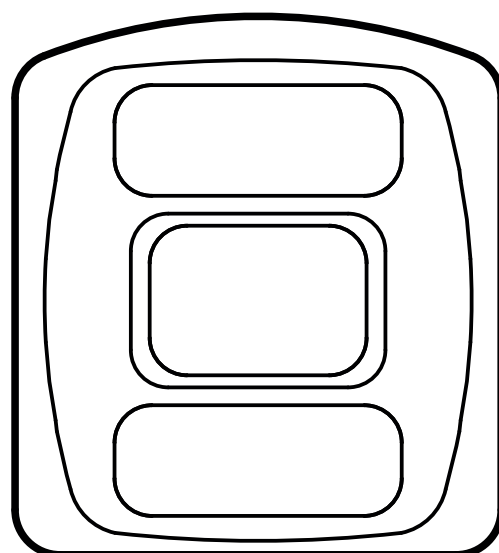




Operating instructions
3D camera

O3D301
O3D303
O3D311
O3D313

UK



Contents

1. Preliminary note	4
1.1 Symbols used	4
1.2 Warnings used	4
1.3 Open source information	5
2. Safety instructions	6
2.1 General	6
2.2 Target group	6
2.3 Electrical connection	6
2.4 Tampering with the unit	6
3. Functions and features	7
4. Items supplied	7
5. Accessories	7
6. Installation	8
6.1 Select installation location	8
6.2 Additional camera installation guidance	9
6.2.1 Typical warning limits for O3D301 / O3D303	9
6.2.2 Typical warning limits for O3D311 / O3D313	10
6.2.3 Reduce surface temperature	10
6.3 Install camera	11
6.4 Mounting accessories	11
7. Electrical connection	12
7.1 Wiring	12
7.1.1 Pin 1 / 3 (24 V / GND)	13
7.1.2 Pin 2 (trigger input)	13
7.1.3 Pin 4 / 5 (ready / cascading)	13
7.2 Wiring examples	14
7.2.1 Trigger image capture with proximity sensor	14
7.2.2 Install several cameras next to each other	15
8. Indicators	16
9. Set-up	17
9.1 Set parameters of the device	17
9.2 Detect object	17
10. Programming example	18
10.1 ifm3Dlib	18
11. Maintenance, repair and disposal	19
11.1 Clean	19
11.2 Update firmware	19
11.3 Replace camera	19
12. Approvals/standards	19
13. Scale drawings	20
13.1 O3D303 / O3D313	20
13.2 O3D301 / O3D311	20
14. Appendix	21
14.1 Required Ports	21
14.2 XML-RPC Interface	21
14.2.1 Sample XML-RPC command	21
14.2.2 XML-RPC Objects	22
14.3 Process Interface	25
14.3.1 Sending Commands	25
14.3.2 Receiving Images	26
14.3.3 Image data	26
14.3.4 Additional Information for CONFIDENCE_IMAGE	30

14.3.5	Configuration of PCIC Output	31
14.4	Process Interface Command Reference	36
14.4.1	a Command (activate application)	36
14.4.2	A? Command (occupancy of application list)	36
14.4.3	c Command (upload PCIC output configuration)	37
14.4.4	C? Command (retrieve current PCIC configuration)	37
14.4.5	E? Command (request current error state)	37
14.4.6	G? Command (request device information)	38
14.4.7	H? Command (return a list of available commands)	39
14.4.8	I? Command (request last image taken)	40
14.4.9	o Command (set logic state of a ID)	40
14.4.10	O? Command (request state of a ID)	41
14.4.11	p Command (turn PCIC output on or off)	41
14.4.12	S? Command (request current decoding statistics)	42
14.4.13	t Command (execute asynchronous trigger)	42
14.4.14	T? Command (execute synchronous trigger)	43
14.4.15	v Command (set current protocol version)	43
14.4.16	V? Command (request current protocol version)	43
14.5	Error codes	44
14.6	XML-RPC Command Reference	45
14.6.1	Parameter API	45
14.6.2	Main Object	46
14.6.3	Session Object	49
14.6.4	Edit Mode Object	51
14.6.5	Device Config Object	52
14.6.6	Device/Network Config Object	56
14.6.7	Application Config Object	56
14.6.8	Application/Imager Config Object	58
14.6.9	Image Settings and Filter Parameters	65



Copyright

Microsoft®, Windows®, Windows Vista®, Windows 7®, Windows 8®, Windows 8.1® and Windows 10® are registered trademarks of Microsoft Corporation.

Adobe® and Acrobat® are registered trademarks of Adobe Systems Inc.



All trademarks and company names used are subject to the copyright of the respective companies.

1. Preliminary note

This document is intended for specialists. These specialists are people who are qualified by their appropriate training and their experience to see risks and to avoid possible hazards that may be caused during operation or maintenance of the device. The document contains information about the correct handling of the device.

Read this document before use to familiarise yourself with operating conditions, installation and operation. Keep this document during the entire duration of use of the device.

1.1 Symbols used

- ▶ Instructions
- > Reaction, result
- [...] Designation of keys, buttons or indications
- Cross-reference
-  Important note
Non-compliance may result in malfunction or interference.
-  Information
Supplementary note

1.2 Warnings used

NOTE

Warning of damage to property.

1.3 Open source information

This product can contain Free Software or Open Source Software from various software developers which is subject to the following licenses: General Public License version 1, version 2 and version 3 (General Public License version 3 in conjunction with the GNU Compiler Collection Runtime Library Exception version 3.1), Lesser General Public License version 2.1, Lesser General Public License version 3, Berkeley Software Distribution (BSD-2-Clause, BSD-3-Clause, BSD-4-Clause), MIT-License (MIT), Python Software Foundation License 2.0, Pearl Artistic License and Artistic License 2.0, Microsoft Public License, Apache Software License Version 1.0, 1.1 und 2.0, ISC License, libpng License, zlib License, the Academic Free License version 2.1, Mozilla Public License 2.0.

For the components subject to the General Public License in their respective versions the following applies:

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation. If version 1 applies to the software: either version 1 of the License or (at your option) any later version; if version 2 (or 2.1) applies to the software: either version 2 (or 2.1) of the License or (at your option) any later version; if version 3 applies to the software: either version 3 of the License or (at your option) any later version. The following disclaimer of the software developers applies to the software components that are subject to the General Public License or the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License and the GNU Lesser General Public License for more details.

The responsibility of ifm electronic gmbh for ifm products, in the case of product-specific software, remains unaffected by the above disclaimer. Please note that the firmware for the ifm products is in some cases provided free of charge. The price of the ifm products has then to be paid for the respective device itself (hardware) and not for the firmware. For the latest information on the license agreement for your product please visit www.ifm.com

For binaries that are licensed under any version of the GNU General Public License (GPL) or the GNU LGPL or Mozilla Public License 2.0 you may obtain the complete corresponding source code of the GPL software from us by sending a written request to: opensource@ifm.com or to ifm electronic gmbh, Friedrichstraße 1, 45128 Essen, Germany.

We charge €30 for each request. Please write "source for product Y" in the memo line of your payment. Your request should include (i) the name of the covered binary, (ii) the name and the version number of the ifm product, (iii) your name and (iv) your return address.

The following applies to components covered by the General Public License in their respective versions. This offer is valid to anyone in receipt of this information. This offer is valid for at least three years (from the date you received the GPL/LGPL covered code).

2. Safety instructions

2.1 General

These instructions are an integral part of the device. They contain texts and figures concerning the correct handling of the device and must be read before installation or use.

Observe the operating instructions. Non-observance of the instructions, operation which is not in accordance with use as prescribed below, wrong installation or incorrect handling can seriously affect the safety of operators and machinery.

2.2 Target group

These instructions are intended for authorised persons according to the EMC and low-voltage directives. The device must be installed, connected and put into operation by a qualified electrician.

2.3 Electrical connection

Disconnect the unit externally before handling it.

The connection pins may only be supplied with the signals indicated in the technical data and on the device label and only the approved accessories of ifm may be connected.

2.4 Tampering with the unit

In case of malfunctions or uncertainties please contact the manufacturer. Any tampering with the device can seriously affect the safety of operators and machinery. This is not permitted and leads to the exclusion of any liability and warranty claims.

3. Functions and features

The O3D3xx 3D camera is an optical camera which measures the distance between the camera and the nearest surface point by point using the time-of-flight principle. The O3D3xx 3D camera illuminates the scene with an internal infrared light source and calculates the distance by means of the light reflected from the surface.

The O3D3xx 3D camera supplies data which describes the captured scene three-dimensionally. This distance data can be output via Ethernet and evaluated by the user. Parameter setting of the O3D3xx 3D camera is also done via Ethernet.

The O3D3xx 3D camera may only be used under the operating conditions specified in the data sheet.

The device safety is rated for use under the following operating conditions:

- Indoor use
- Altitudes up to 2000 m
- Relative air humidity up to max. 90 %, non condensing
- Pollution degree 3

Because of the requirements for electromagnetic interference emissions, the device is intended for use in industrial environments. The device is not designed for use in domestic areas.



The device may only be used under the operating conditions specified in the data sheet.

4. Items supplied

- O3D3xx 3D camera
- Brief instructions



The data sheet and other documentation (software manual, etc.) are available on our website:

www.ifm.com

5. Accessories

The following accessories are needed for the operation of the camera:

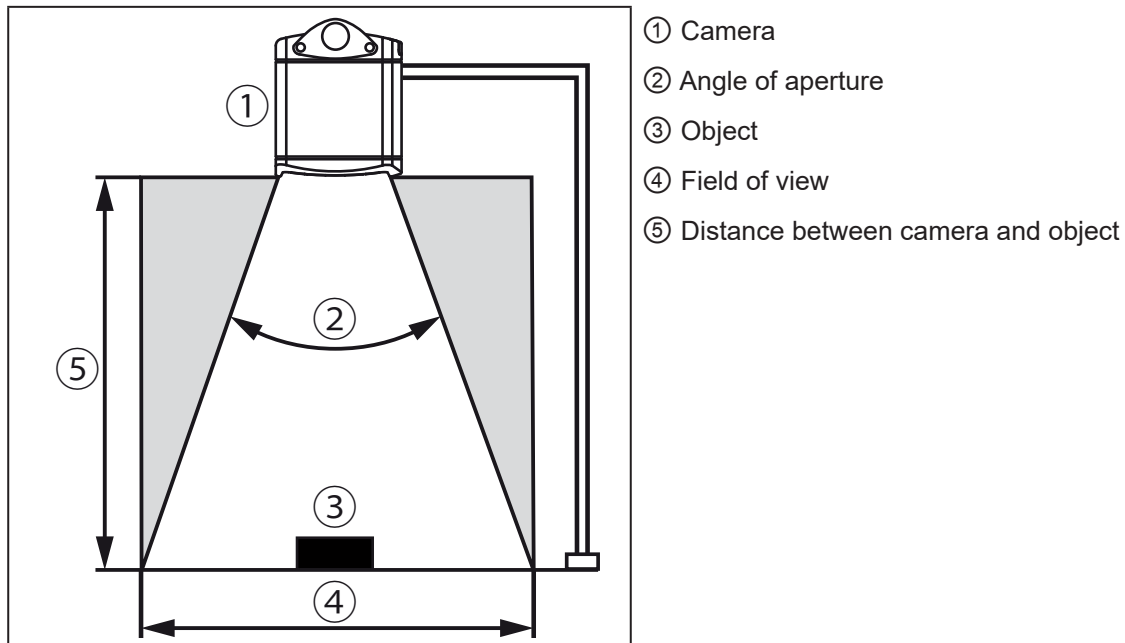
Article number	Description
E11950	Power supply cable for camera/sensor
E11898	M12 industrial Ethernet connection cable



The ifm Vision Assistant software is available free of charge on our website: www.ifm.com

6. Installation

The chapter describes what has to be observed before installation and how to install the camera.



6.1 Select installation location

Observe the following instructions for the selection of the installation location:

- ▶ The object ③ must be completely in the field of view ④.
- > The size of the field of view depends on the camera type and is indicated in the data sheet. The size of the field of view also depends on the distance of the camera to the object ⑤: With increasing distance the field of view becomes larger.
- ▶ Take tolerances into account when positioning the object.
- ▶ When determining the distance between camera and object ⑤ take the measuring range of the camera into account.
- > The measuring range is indicated in the data sheet of the camera.
- ▶ Select a distance as small as possible between camera and object ⑤.
- > If the distance is as small as possible, the object is detected with the maximum resolution.
- ▶ Avoid any strong ambient light and sunlight at the installation location.
- > An extraneous light level of over 8 klx (with solar spectrum) causes measurement errors. In fact, only the infrared component between 800 and 900 nm is of concern.
- ▶ Avoid installation in heavily polluted environments.
- > In heavily polluted environments the camera lens will get dirty despite downwards orientation ①.
- ▶ Avoid transparent panes between the camera ① and the object ③.
- > Transparent panes reflect part of the light even if a very clean glass pane is used.



If the instructions are not observed, measurement errors may occur.

6.2 Additional camera installation guidance

The surface temperature of the camera depends on the operating mode, the parameter selection and the thermal exposure of the camera to the environment.



Make sure that the camera complies with the following requirement:

The surface temperature for easily accessible surfaces may be max. 25 °C higher than the ambient temperature (to IEC 61010-2-201).

The following diagrams contain typical warning limits as a reference for the installer.



The diagrams are valid for the following operating modes:

- Low [1 exposure]
- Moderate [2 exposures]
- High [3 exposures]

In the event of moderate and high exposures the typical warning limits must be determined via the sum of the exposure times. The exposure times are indicated in the software ifm Vision Assistant.

Follow one of the instructions if the warning limits are exceeded:

- ▶ Reduce surface temperature (→ 6.2.3).
- ▶ Mount the camera in a location or housing that provides protection from the heat source but maintains air circulation around the camera.
- > An increase in the surface temperature of the camera should be prevented.

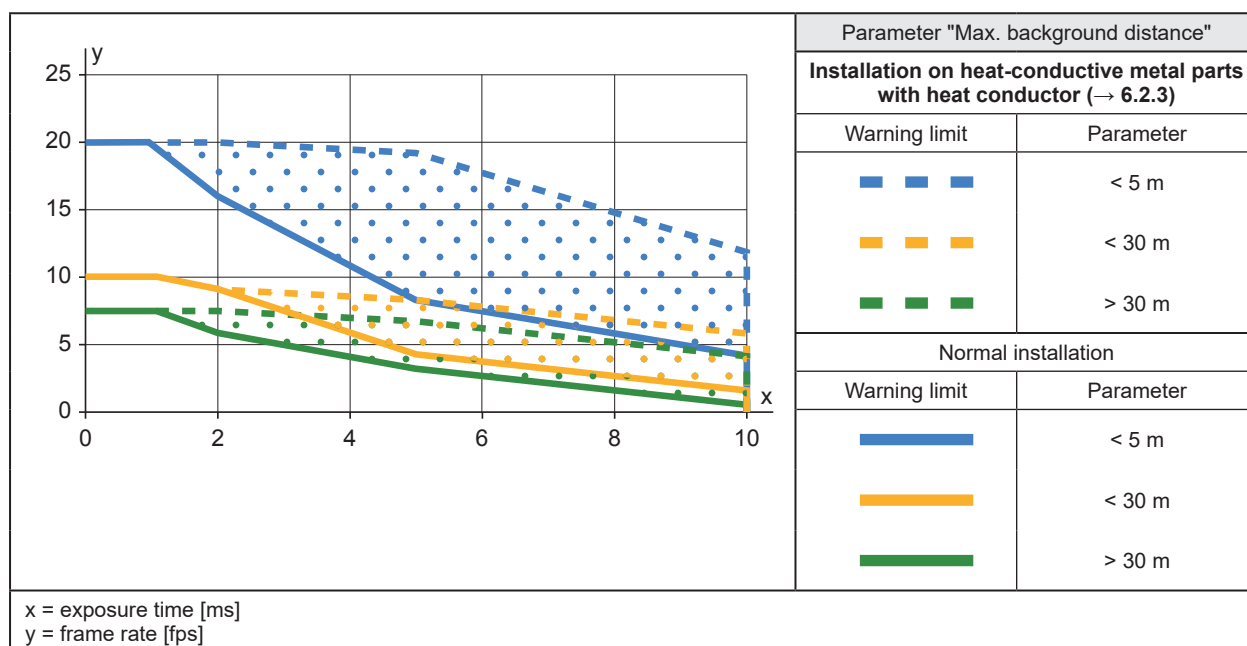


The parameter "Max. background distance" is set in the ifm Vision Assistant. In the diagrams the warning limits of the parameter are shown with dashed and continuous lines.

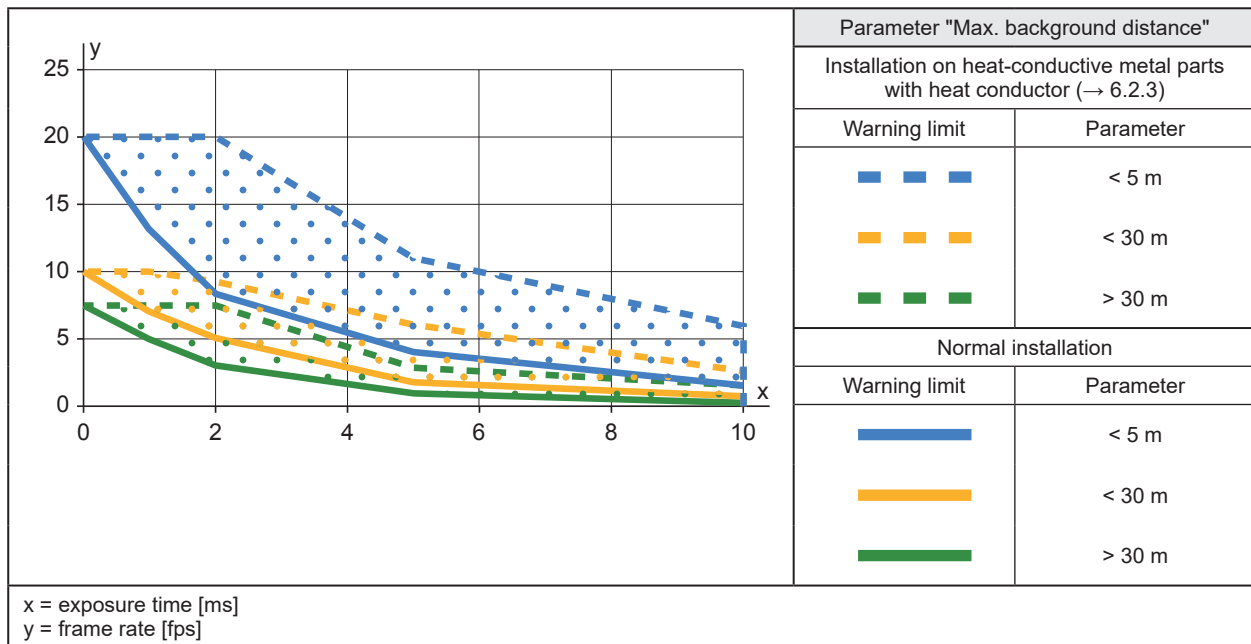
If the camera is in one of the dotted areas, the surface temperature must be reduced (→ 6.2.3).
If the warning limit is exceeded despite a heat-dissipating installation, it is possible to additionally mount the contact protection.

If you stay below the typical warning limits in case of normal installation, no measures need to be taken.

6.2.1 Typical warning limits for O3D301 / O3D303



6.2.2 Typical warning limits for O3D311 / O3D313



6.2.3 Reduce surface temperature

With the following measures the surface temperature can be reduced:

- ▶ Mount the camera on heat-conductive metal parts.
 - > A large-surface contact of the camera with metal parts increases heat dissipation (e.g. aluminium).
- ▶ Use a heat conductor when mounting the camera on metal parts.
 - > The heat-conductive effect is increased by means of the heat conductor. The heat conductor is available as accessories (→ 6.4).
- ▶ Reduce obstructions around the camera. Reduce the density of objects mounted near to the camera.
 - > Obstructions around the camera and a high installation density may have a negative impact on convection (air movement).
- ▶ Mount one or two heat sinks on the camera.
 - > The heat sinks increase the surface of the camera, reducing the surface temperature. The heat sinks are available as accessories (→ 6.4).
- ▶ Reduce exposure time, frame rate or max. background distance.
 - > The operating mode used and the parameters can increase the surface temperature.

6.3 Install camera

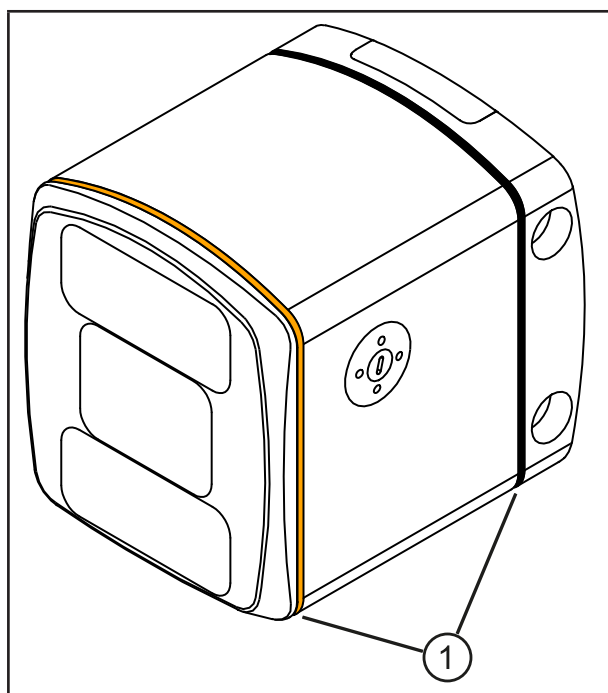
Observe the following instructions when installing the camera:

- ▶ Mount the camera using 2x M5 screws or mounting set.
- > The bore dimensions for the M5 screws are indicated in the data sheet.
- > The mounting set is available as accessories (→ 6.4).
- ▶ Use strain reliefs for all cables connected to the camera.
- ▶ Do not apply force to the seals of the device (see next figure).
- > If the seals are subjected to external mechanical force, the IP protection class specified in the data sheet cannot be guaranteed. Prevent force from acting on the seals with the following measures: provide for a recess in the fixture (for example a groove) or use spacers.

Observe the following instructions when installing an O3D301 and O3D311:

- ▶ Mount the camera so that the focus adjustment screw can be accessed with a screw driver.
- > The position of the focus adjustment screw is indicated in the scale drawing (→ 13).

! If the device is permanently used in wet areas, the nut of the M12 Industrial Ethernet cable (e.g. E11898) may corrode. Use a cable with a high-grade stainless steel nut for permanent use in wet areas.



①: Seals of the device

6.4 Mounting accessories

Depending on the location and type of installation, you can use the following mounting accessories:

Article number	Description
E3D301	Smart Camera mounting set
E3D302	Smart Camera cooling element
E3D303	Smart Camera heat conductor
E3D304	2x Smart Camera cooling element

i You can find more information about the accessories at: www.ifm.com

7. Electrical connection

Observe the following instructions before electrical installation.

NOTE

The camera must be connected by a qualified electrician. Observe the electrical data in the data sheet.

Camera of protection class III (PC III).

The electrical supply must only be made via PELV circuits.

Electric supply must comply with UL61010-1, chap. 9.4 - Limited Energy:

The overcurrent protection device must switch off a current of 6.6 A in 120 s. For the correct rating of the overcurrent protection device take the technical data of the camera and wiring into account.

The separation of external circuits must comply with UL61010-2-201, fig. 102.

For cable lengths > 30 m use an additional protection against surge voltages to IEC 6100-4-5.

Disconnect power before connecting the camera.



For the scope of validity cULus:

Minimum temperature rating of the cable to be connected to the field wiring terminals: 70 °C.

7.1 Wiring

	① Ethernet	
	M12 socket, D-coded, 4 poles	
		1 TD + 2 RD + 3 TD - 4 RD - S Shield
	② Power supply	
M12 connector, A-coded, 5 poles		
		1 U+ 2 Trigger input 3 GND 4 Switching output 1 - ready 5 Switching output 2 - cascading



Cover unused Ethernet connection with the protective cap (E73004).

Tightening torque 0.6...0.8 Nm.



The behaviour of the switching inputs and outputs can be set with the software ifm Vision Assistant. The setting PNP or NPN always applies to all switching inputs and outputs.

When installing actuators and sensors make sure that the setting is correct (e.g. photoelectric sensors for triggering).

The switching outputs can also be operated as pulse outputs which reset their switching signal after an adjustable time.

7.1.1 Pin 1 / 3 (24 V / GND)

The permissible voltage range is indicated in the data sheet of the camera.

7.1.2 Pin 2 (trigger input)

The image capture of the camera can be triggered with a switching signal via the trigger input.

The following trigger edges can be used:

- Falling edge triggers image capture
- Rising edge triggers image capture
- Falling and rising edges trigger image capture



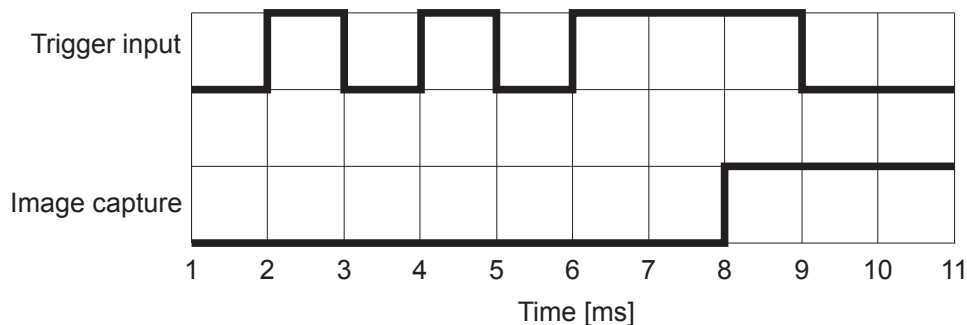
Further possibilities to trigger the camera:

- Process interface command (→ 14.3)
- Continuous image capture with fixed frame rate



The trigger input is internally debounced. Depending on the electrical installation debouncing of the trigger wire is not necessary.

Internal debouncing prevents several short pulses from triggering. The pulse must be at least 2 ms long to be recognised as a trigger.



7.1.3 Pin 4 / 5 (ready / cascading)

The electrical data of the switching outputs 1 and 2 (ready / cascading) is indicated in the data sheet.

The switching outputs provide the following camera status set as default:

- Switching output 1: "Ready for trigger"
- Switching output 2: "Image capture finished"



"Switching output switched" means that the respective camera status has occurred.

Depending on the setting the camera status can have one of the following values:

- "Ready for trigger"

The camera signals that a new image can be captured. Only with this camera status trigger operations are processed. For the continuous image capture the status "Ready for trigger" is not output.
- "Image capture finished"

The camera signals that the image capture is finished. The camera status can be used for cascading cameras.
- "Evaluation finished"

The camera signals that image processing is finished. At that moment the switching outputs are already updated. The image data is transmitted via Ethernet.
- "Error"

The camera signals an internal error. Detailed information about errors can be requested via Ethernet.

7.2 Wiring examples

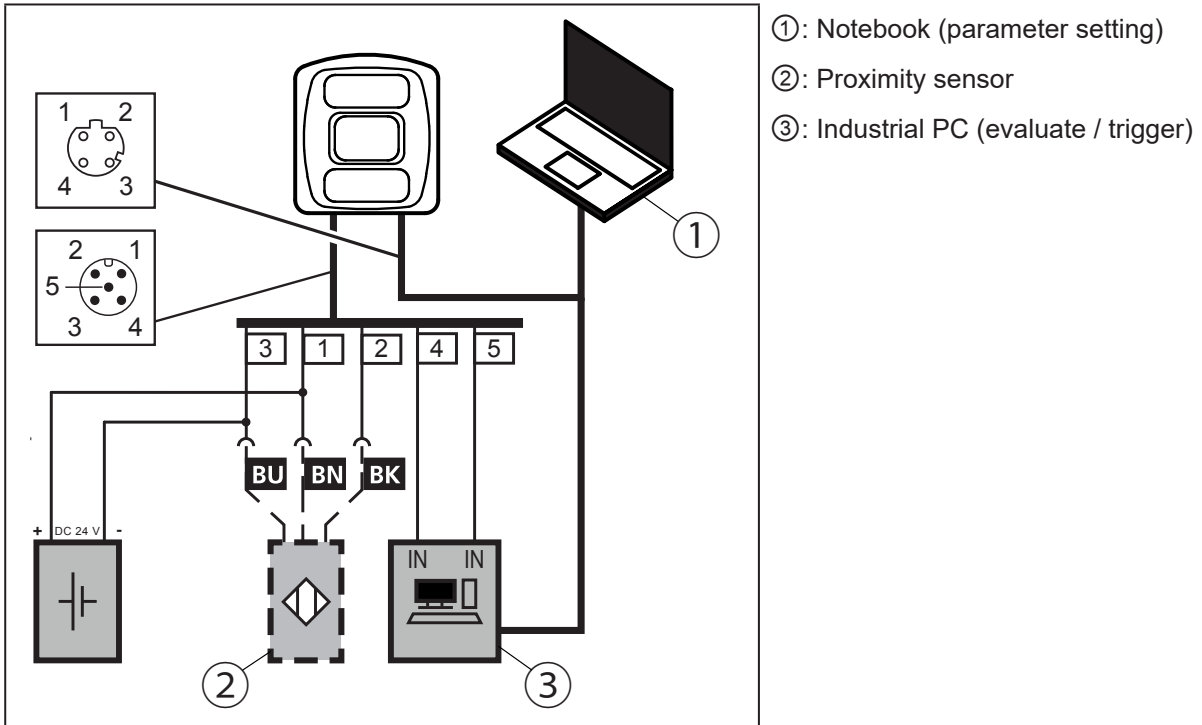
Wiring examples of the camera are given below.

7.2.1 Trigger image capture with proximity sensor

The camera can be triggered externally:

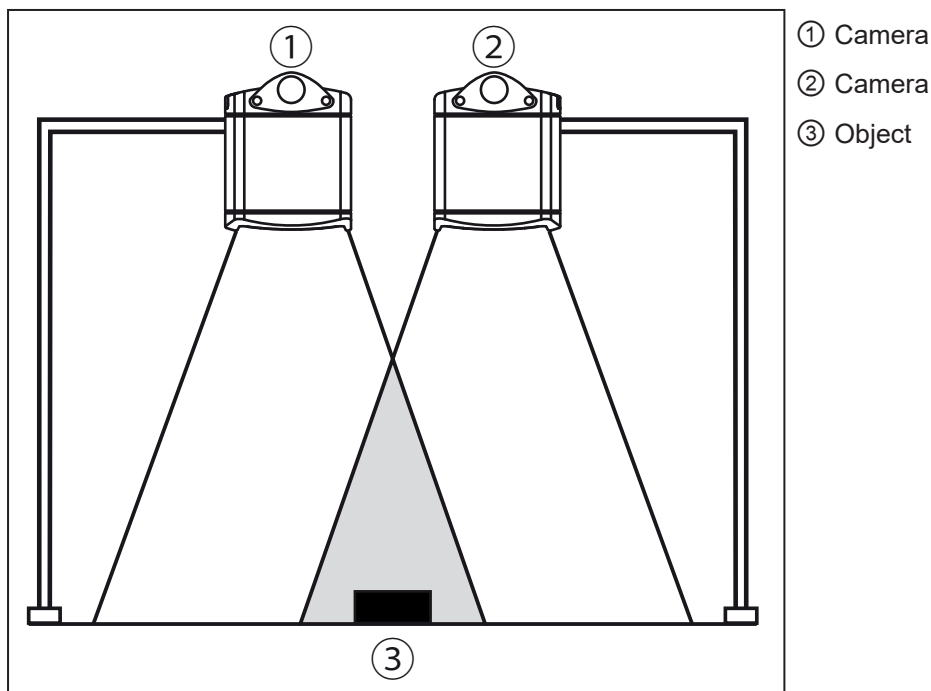
- via Ethernet
- via a proximity sensor connected to the trigger input

The following illustration shows the wiring of the camera with a proximity sensor.



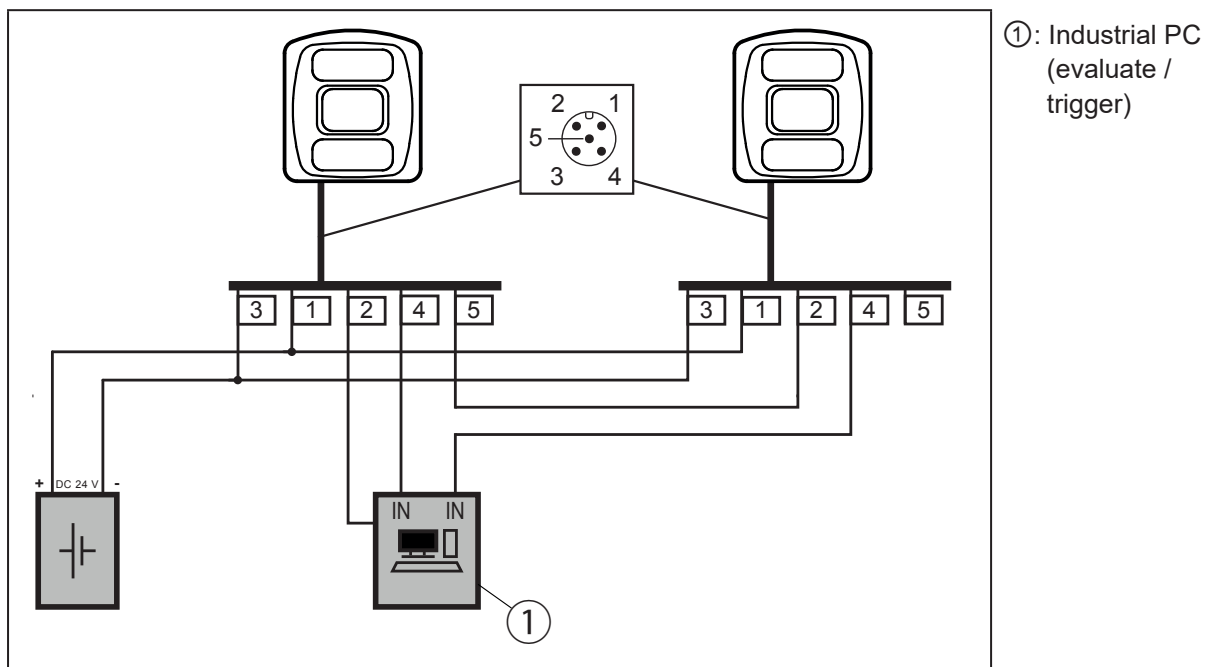
7.2.2 Install several cameras next to each other

Cameras installed next to each other can cause measurement errors due to simultaneous exposure.



The measurement errors can be avoided in two ways:

- Cascade cameras via HW trigger
During cascading a controller triggers the image capture of the 1st camera. After completion of the image capture the 1st camera automatically triggers the 2nd camera. The 2nd camera signals the end of the sequence to the controller.



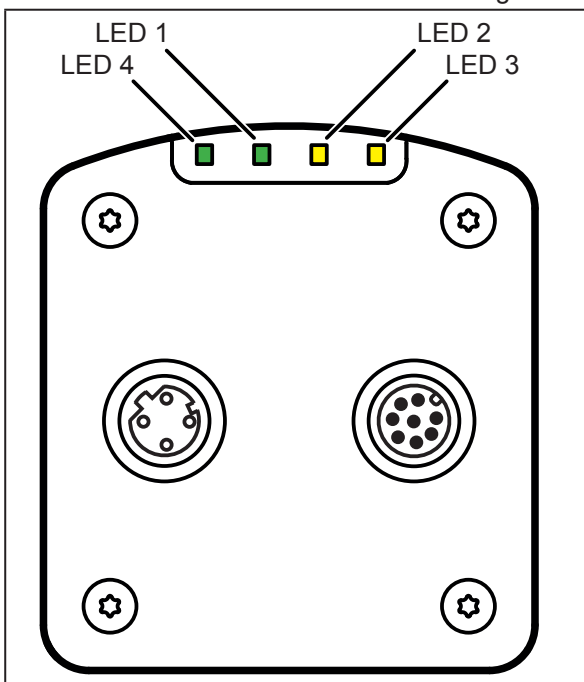
- Use different frequency channels
With the software ifm Vision Assistant each camera can be assigned its own frequency channel. The different frequency channels reduce the occurrence of measurement errors.

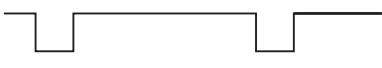
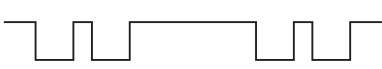


The ifm Vision Assistant software is available free of charge on our website: www.ifm.com

8. Indicators

Via the LED indicators 1 - 4 the camera signals the current operating state.



LED 4 (Ethernet)	LED 1 (Power)	LED 2 (Out 1)	LED 3 (Out 2)	Description
	On			Camera is ready for operation, supply voltage applied
	Flashes at 0.5 Hz			No parameters set or parameter setting was not loaded into the camera On  Off
	Flashes 2x at 0.5 Hz			Camera is in the parameter setting mode On  Off
		On		Switching output 1 switched
		Flashes at 8 Hz		Switching output 1 shorted
			On	Switching output 2 switched
			Flashes at 8 Hz	Switching output 2 shorted
On				Ethernet connected
Flashes				Ethernet transmitting data
Off				Ethernet not connected
		Flashes at 8 Hz	Flashes at 8 Hz	Camera signals internal error
		Flashes at 2 Hz	Flashes at 2 Hz	Camera signals correctable error. The error information can be read via Ethernet
		Running light ⇒		Camera booting
		Running light ←		Camera carrying out firmware update

9. Set-up

After power on the camera is put into operation. After 15 seconds the camera is in the evaluation mode where saved applications are executed. The indicators signal the current operating state (→ 8).



Up to 32 applications can be saved on the camera. Typically, an application contains the following parameters:

- Image capture: e.g. triggering of the image capture, exposure time, image processing filter
- Interface: Ethernet, switching outputs

The respective application can be activated with the software ifm Vision Assistant or via process interface commands.

UK

9.1 Set parameters of the device

The device parameters can be set in two ways:

- Software ifm Vision Assistant (→ see software manual)
- ifm3Dlib (third-party product → <https://github.com/ifm/ifm3d>)
Programming example for ifm3Dlib: (→ 10)
- ROS (third-party product → <https://github.com/ifm/ifm3d-ros>)
- XML-RPC command (→ 14.6)



The software ifm Vision Assistant and detailed information about the measuring principle of the device and it's effect are described in the software manual.

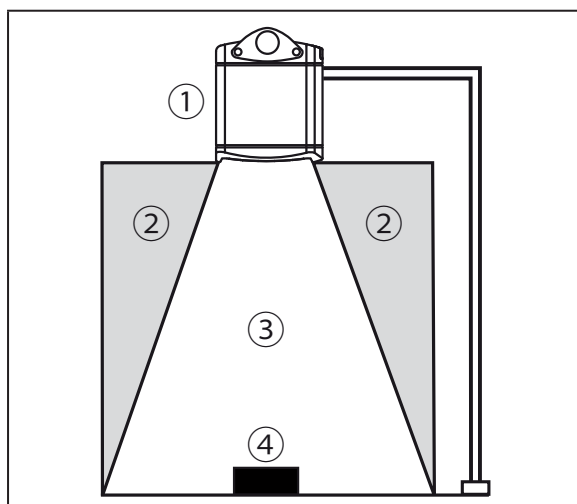
The software manual is available on our website: www.ifm.com



The library ifm3Dlib and the wrapper ROS are programmed by ifm electronic. Both packages are available for Linux under Apache License Version 2.0.

9.2 Detect object

The conditions which lead to a high detection rate of objects are described below.



- ① Camera
- ② Zone of influence
- ③ Field of view
- ④ Object

Optimum detection of an object ④ is given if the following conditions are met:

- Object is positioned in the field of view ③
- Object is the nearest visible object to the camera ①
- Zone of influence ② is clear from objects (obstructions etc.)
- Lens window of the camera is free from soiling.



If the conditions are not met, measurement errors may occur.

10. Programming example



Preferably use the ifm3Dlib for access to the device under Linux. The library has been tested and is the reference implementation for C++.

The library is supported by ifm electronic and the company Lovepark Robotics. The Apache-2 licence does not allow commercial use.

10.1 ifm3Dlib

Below a short C++ example how to address the device with ifm3Dlib.

```
auto cam = ifm3d::Camera::MakeShared();
auto fg = std::make_shared<ifm3d::FrameGrabber>
    (cam, (ifm3d::IMG_AMP|ifm3d::IMG_RDIS|ifm3d::IMG_CART));
auto img = std::make_shared<ifm3d::ImageBuffer>();
if (! fg->WaitForFrame(img.get(), 1000))
{
    std::cerr << "Timeout waiting for camera!" << std::endl;
    return -1;
}
pcl::io::savePCDFFileASCII("point_cloud.pcd", *(img->Cloud()));
imwrite("amplitude.png", img->AmplitudeImage());
imwrite("radial_distance.png", img->DistanceImage());
```

In the example the device transmits the data set. The amplitude image and the radial distance from the data set is saved as PNG file. The Cartesian coordinates are saved as PCL file.

11. Maintenance, repair and disposal

Observe the following instructions:

- ▶ Do not open the device as it does not contain any components which can be maintained by the user. The device must only be repaired by the manufacturer.
- ▶ Dispose of the camera in accordance with the national environmental regulations.

11.1 Clean

Observe the following instructions before cleaning the camera:

- ▶ Use clean and lint-free cloth.
- ▶ Use glass cleaner as cleaning agent.



If the instructions are not observed, scratches on the lens window may cause measurement errors.

11.2 Update firmware

With the software ifm Vision Assistant the firmware of the camera can be updated.



Parameters saved in the camera get lost by the firmware update. Create a backup copy of the parameters before updating the firmware:

- ▶ Before updating the firmware export parameters.
- ▶ After updating the firmware import parameters.



Firmware updates are available on our website: www.ifm.com

11.3 Replace camera

The parameters are lost when a device is replaced. Create a backup copy of the parameters before replacing the device:

- ▶ Export the parameters of the old device before replacement.
- ▶ Import the parameters into the new device after replacement.



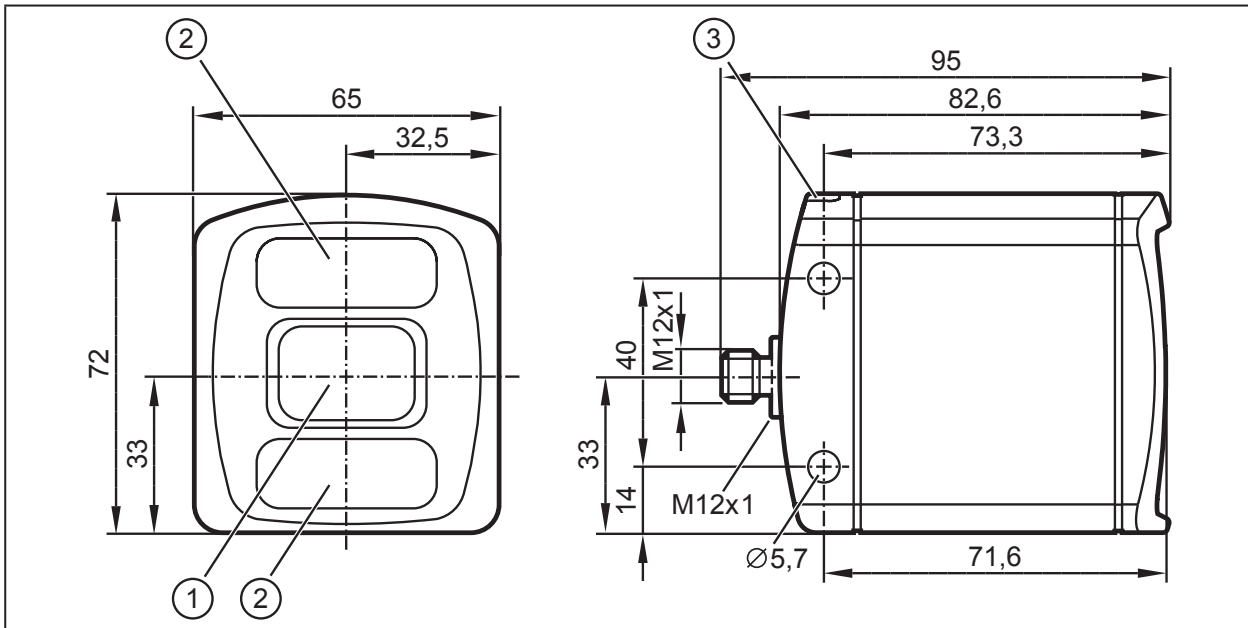
With the export and import of parameters several cameras can be quickly provided with the same parameters.

12. Approvals/standards

The EU declaration of conformity is available at: www.ifm.com

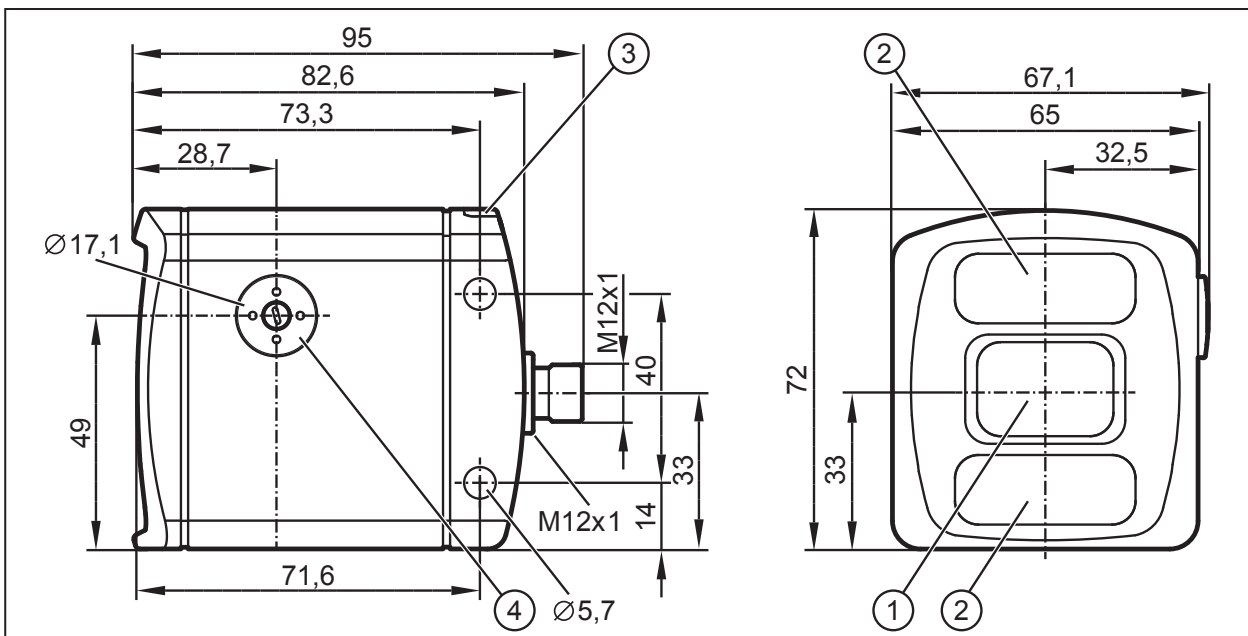
13. Scale drawings

13.1 O3D303 / O3D313



- ①: Lens
- ②: Illumination unit
- ③: LED 2 colours (yellow/green)

13.2 O3D301 / O3D311



- ①: Lens
- ②: Illumination unit
- ③: LED 2 colours (yellow/green)
- ④: Focal setter

14. Appendix

14.1 Required Ports

The following ports are required for the camera configuration using XML-RPC and for receiving data on the process interface. They must not be blocked by a firewall or router.

- TCP/HTTP: 80
- TCP: 50010

If the ifm Vision Assistant is used, the following additional ports must also be available:

- UDP: 3321
- TCP/HTTP: 8080

It is possible to configure another port than 50010 for the process interface. If a different port is used, it must not be blocked either.

14.2 XML-RPC Interface

In case the O3D3xx camera should not be configured by the “ifmVisionAssistant”, the XML-RPC interface can be used instead.



General information about XML-RPC is found on the website <http://xmlrpc.scripting.com/spec>

To send a command via the XML-RPC interface the command is in a special layout. In this command, linefeeds and carriage returns are essential.



Every command which is sent via the XML-RPC interface must end with carriage return <CR> and linefeed <LF>.

Several commands will use different URLs in the XML-RPC header.

14.2.1 Sample XML-RPC command

All following XML-RPC commands will have this type of layout:

```
POST /RPC3 HTTP/1.0<CR><LF>
User-Agent: Frontier/5.1.2 (WinNT)<CR><LF>
Host: betty.userland.com<CR><LF>
Content-Type: text/xml<CR><LF>
Content-length: 181<CR><LF>
<CR><LF>
<?xml version="1.0"?><CR><LF>
<methodCall><CR><LF>
<methodName>examples.getStateName</methodName><CR><LF>
<params><CR><LF>
<param><CR><LF>
<value><i4>41</i4></value><CR><LF>
</param><CR><LF>
</params><CR><LF>
</methodCall><CR><LF>
```

The following example contains one O3D3xx command:

```
POST /api/rpc/v1/com.ifm.efector/ HTTP/1.1 <CR><LF>
User-Agent: Frontier/5.1.2 (WinNT)<CR><LF>
Host: 192.168.0.69<CR><LF>
Content-Type: text/xml<CR><LF>
Content-length: 94<CR><LF>
<CR><LF>
<?xml version="1.0"?><CR><LF>
<methodCall><CR><LF>
<methodName>getParameter</methodName><CR><LF>
</methodCall><CR><LF>
```

14.2.2 XML-RPC Objects

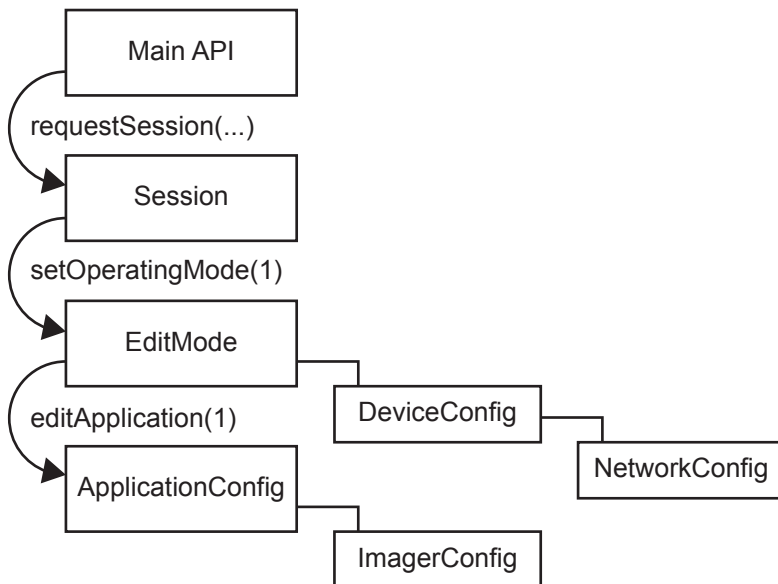
To communicate and to configure the device via XML-RPC the XML-RPC commands have to use different XML-RPC objects. Different commands need different XML-RPC objects (see XML-RPC command references).

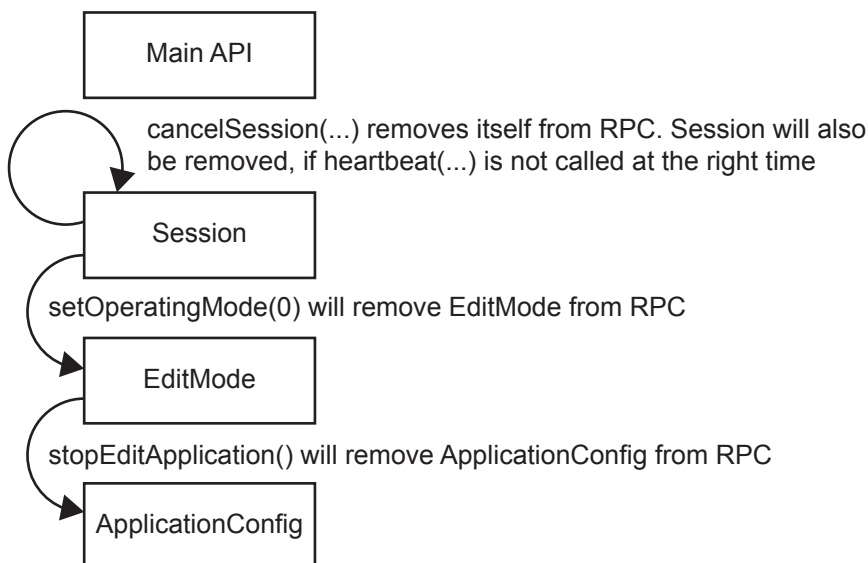
The interface of O3D3xx is structured in an object-oriented way. Some of the objects are available all the time, others are only available after bringing the device into a special mode by calling a method on an already available object. This mechanism is used to create system requirements (e.g. password protection).



It could be necessary to send heartbeats so that there will be no session timeout.

The following diagram should give an overview how objects are related to each other and which methods must be called to make others available:





UK

Main Object

Object-URI: /api/rpc/v1/com.ifm.efector/

This is the main object of RPC. It contains methods to open a session. The session contains methods for activating the edit mode. Most of its methods are only getters, because it should be possible to protect editing with a password.

Session Object

Object URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/

The URL part "d21c80db5bc1069932fbb9a3bd841d0b" is the session ID. It is returned by the command "requestSession" of the main object. If the command "requestSession" is called without a user-defined session ID, which can be passed as a parameter, a random session ID is generated automatically.

EditMode Object

Object URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/

This object is only available if the device is in the edit operating mode. The index of applications must be between 1 and 32. The device must only support 32 applications and the indexes must start at 1.

DeviceConfig Object

Object-URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/device/

Device/NetworkConfig Object

Object URI e.g.:

/api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/device/network/

Application Config Object (editable application)

Object URI e.g.:

/api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/application/

Application/Imager Config Object (O3D3xx)

Object URI e.g.:

/api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/application/imager_001/

As there is only one imager config on O3D3xx, the ID must be fixed to "001". Data of this object is persistently saved when calling "save" on the application config object. The imager config RPC object has multiple sub-types. Only parameters relevant for a specific type are available while it is active. They are based on frequency (extending the distance) and integration intervals (extending the measurement details).

Type names, based on GUI draft (under 5 metres -> single frequency, up to 30 metres -> double frequency, more than 30 metres -> triple frequency.):

under5m_low

under5m_moderate

under5m_high

upto30m_low

upto30m_moderate

upto30m_high

morethan30m_low

morethan30m_moderate

Image Settings and Filter Parameters

There is an RPC object for spatial filter parameters in each imager configuration.

Object URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/application/imager_001/spatialfilter

There is an RPC object for temporal filter parameters in each imager configuration.

Object URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/application/imager_001/temporalfilter

Data of these objects is persistently saved when calling "save" on application config object.

14.3 Process Interface

The process interface is used during the normal operation mode to get operational data (e.g. 3D images, process values) from the O3D3xx.

14.3.1 Sending Commands

For sending commands via the process interface the commands have to be sent with a special protocol and as ASCII character strings. This protocol conforms to the version 3 of the O2V/O2D products.

Structure of the protocol:

<Ticket><length>CR LF <Ticket><content>CR LF

Abbreviation	Description	ASCII code (dec)	ASCII code (hex)
CR	Carriage Return	13	D
LF	Linefeed	10	A
< >	Marking of a placeholder (e.g. <code> is a placeholder for code)		
[]	Optional argument (possible but not required)		

Command	Description
<content>	It is the command to the device (e.g. trigger the unit).
<ticket>	It is a character string of 4 digits between 0-9. If a message with a specific ticket is sent to the device, it will reply with the same ticket. A ticket number must be > 0999. Use a ticket number from the range 1000 - 9999.
<length>	It is a character string beginning with the letter 'L' followed by 9 digits. It indicates the length of the following data (<ticket><content>CR LF) in bytes.

They are different protocol versions available:

Version	Input format	Output format
V1	<Content>CR LF	as input
V2	<Ticket><Content>CR LF	as input
V3	<Ticket><Length>CR LF<Ticket><Content>CR LF	as input
V4	<Content>CR LF	<length>CR LF<Content>CR LF



The default protocol version is "V3". It is recommended to use protocol version 3 for machine to machine communication. This is due to the fact that only version 3 supports asynchronous messages and provides length information.

Ticket numbers for asynchronous messages:

Ticket number	Description
0000	Asynchronous results
0001	Asynchronous error messages / codes
0010	Asynchronous notifications / message codes

14.3.2 Receiving Images

For receiving the image data a TCP/IP socket communication is established. The default port number is 50010. The port number may differ based on the configuration. After opening the socket communication, the O3D3XX device will automatically (if the device is in free run mode) send the data through this socket to the TCP/IP client (PC).

PCIC output per frame. The following data is submitted in this sequence:

Component	Content
Ticket and length information	(→ 14.4.14)
Ticket	„0000“
Start sequence	String "star" (4 bytes)
Normalised amplitude image Output format: 16-bit unsigned integer	1 image
Distance image Output format: 16-bit integer. Unit: mm.	1 image
X image Output format: 16-bit signed integer. Unit: mm.	1 image
Y image Output format: 16-bit signed integer. Unit: mm.	1 image
Z image Output format: 16-bit signed integer. Unit: mm.	1 image
Confidence image Output format: 8-bit unsigned integer	1 image
Diagnostic data	
Stop sequence	String "stop" (4 bytes)
Ticket signature	<CR><LF>

14.3.3 Image data

For every image there will be a separate chunk. The chunk is part of the response frame data of the process interface.

The header of each chunk contains different kinds of information. This information is separated into bytes. The information contains e.g. the kind of image which will be in the "PIXEL_DATA" and the size of the chunk.

Offset	Name	Description	Size [byte]
0x0000	CHUNK_TYPE	Defines the type of the chunk. For each distinct chunk an own type is defined.	4
0x0004	CHUNK_SIZE	Size of the whole image chunk in bytes. After this count of bytes the next chunk starts.	4
0x0008	HEADER_SIZE	Number of bytes starting from 0x0000 until PIXEL_DATA.	4
0x000C	HEADER_VERSION	Version number of the header	4
0x0010	IMAGE_WIDTH	Image width in pixel	4
0x0014	IMAGE_HEIGHT	Image height in pixel	4
0x0018	PIXEL_FORMAT	Pixel format	4

Offset	Name	Description	Size [byte]
0x001C	TIME_STAMP	Time stamp in microseconds (deprecated)	4
0x0020	FRAME_COUNT	Frame counter	4
0x0024	STATUS_CODE	Errors of the device	4
0x0028	TIME_STAMP_SEC	Time stamp in seconds	4
0x002C	TIME_STAMP_NSEC	Time stamp in nanoseconds	4
0x0030	PIXEL_DATA	The pixel data in the given type and dimension of the image. Padded to 4-byte boundary.	4

Available chunk types:

Constant	Value	Description
RADIAL_DISTANCE_IMAGE	100	<p>Each pixel of the distance matrix denotes the ToF distance measured by the corresponding pixel or group of pixels of the imager. The distance value is corrected by the camera's calibration, excluding effects caused by multipath and multiple objects contributions (e.g. "flying pixels"). Reference point is the optical centre of the camera inside the camera housing.</p> <p>Invalid PMD pixels (e.g. due to saturation) have a value of zero.</p> <p>Data type: 16-bit unsigned integer (little endian)</p> <p>Unit: millimetres</p>
NORM_AMPLITUDE_IMAGE	101	<p>Each pixel of the normalized amplitude image denotes the raw amplitude (see amplitude image below for further explanation), normalized to exposure time. Furthermore, vignetting effects are compensated, ie the darkening of pixels at the image border is corrected. The visual impression of this grayscale image is comparable to that of a common 2D camera.</p> <p>Invalid PMD pixels (e.g. due to saturation) have an amplitude value of 0.</p> <p>Data type: 16-bit unsigned integer</p>
AMPLITUDE_IMAGE	103	<p>Each pixel of the amplitude matrix denotes the amount of modulated light (i.e. the light from the camera's active illumination) which is reflected by the appropriate object. Higher values indicate higher PMD signal strengths and thus a lower amount of noise on the corresponding distance measurements. The amplitude value is directly derived from the PMD phase measurements without normalisation to exposure time. In multiple exposure mode, the lack of normalisation may lead (depending on the chosen exposure times) to inhomogeneous amplitude image impression, if a certain pixel is taken from the short exposure time and some of its neighbours are not.</p> <p>Invalid PMD pixels (e.g. due to saturation) have an amplitude value of 0.</p> <p>Data type: 16-bit unsigned integer</p>
GRAYSCALE_IMAGE	104	<p>Each pixel of the amplitude matrix denotes the amount of modulated light which is reflected by the appropriate object (i.e. the light from the camera's active illumination). Higher values indicate higher PMD signal strengths and thus a lower amount of noise on the corresponding distance measurements. The amplitude value is directly derived from the PMD phase measurements without normalisation to exposure time.</p>

Constant	Value	Description
CARTESIAN_X_COMPONENT	200	The X matrix denotes the X component of the Cartesian coordinate of a PMD 3D measurement. The origin of the camera's coordinate system is in the middle of the lens' front glass, if the extrinsic parameters are all set to 0. Data type: 16-bit signed integer Unit: millimetres
CARTESIAN_Y_COMPONENT	201	The Y matrix denotes the Y component of the Cartesian coordinate of a PMD 3D measurement. The origin of the camera's coordinate system is in the middle of the lens' front glass, if the extrinsic parameters are all set to 0. Data type: 16-bit signed integer Unit: millimetres
CARTESIAN_Z_COMPONENT	202	The Z matrix denotes the Z component of the Cartesian coordinate of a PMD 3D measurement. The origin of the camera's coordinate system is in the middle of the lens' front glass, if the extrinsic parameters are all set to 0. Data type: 16-bit signed integer Unit: millimetres
CARTESIAN_ALL	203	CARTESIAN_X_COMPONENT, CARTESIAN_Y_COMPONENT, CARTESIAN_Z_COMPONENT
UNIT_VECTOR_ALL	223	The unit vector matrix contains 3 values [ex, ey, ez] for each PMD pixel, i.e. the data layout is [ex_1,ey_1,ez_1, ... ex_N, ey_N, ez_N], where N is the number of PMD pixels. Data type: 32-bit floating point number (3x per pixel)
CONFIDENCE_IMAGE	300	See Additional Information for Image Data (→ 14.3.4)
DIAGNOSTIC	302	See Receiving Images (→ 14.3.2)
JSON_DIAGNOSTIC	305	Items with JSON formatted diagnostic data is formatted like this: <pre>{ "AcquisitionDuration": 20.391, "EvaluationDuration": 37.728, "FrameDuration": 37.728, "FrameRate": 15.202, "TemperatureIllu": 52.9 }</pre> Unit for durations: millimetres Unit for framerates: Hz Unit for temperature: °C

Constant	Value	Description
EXTRINSIC_CALIB	400	<p>The transformation from one cartesian coordinate system to another is defined by a 6 degrees of freedom vector (DOF): [trans_x, trans_y, trans_z, rot_x, rot_y, rot_z]. Let R be the product of the common "clockwise" 3D-rotation matrices: $R = R_x * R_y * R_z$</p> <p>The transformation of a point P is specified by $P_t = R * P + [trans_x, trans_y, trans_z]$.</p> <p>The device extrinsic calibration can be set by the user, but it may be changed by an automatic calibration feature of the device.</p> <p>Data type: 32-bit floating point number (little endian)</p> <p>Unit for trans_x, trans_y, trans_z: millimetres</p> <p>Unit for rot_x, rot_y, rot_z: °</p>
JSON_MODEL	500	Model data in JSON
MODEL_ROIMASK	501	ROI mask for internal debugging purposes
SNAPSHOT_IMAGE	600	Snapshot image

Pixel format:

Constant	Value	Description
FORMAT_8U	0	8-bit unsigned integer
FORMAT_8S	1	8-bit signed integer
FORMAT_16U	2	16-bit unsigned integer
FORMAT_16S	3	16-bit signed integer
FORMAT_32U	4	32-bit unsigned integer
FORMAT_32S	5	32-bit signed integer
FORMAT_32F	6	32-bit floating point number
FORMAT_64U	7	64-bit unsigned integer
FORMAT_64F	8	64-bit floating point number
Reserved	9	N/A
FORMAT_32F_3	10	Vector with 3x32-bit floating point number

14.3.4 Additional Information for CONFIDENCE_IMAGE

Further information for the confidence image:

Bit	Value	Description
0	1 = pixel invalid	Pixel invalid The pixel is invalid. To determine whether a pixel is valid or not only this bit needs to be checked. The reason why the bit is invalid is recorded in the other confidence bits.
1	1 = pixel saturated	Pixel is saturated Contributes to pixel validity: yes
2	1 = bad A-B symmetry	A-B pixel symmetry The A-B symmetry value of the four phase measurements is above threshold. Remark: This symmetry value is used to detect motion artefacts. Noise (e.g. due to strong ambient light or very short integration times) or PMD interference may also contribute. Contributes to pixel validity: yes
3	1 = amplitude below minimum amplitude threshold	Amplitude limits The amplitude value is below minimum amplitude threshold. Contributes to pixel validity: yes
4+5	Bit 5, bit 4 0 0 = unused 0 1 = shortest exposure time (only used in 3 exposure mode) 1 0 = middle exposure time in 3 exposure mode, short exposure in double exposure mode 1 1 = longest exposure time (always 1 in single exposure mode)	Exposure time indicator The two bits indicate which exposure time was used in a multiple exposure measurement. Contributes to pixel validity: no
6	1 = pixel is clipped	Clipping box on 3D data If clipping is active this bit indicates that the pixel coordinates are outside the defined volume. Contributes to pixel validity: yes
7	1 = suspect/defective pixel	Suspect pixel This pixel has been marked as "suspect" or "defective" and values have been replaced by interpolated values from the surroundings. Contributes to pixel validity: no

14.3.5 Configuration of PCIC Output

The user has the possibility to define his own PCIC output. This configuration is only valid for the current PCIC connection. It does not affect any other connection and will get lost after disconnecting.

For configuring the PCIC output a “flexible” layouter concept is used, represented by a JSON string. The format of the default configuration is as follows:

```
{
  "layouter": "flexible",
  "format": { "dataencoding": "ascii" },
  "elements": [
    { "type": "string", "value": "star", "id": "start_string" },
    { "type": "blob", "id": "normalized_amplitude_image" },
    { "type": "blob", "id": "x_image" },
    { "type": "blob", "id": "y_image" },
    { "type": "blob", "id": "z_image" },
    { "type": "blob", "id": "confidence_image" },
    { "type": "blob", "id": "diagnostic_data" },
    { "type": "string", "value": "stop", "id": "end_string" }
  ]
}
```

This string can be retrieved by the C? command, altered and sent back using the c command.

The layout software has the following main object properties:

Name	Description	Details
layouter	Defines the basic data output format. So far only “flexible” is supported	Type: string
format	Defines format details, the definitions in the main object are the defaults for any of the following data elements (e.g. if it says dataencoding=binary, all data elements should be binary encoded instead of ASCII).	Type: object
elements	List of data elements which must be written.	Type: array of objects

The actual data is defined within the “elements” properties and may consist of these settings:

Name	Description	Details
type	Defines the type of data which must be written. The data might be stored in a different type (e.g. stored as integer but should be output as Float32) The type "records" will need some special handling.	Type: string
id	Defines an identifier for this data element. If there is no fixed value (property "value"), the data should be retrieved via id.	Type: string
value	Optional property for defining a fixed output value.	Type: any JSON value
format	Type-dependent option for fine-tuning the output format. E.g. cut an integer to less than 4 bytes.	Type: object

Available values for the type property:

Type	Description
records	Defines that this element represents a list of records. If type is set to "records", there must be an "elements" property. The "elements" property defines which data should be written per record.
string	Data is written as string. Most of the time this will be used with "value" property to write fixed start, end or delimiter text. Text encoding should be UTF8 if there is nothing else specified in format properties.
float32	Data is written as floating point number. This has a lot of formatting options (at least with "flexible" layout software) See following section about format properties.
uint32	Data is written as integer. This has a lot of formatting options (at least with "flexible" layout software) See following section about format properties.
int32	Data is written as integer. This has a lot of formatting options (at least with "flexible" layout software) See following section about format properties.
uint16	Limits the output to two bytes in binary encoding, besides the binary limitation it acts like uint32.
int16	Limits the output to two bytes in binary encoding, besides the binary limitation it acts like int32.
uint8	Limits the output to one byte in binary encoding, besides the binary limitation it acts like uint32.
int8	Limits the output to one byte in binary encoding, besides the binary limitation it acts like int32.
blob	Data is written as a BLOB (byte by byte as if it came from the data provider). (Binary Large Object)

Depending on the desired data format the user may tune his output data with further "format" properties.

Common format properties:

Format properties	Allowed values	Default
dataencoding	"ascii" or "binary" can be defined in top-level-object and overwritten by element objects.	"ascii"
scale	"float value with decimal separator" to scale the results for output byte width	1.0
offset	"float value with decimal separator"	0.0

Binary format properties:

Format properties	Allowed values	Default
order	Little, big and network	Little

ASCII format properties:

Format properties	Allowed values	Default
width	Output width. If the resulting value exceeds the width field the result will not be truncated.	0
fill	Fill character	" "
precision	Precision is the number of digits behind the decimal separator.	6
displayformat	Fixed, scientific	Fixed
alignment	Left, right	Right
decimalseparator	7-bit characters for e.g. "."	."
base	Defines if the output should be: <ul style="list-style-type: none"> • binary (2) • octal (8) • decimal (10) • hexadecimal (16) 	10

UK

Example of a format configuration of the temperature (id: temp_illu) element.

1. Illumination temperature like this "33,5__":

```
c000000226{ "layouter": "flexible", "format": { "dataencoding": "ascii" },
"elements": [ { "type": "float32", "id": "temp_illu", "format": { "width": 7,
"precision": 1, "fill": "_", "alignment": "left", "decimalseparator": "," }
} ] }
```

2. Illumination temperature as binary (16-bit integer, 1/10 °C):

```
c000000194{ "layouter": "flexible", "format": { "dataencoding": "ascii"
}, "elements": [ { "type": "int16", "id": "temp_illu", "format": {
"dataencoding": "binary", "order": "network", "scale": 10 } } ] }
```

3. Illumination temperature in °F (e.g. "92.3 Fahrenheit"):

```
c000000227{ "layouter": "flexible", "format": { "dataencoding": "ascii" },
"elements": [ { "type": "float32", "id": "temp_illu", "format": { "precision":
1, "scale": 1.8, "offset": 32 } }, { "type": "string", "value": " Fahrenheit"
} ] }
```

The following element IDs are available:

ID	Description	Native data type
activeapp_id	Active application, shows which of the 32 application-configurations is currently active	32-bit unsigned integer
all_cartesian_vector_matrices	All Cartesian images (X+Y+Z) concatenated to one package	16-bit signed integer
all_unit_vector_matrices	Matrix of unit vectors. Each element consists of a 3 component vector [e_x, e_y, e_z]	Float32
amplitude_image	PMD raw amplitude image	16-bit unsigned integer
confidence_image	Confidence image	8-bit unsigned integer
distance_image	Radial distance image	16-bit unsigned integer unit: millimetres
evaltime	Evaluation time for current frame in milliseconds	32-bit unsigned integer
extrinsic_calibration	Extrinsic calibration, consisting of 3 translation parameters (unit: millimeters) and 3 angles (unit: degree): [t_x, t_y, t_z, alpha_x, alpha_y, alpha_z]	Float32
framerate	Current frame rate in Hz	Float32
normalized_amplitude_image	Normalized amplitude image	16-bit unsigned integer
temp_front1	Invalid temperature, the output is 3276.7	Float32, unit: °C
temp_illu	Temperature measured in the device while capturing this result Measured on the illumination board	Float32, unit: °C
x_image y_image z_image	Cartesian coordinates for each pixel Each dimension is a separate image	16-bit signed integer

For the main object on devices with statistics feature the following IDs are available:

ID	Description	Native data type
statistics_overall_count	Allows the user to output the statistics value with the result of the frame, maps to ModelResults: adv_statistics.number_of_frames	uint32
statistics_passed_count	Allows the user to output the statistics value with the result of the frame, maps to ModelResults: adv_statistics.number_of_passed_frames	uint32
statistics_failed_count	Allows the user to output the statistics value with the result of the frame, maps to ModelResults: adv_statistics.number_of_failed_frames	uint32
statistics_aborted_count	Allows the user to output the statistics value with the result of the frame, maps to ModelResults: adv_statistics.number_of_aborted_frames	uint32
statistics_acquisition_time_min	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_acquisition.min	float32
statistics_acquisition_time_mean	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_acquisition.mean	float32
statistics_acquisition_time_max	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_acquisition.max	float32
statistics_evaluation_time_min	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_evaluation.min	float32
statistics_evaluation_time_mean	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_evaluation.mean	float32
statistics_evaluation_time_max	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_evaluation.max	float32
statistics_frame_duration_min	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_duration.min	float32
statistics_frame_duration_mean	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_duration.mean	float32
statistics_frame_duration_max	Allows the user to output the statistics value with the result of the frame,maps to ModelResults: adv_statistics.frame_duration.max	float32

UK

14.4 Process Interface Command Reference



All received messages which are sent because of the following commands will be sent without “start”/”stop” at the beginning or ending of the string.

14.4.1 a Command (activate application)

Command	a<application number>	
Description	Activates the selected application	
Type	Action	
Reply	*	
	!	<ul style="list-style-type: none"> • Application not available • <application number> contains wrong value • External application switching activated • Device is in an invalid state for this command, e.g. configuration mode
	?	Invalid command length
Note	<application number> 2 digits for the application number as decimal value	

14.4.2 A? Command (occupancy of application list)

Command	A?	
Description	Requests the occupancy of the application list	
Type	Request	
Reply	<amount><t><number active application><t> ... <number><t><number>	
	?	Invalid command length
	!	Invalid state (e.g. no application active)
Note	<amount> char string with 3 digits for the amount of applications saved on the device as decimal number <t> tabulator (0x09) <number active application> 2 digits for the active application <number> 2 digits for the application number	The active application is repeated within the application list.

14.4.3 c Command (upload PCIC output configuration)

Command	c<length><configuration>	
Description	Uploads a PCIC output configuration lasting this session	
Type	Action	
Reply	*	
	!	<ul style="list-style-type: none"> • Error in configuration • Wrong data length
	?	Invalid command length
Note	<length> 9 digits as decimal value for the data length <configuration> configuration data	

UK

14.4.4 C? Command (retrieve current PCIC configuration)

Command	C?	
Description	Retrieves the current PCIC configuration	
Type	Request	
Reply	<length><configuration>	
	?	Invalid command length
Note	<length> 9 digits as decimal value for the data length <configuration> configuration data	

14.4.5 E? Command (request current error state)

Command	E?	
Description	Requests the current error state	
Type	Request	
Reply	<code>	
	!	Invalid state (e.g. configuration mode)
	?	Invalid command length
Note	<ul style="list-style-type: none"> • <code> Error code with 8 digits as a decimal value. It contains leading zeros. 	

14.4.6 G? Command (request device information)

Command	G?	
Description	Requests device information	
Type	Request	
Reply	<p><vendor><t><article number><t> <name><t><location><t><description><t><ip> <subnet mask><t><gateway><t><MAC><t><DHCP><t><port number></p>	
Note	<ul style="list-style-type: none"> • <vendor> IFM ELECTRONIC • <t> Tabulator (0x09) • <article number> e.g. O3D300 • <name> UTF8 Unicode string • <location> UTF8 Unicode string • <description> UTF8 Unicode string • <ip> IP address of the device as ASCII character sting e.g. 192.168.0.96 • <port number> port number of the XML-RPC • <subnet mask> subnet mask of the device as ASCII e.g. 192.168.0.96 • <gateway> gateway of the device as ASCII e.g 192.168.0.96 • <MAC> MAC adress of the device as ASCII e.g. AA:AA:AA:AA:AA:AA • <DHCP> ASCII string "0" for off and "1" for on 	

14.4.7 H? Command (return a list of available commands)

Command	H?	
Description	Returns a list of available commands	
Type	Request	
Reply	<p>H? - show this list</p> <p>t - execute Trigger</p> <p>T? - execute Trigger and wait for data</p> <p>o<io-id><io-state> - sets IO state</p> <p>O<io-id>? - get IO state</p> <p>I<image-id>? - get last image of defined type</p> <p>A? - get application list</p> <p>p<state> - activate / deactivate data output</p> <p>a<application number> - set active application</p> <p>E? - get last error</p> <p>V? - get current protocol version</p> <p>v<version> - sets protocol version</p> <p>c<length of configuration file><configuration file> - configures process date formatting</p> <p>C? - show current configuration</p> <p>G? - show device information</p> <p>S? - show statistics</p> <p>L? - retrieves the connection ID</p> <p>f<id><reserved><value> - set parameter value</p>	

UK

14.4.8 I? Command (request last image taken)

Command	I<image-ID>?	
Description	Request last image taken	
Type	Request	
Reply	<length><image data>	
	!	<ul style="list-style-type: none"> No image available Wrong ID
	?	<ul style="list-style-type: none"> Invalid command length
Note	<image-ID> 2 digits for the image type <length> char string with exactly 9 digits as decimal number for the image data size in bytes <image data> image data	Valid image ID: 01 - amplitude image 02 - normalised amplitude image 03 - distance image 04 - X image (distance information) 05 - Y image (distance information) 06 - Z image (distance information) 07 - confidence image (status information) 08 - extrinsic calibration 09 - unit_vector_matrix_ex, ey,ez 10 - last result output as formatted for this connection 11 - all distance images: X, Y, and Z

14.4.9 o Command (set logic state of a ID)

Command	o<IO-ID><IO-state>	
Description	Sets the logic state of a specific ID	
Type	Action	
Reply	*	
	!	Invalid state (e.g. configuration mode)
	?	Invalid command length
Note	<ul style="list-style-type: none"> <IO-ID> 2 digits for digital output: "01" for IO1 "02" for IO2 "03" for IO3 <IO-state> 1 digit for the state: "0" for logic state low "1" for logic state high 	

14.4.10 O? Command (request state of a ID)

Command	O<IO-ID>?	
Description	Requests the state of a specific ID	
Type	Request	
Reply	<IO-ID><IO-state>	
	!	<ul style="list-style-type: none"> Invalid state (e.g. configuration mode) Wrong ID
	?	Invalid command length
Note	<ul style="list-style-type: none"> <IO-ID> 2 digits for digital output: "01" for IO1 "02" for IO2 "03" for IO3 <IO-state> 1 digit for the state: "0" for logic state low "1" for logic state high 	<p>The camera supports ID 1 and ID 2. The sensor supports ID 1, ID 2 and ID 3.</p>

UK

14.4.11 p Command (turn PCIC output on or off)

Command	p<state>	
Description	Turns the PCIC output on or off	
Type	Action	
Reply	*	
	!	<state> contains wrong value
	?	Invalid command length
Note	<state> 1 digit 0: deactivates all asynchronous output 1: activates asynchronous result output 2: activates asynchronous error output 3: activates asynchronous error and data output 4: activates asynchronous notifications 5: activates asynchronous notifications and asynchronous result 6: activates asynchronous notifications and asynchronous error output 7: activates all outputs	<p>On device restart the value configured within the application is essential for the output of data.</p> <p>This command can be executed in any device state.</p> <p>By default the error codes will not be provided by the device.</p>

14.4.12 S? Command (request current decoding statistics)

Command	S?	
Description	Requests current decoding statistics	
Type	Request	
Reply	<number of results><t><number of positive decodings><t><number of false decodings>	
	!	No application active
Note	<t> tabulator (0x09) <number of results> Images taken since application start. 10 digits decimal value with leading 0s <number of positive decodings> Number of decodings leading to a positive result. 10 digits decimal value with leading 0s <number of false decodings> Number of decodings leading to a negative result. 10 digits decimal value with leading 0s	

14.4.13 t Command (execute asynchronous trigger)

Command	t	
Description	Executes trigger. The result data is send asynchronously	
Type	Action	
Reply	*	Trigger was executed, the device captures an image and evaluates the result.
	!	<ul style="list-style-type: none"> ● Device is busy with an evaluation ● Device is in an invalid state for this command, e.g. configuration mode ● Device is set to a different trigger source ● No active application

14.4.14 T? Command (execute synchronous trigger)

Command	T?	
Description	Executes trigger. The result data is send synchronously	
Type	Request	
Reply	Process data within the configured layout	Trigger was executed, the device captures an image, evaluates the result and sends the process data.
	!	<ul style="list-style-type: none"> • Device is busy with an evaluation • Device is in an invalid state for this command, e.g. configuration mode • Device is set to a different trigger source • No active application

UK

14.4.15 v Command (set current protocol version)

Command	v<version>	
Description	Sets the current protocol version. The device configuration is not affected	
Type	Action	
Reply	*	
	!	Invalid version
	?	Invalid command length
Note	<version> 2 digits for the protocol version	(→ 14.3.1)



The default protocol version is „V3“.

14.4.16 V? Command (request current protocol version)

Command	V?	
Description	Requests current protocol version	
Type	Request	
Reply	<current version><empty><min version><empty><max version>	
Note	<current version> 2 digits for the currently set version <empty> space sign: 0x20 <min/max version> 2 digits for the available min and max version that can be set	

14.5 Error codes

By default the error codes will not be provided by the device. The p command can activate their provision (→ 14.4.11).

Error code ID	Description
10000001	Maximum number of connections exceeded
110001001	Boot timeout
110001002	Fatal software error
110001003	Unknown hardware
110001006	Trigger overrun
110002000	Short circuit on Ready for Trigger
110002001	Short circuit on OUT1
110002002	Short circuit on OUT2
110002003	Reverse feeding
110003000	Vled overvoltage
110003001	Vled undervoltage
110003002	Vmod overvoltage
110003003	Vmod undervoltage
110003004	Mainboard overvoltage
110003005	Mainboard undervoltage
110003006	Supply overvoltage
110003007	Supply undervoltage
110003008	VFEMon alarm
110003009	PMIC supply alarm
110004000	Illumination overtemperature

14.6 XML-RPC Command Reference

14.6.1 Parameter API



The parameters `setParameter`, `getParameter`, `getAllParameters` and `getAllParameterLimits` are implemented in the following RPC objects:

- Device
- Network
- Application
- ImagerConfig
- Filter
- Model

setParameter

Method name	<code>setParameter</code>
Description	Sets a parameter to a specific value
Input parameters	1. Name of parameter:string 2. New value: string
Output parameters	Empty string (compatibility with classic XmlRPC client)

getParameter

Method name	<code>getParameter</code>
Description	Returns the current value of the parameter
Input parameters	Name of parameter: string
Output parameters	Value of parameter: string

getAllParameters

Method name	<code>getAllParameters</code>
Description	Returns all parameters of the object in one data structure
Input parameters	None
Output parameters	1. Struct (name contains the parameter name, value contains the stringified parameter value)

getAllParameterLimits

Method name	<code>getAllParameterLimits</code>
Description	Returns limits of all numeric parameters, that have limits defined on the device
Input parameters	None
Output parameters	1. Struct of Structs (name in first struct is the parameter name, substructs contains: min :string, max :string) E.g. { "ExposureTime1": { "min": "123", "max": "432" }, "ExposureTime2": { "min": "123", "max": "432" } }

Parameter string encoding

Non-string parameters must be encoded in the following format.

Type	Stringified
bool	"true" / "false" setParameter method also accepts "1"/"0", getter methods must always return "true"/"false"
int	decimal (e.g "-1234" / "1234") Values should be in the range of int32 ($-2^{31} .. 2^{31}$)
double	English floating point notation (optional with exponent) E.g. "1.2", ".3", "4.5e6", "-7E-8", "-inf", "nan"



Structured types (array or structs) can't be put into parameter storage in an general way. Encoding of arrays must specified on specific parameters.

14.6.2 Main Object

getParameter

Method name	<code>getParameter</code>
Description	Getter for the device-global parameters
Input parameters	Name of a device parameter: string
Output parameters	Value of the requested parameter: string

getAllParameters

Method name	<code>getAllParameters</code>
Description	Getter for the parameters described here. This is an additional getter outside of edit sessions, so it is possible to read device information without login.
Input parameters	none
Output parameters	Struct (name contains the parameter name, value contains the stringified parameter value)

getSWVersion

Method name	getSWVersion
Description	Returns version information of all software components
Input parameters	none
Output parameters	Struct of strings (e.g. { "IFM_Software": "0.01.07", "Frontend": "01.05.02", ... }) *mandatory keys: "IFM_Software" "Linux" "Main_Application" "Diagnostic_Controller" "Algorithm_Version" "Calibration_Version" "Calibration_Device"

UK

getHWInfo

Method name	getHWInfo
Description	Returns hardware information of all components
Input parameters	none
Output parameters	Struct of strings (e.g. { "MACAddress": "00:02:01:40:06:C9", "Frontend": "#!01_F340_001_...", ... }) *mandatory keys: "MACAddress" "Connector" "Diagnose" "Frontend" "Illumination" "Mainboard"

getApplicationList

Method name	getApplicationList
Description	Delivers basic information of all applications stored on the device.
Input parameters	none
Output parameters	Array of structs (Index: int, Id: int, Name: string, Description: string)

requestSession

Method name	<code>requestSession</code>
Description	<p>Requests a session object for access to the configuration and for changing the device operating mode.</p> <p>This blocks parallel editing and allows protection of editing with a password.</p> <p>The ID could optionally be defined by the external system but it must be the defined format (32char "hex").</p> <p>If it is called with only one parameter, the device will generate a session ID.</p> <p>The session will start with a default timeout ("SessionTimeout" device parameter), the timeout can be extended by calling "heartbeat".</p> <p>The device will stay in RUN mode.</p> <p>If password is disabled on the device, the value given as password parameter is ignored.</p>
Input parameters	<ol style="list-style-type: none"> 1. Password: string 2. Session ID: string (optional)
Output parameters	Session ID: string

reboot

Method name	<code>reboot</code>
Description	Reboot system, parameter defines which mode/system will be booted
Input parameters	<p>Type of system that should be booted after shutdown: int</p> <p>0: Productive mode</p> <p>1: Recovery mode</p>
Output parameters	Output: string

systemCommand

Method name	<code>systemCommand</code>
Description	Performs a generic command on the device.
Input parameters	<ol style="list-style-type: none"> 1. Command: string 2. Parameter: string
Output parameters	Output: string

14.6.3 Session Object

heartbeat

Method name	heartbeat
Description	Extends the life time of the edit session. If the given value is outside the range of "SessionTimeout", the saved default timeout will be used.
Input parameters	Requested timeout interval till next heartbeat, in seconds: int
Output parameters	The used timeout interval, in seconds: int

UK

cancelSession

Method name	cancelSession
Description	Explicit stop of this session If an application is still in edit mode, it will implicitly do the same as "stopEditingApplication".
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

exportConfig

Method name	exportConfig
Description	Exports the whole configuration of the sensor device
Input parameters	none
Output parameters	Configuration as a data BLOB: binary/base64

importConfig

Method name	importConfig
Description	Imports whole configuration with the option to skip specific parts
Input parameters	1. Configuration as a data BLOB: binary/base64 2. Flags describing which parts should be loaded: 0x0001: Includes configuration (Name, Description, Location, ...) 0x0002: Includes network configuration (IP, DHCP, ...) 0x0010: Includes all application configurations
Output parameters	Empty string (compatibility with classic XmlRPC client)

exportApplication

Method name	exportApplication
Description	Exports one application config
Input parameters	Application index
Output parameters	Application config as a data BLOB: binary/base64

importApplication

Method name	<code>importApplication</code>
Description	Imports an application config and creates a new application with it. The device will put the new application on the first free index.
Input parameters	Application config as one data BLOB: binary/base64
Output parameters	Index of new application

setOperatingMode

Method name	<code>setOperatingMode</code>
Description	Changes the operating mode of the device. Setting this to "edit" will enable the "edit mode object" on RPC.
Input parameters	Mode: integer 0: Run mode 1: Edit mode
Output parameters	Empty string (compatibility with classic XmlRPC client)

setTemporaryApplicationParameters

Method name	<code>setTemporaryApplicationParameters</code>
Description	<p>Set application parameters in run mode.</p> <p>The parameter names follow a prefix scheme similarly to the object hierarchy within the XMLRPC interface. For example</p> <ul style="list-style-type: none"> parameters of the application object have no prefix, parameters of the imager configuration object have the prefix "imager_001/", parameters of the model with ID 2 have the prefix "model_002/" <p>The parameters "imager_001/ExposureTime", "imager_001/ExposureTimeRatio" and "imager_001/Channel" of the imager configuration are supported. All additional parameters are ignored.</p> <p>If a parameter appears more than once in the parameter list, the behavior is undefined which value is chosen for the parameter. Exposure times are clamped to their allowed range, depending on the exposure mode.</p> <p>The complete set of parameters depending on the exposure mode must be provided. For example</p> <ul style="list-style-type: none"> "ExposureTime" only for single exposure modes, "ExposureTime" and "ExposureTimeRatio" for double exposure modes. <p>Otherwise the behavior is undefined. "Channel" parameter values outside of the allowed range of the used exposure mode are ignored, and for non-numeric values the behavior is undefined.</p> <p>Example: <code>setTemporaryApplicationParameters [{"imager_001/ExposureTime": "100"}]</code></p>
Input parameters	Parameter list (struct containing key value pairs, consisting of keys: parameter names and values: new parameter values)
Output parameters	Empty string (compatibility with classic XmlRPC client)



The changes are not persistent and are lost when entering edit mode or turning the device off.

14.6.4 Edit Mode Object

factoryReset

Method name	factoryReset
Description	Resets all configurations to factory settings
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)



A factory reset will delete all applications which are saved on the camera.

UK

editApplication

Method name	editApplication
Description	Puts a specified application into the edit status. This will attach an application object to the RPC interface. The name of the object will be application independent. This does not change the "ActiveApplication" parameter.
Input parameters	Application index: int
Output parameters	Empty string (compatibility with classic XmlRPC client)

stopEditingApplication

Method name	stopEditingApplication
Description	Tells the device that editing this application was finished. Unsaved changes are discarded.
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

createApplication

Method name	createApplication
Description	Creates an "empty" application. The embedded side should initialise all needed parameters and structures.
Input parameters	none
Output parameters	Index of new application: int

copyApplication

Method name	copyApplication
Description	Creates a new application by copying the configuration of another application. The device will generate an ID for the new application and put it on a free index.
Input parameters	Index of the application which should be copied: int
Output parameters	Index of new application: int

deleteApplication

Method name	deleteApplication
Description	Deletes the application from sensor If the deleted application was the active one, the sensor will have no active application anymore until the user picks one.
Input parameters	Index of application: int
Output parameters	Empty string (compatibility with classic XmlRPC client)

moveApplications

Method name	moveApplications
Description	Moves applications to other index. There must be all applications in the new list, none of them duplicated and no index used twice. The ID is a fixed value that stays the same as long as the application stays on the sensor. The index could be changed and is used to address the application via PCIC, XML-RPC and digital IO.
Input parameters	Array of structs (Id: int, Index: int)
Output parameters	Empty string (compatibility with classic XmlRPC client)

14.6.5 Device Config Object**activatePassword**

Method name	activatePassword
Description	Sets a password and activates it for the next edit session. Making this change persistently requires to call "save" on device config.
Input parameters	Password: string
Output parameters	Empty string (compatibility with classic XmlRPC client)

disablePassword

Method name	disablePassword
Description	Disables the password protection. Making this change persistently requires to call "save" on device config.
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

save

Method name	save
Description	Stores current configuration in persistent memory. If this is not called after changing device parameters (via setParameter), changes will get lost on reboot.
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

Parameters of device config

Methods for parameter access are defined here:

Parameter name	Data type	Description
Name	String (utf8)	User-defined name of the device (max. 64 characters).
Description	String (utf8)	User-defined description of the device (max. 500 characters).
ActiveApplication	Int *has limits	Index of active application This applies only to RUN mode: * defines the application active on startup (if static-application switching is disabled) * contains the current active application (could also be changed via PCIC command) * 0 means no application is active
PcicTcpPort	Int	TCP/IP port for PCIC connections.
PcicProtocolVersion	Int *has limits	Sub-protocol of PCIC, see specification of PCIC.
IOLogicType	Int *has limits	Defines logic type of all digital pins. Allowed values: 0: NPN 1: PNP
IODebouncing	Bool	Applies to all inputs
IOExternApplicationSwitch	Int *has limits	Allowed values: 0: off 1: static via I/O 2: pulse driven via I/O 3: pulse driven via trigger

Parameter name	Data type	Description
SessionTimeout	Int *has limits	Number of seconds which a session stays before a call to "heartbeat" method is needed
ServiceReportFailedBuffer	Int *has limits	Number of buffers reserved for failed results
ServiceReportPassedBuffer	Int *has limits	Number of buffers reserved for passed results
ExtrinsicCalibTransX	Double Unit: millimetres	Extrinsic calibration, transition in X direction
ExtrinsicCalibTransY	Double Unit: millimetres	Extrinsic calibration, transition in Y direction
ExtrinsicCalibTransZ	Double Unit: millimetres	Extrinsic calibration, transition in Z direction
ExtrinsicCalibRotX	Double Unit: degrees	Extrinsic calibration, rotation around X axis
ExtrinsicCalibRotY	Double Unit: degrees	Extrinsic calibration, rotation around Y axis
ExtrinsicCalibRotZ	Double Unit: degrees	Extrinsic calibration, rotation around Z axis
IPAddressConfig	Int	readonly: The GUI requires to know if the device is on a discovery IP address for multiple-use cases. This information was extended to reflect all kinds of IP-address situations. Allowed values: 0: Static (IP address explicitly defined inside the device) 1: DHCP (using a DHCP server in the network) 2: LinkLocal (configured to DHCP, but no server which provided an address) 3: Discovery (changed by IP4Discovery mechanism)
PasswordActivated	Bool	readonly: Is true if the password protection is enabled
OperatingMode	Int	readonly: Mode of device (RUN, EDIT) see "setOperatingMode" (the setter is outside the edit mode but inside session)
DeviceType	String	readonly: Delivers a type description, unique by imager, evaluation logic and device interface.
ArticleNumber	String	readonly: Official catalogue number
ArticleStatus	String	readonly: Official two-letter status code
UpTime	Double	readonly: Hours since last reboot
ImageTimestampReference	Int Unit: microseconds	readonly: This returns the current timestamp as a reference for the timestamps in the received images.
TemperatureFront1	Double Unit: celsius	Invalid temperature, the output is 3276.7

Parameter name	Data type	Description
TemperatureFront2	Double Unit: celsius	Invalid temperature, the output is 3276.7
TemperatureIllu	Double Unit: celsius	readonly: Temperature measured in the device. Measured on the illumination board.

*has limits: parameters with this marker are listed in the reply of getAllParameterLimits method.

Default values of device config parameters

The default values of the device configuration parameters are:

Parameter name	Data type	Description
Name	String (utf8)	"New sensor"
Description	String (utf8)	""
ActiveApplication	Int *has limits	0
PcicTcpPort	Int	50010
PcicProtocolVersion	Int *has limits	3
IOLogicType	Int *has limits	1
IODebouncing	Bool	true
IOExternApplicationSwitch	Int *has limits	0
SessionTimeout	Int *has limits	30
ExtrinsicCalibTransX	Double Unit: millimetres	0.0
ExtrinsicCalibTransY	Double Unit: millimetres	0.0
ExtrinsicCalibTransZ	Double Unit: millimetres	0.0
ExtrinsicCalibRotX	Double Unit: degrees	0.0
ExtrinsicCalibRotY	Double Unit: degrees	0.0
ExtrinsicCalibRotZ	Double Unit: degrees	0.0
IPAddressConfig	Int	0
PasswordActivated	Bool	false
OperatingMode	Int	0
ServiceReportFailedBuffer	Int	15
ServiceReportPassedBuffer	Int	15

For all other device config parameters there are no defined default values because they are either device-dependent (DeviceType, ArticleNumber, ArticleStatus) or volatile (UpTime, ImageTimestampReference).

Minimum and maximum values of device config parameters

The minimum and maximum values of the device configuration parameters are:

Parameter name	Minimum value	Maximum value
ActiveApplication	0	32
PcicProtocolVersion	1	4
IOLogicType	0	1
IOExternApplicationSwitch	0	3
SessionTimeout	5	300

14.6.6 Device/Network Config Object

saveAndActivateConfig

Method name	saveAndActivateConfig
Description	Reinitialise the network interface so that it uses the configuration which was set by the other RPC methods. There will be no XMLRPC reply because the network interface is instantly reset.
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

14.6.7 Application Config Object

save

Method name	save
Description	Stores current configuration in persistent memory. This is also be possible if the application is not yet in an "activatable" status.
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

forceTrigger

Method name	forceTrigger
Description	Executes a software trigger of currently active application.
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

Validate

Method name	validate
Description	Validates the application. This means it checks if the application can be activated.
Input parameters	none
Output parameters	Array of fault structs (Id: int, Text: string)
Fault scenarios	none

Parameters of application

Methods for parameter access are defined here:

Parameter name	Data type	Description
Name	String (utf8)	User-defined name of the application (max. 64 characters).
Description	String (utf8)	User-defined description of the application (max. 500 characters).
TriggerMode	Int *has limits	Allowed values: 1: free run 2: process interface 3: positive edge 4: negative edge 5: positive and negative edge
PcicTcpResultSchema	String	It defines which images and result data will be sent. It will also define the order of data elements and additional separators. Contains single-enabling/disabling of AmplitudeImage, IntensityImage, DistanceImage, XImage, YImage, ZImage, ConfidenceImage, DiagnosticData (→ 14.3.5)
LogicGraph	String	JSON string describing a flow graph which allows to program the logic between model results and output pins.
Type	String	Internal use
TemplateInfo	String	A generic JSON storage, where the GUI could store additional data about the used template GUI (versions and additional parameter decisions). This data should not be used by the device, it should only be stored on the device.

*has limits: parameters with this marker are listed in the reply of getAllParameterLimits method

Default values of application parameters

The default values of application parameters are:

Parameter name	Data type	Description
Name	String (utf8)	"new application"
Description	String (utf8)	""
TriggerMode	Int *has limits	1
PcicTcpResultSchema	String	""
LogicGraph	String	""
Type	String	"Camera"
TemplateInfo	String	""

Minimum and maximum values of application parameters

The minimum and maximum values of application parameters are:

Parameter name	Minimum value	Maximum value
TriggerMode	1	5

14.6.8 Application/Imager Config Object

changeType

Method name	changeType
Description	Changes the type of imager configuration. This changes setting of available parameters and might also change available RPC methods.
Input parameters	Type: string
Output parameters	Empty string (compatibility with classic XmlRPC client)

availableTypes

Method name	availableTypes
Description	Lists all available imager configuration types.
Input parameters	none
Output parameters	Array of strings

Parameters of all types of application imager config

Methods for parameter access are defined here:

Parameter name	Data type	Description
Type	String	readonly: Type of imager configuration, see Change Type Method
FrameRate	Double *has limits	Target frame rate in frames per second for free run mode.
ClippingLeft	Double *has limits	Lower value of clipping area in width
ClippingTop	Double *has limits	Lower value of clipping area in height
ClippingRight	Double *has limits	Upper value of clipping area in width
ClippingBottom	Double *has limits	Upper value of clipping area in height
ContinuousAutoExposure	Bool	Enables the continuous adaptation of the integration time during decoding

Parameter name	Data type	Description
SpatialFilterType	Int *has limits	Allowed values: 0: off 1: median filter 2: mean filter 3: bilateral filter
TemporalFilterType	Int *has limits	Allowed values: 0: off 1: temporal mean filter 2: adaptive exponential filter
EnableFilterDistanceImage	Bool	Activates the filter for the distance image
EnableFilterAmplitudeImage	Bool	Activates the filter for the amplitude image
SymmetryThreshold	Double *has limits	
MinimumAmplitude	Double *has limits	Defines the minimum amplitude used for the validity of a pixel.
TwoFreqMaxLineDistPercentage	Double *has limits	
ThreeFreqMax2FLineDistPercentage	Double *has limits	
ThreeFreqMax3FLineDistPercentage	Double *has limits	
EnableAmplitudeCorrection	Bool	Enables the correction of the amplitude values
EnableRectificationDistanceImage	Bool	Enables the rectification of the distance image
EnableRectificationAmplitudeImage	Bool	Enables the rectification of the normalized amplitude image
ExposureTimeList	String	readonly: A list of all current exposure times separated by ";" It should contain 3 values in "_high" types, 2 values in "_moderate" types and 1 value in "_low" types. The list is sorted in ascending order.
MaxAllowedLEDFrameRate	Double readonly	Maximum allowed frame rate for current settings, which complies with the LED duty cycle
Resolution	Int *has limits	Resolution of output image: 0: 176 x 132 (2x2 binning) 1: 352 x 264 (no binning), only available for 100k camera
EnableFastFrequency	Bool	Enables rolling evaluation in multi-frequency modes (parameter will be ignored in single-frequency modes)
ClippingCuboid	JSON	Object describing the clipping cuboid

Parameter name	Data type	Description
AutoExposureReferenceType	Int *has limits	Select part of the image to be used for continuous autoexposure: 0: whole image 1: ROIs (→ AutoExposureReferenceROI) 2: Reference Point (→ AutoExposureReferencePointX and AutoExposureReferencePointY)
AutoExposureReferenceROI	String	ROI definition for AutoExposureReferenceType "1"
AutoExposureReferencePointX	Int *has limits	X coordinate of reference point used for AutoExposureReferenceType "2"
AutoExposureReferencePointY	Int *has limits	Y coordinate of reference point used for AutoExposureReferenceType "2"
AutoExposureMaxExposureTime	Int *has limits	Maximum exposure time that should be used when continuous autoexposure is activated (→ AutoExposureReferenceType)

*has limits: parameters with this marker are listed in the reply of getAllParameterLimits method

Default values of common imager config parameters

The default values of the common imager configuration parameters are:

Parameter name	Data type	Description
Type	String	"under5m_low"
FrameRate	Double	5.0
ContinuousAutoExposure	Bool	false
SpatialFilterType	Int	0
TemporalFilterType	Int	0
EnableFilterDistanceImage	Bool	true
EnableFilterAmplitudeImage	Bool	true
SymmetryThreshold	Double	0.4
MinimumAmplitude	Double	42
TwoFreqMaxLineDistPercentage	Double	80
ThreeFreqMax2FLineDistPercentage	Double	80
ThreeFreqMax3FLineDistPercentage	Double	80
EnableAmplitudeCorrection	Bool	true
EnableRectificationDistanceImage	Bool	false
EnableRectificationAmplitudeImage	Bool	false
Resolution	Int	0
EnableFastFrequency	Bool	false
ClippingCuboid	String	'{"XMin": -3.402823e+38, "XMax": 3.402823e+38, "YMin": -3.402823e+38, "YMax": 3.402823e+38, "ZMin": -3.402823e+38, "ZMax": 3.402823e+38}'
AutoExposureReferenceType	Int	0
AutoExposureReferenceROI	String	'{"ROIs":[{"id":0,"group":0, "type":"Rect", "width":130, "height":100, "angle":0, "center_x":88, "center_y":66}]}'
AutoExposureReferencePointX	Int	88
AutoExposureReferencePointY	Int	66

Parameter name	Data type	Description
AutoExposureMaxExposureTime	Int	10000

Minimum and maximum values of common imager config parameters

The minimum and maximum values of the common imager configuration parameters are:

Parameter name	Minimum value	Maximum value
FrameRate	0.0167	30.0
SpatialFilterType	0	3
TemporalFilterType	0	2
SymmetryThreshold	0	
MinimumAmplitude	0	
TwoFreqMaxLineDistPercentage	0	100
ThreeFreqMax2FLineDistPercentage	0	100
ThreeFreqMax3FLineDistPercentage	0	100
Resolution	0	1
AutoExposureReferenceType	0	2
AutoExposureReferencePointX	1	352
AutoExposureReferencePointY	1	264
AutoExposureMaxExposureTime	10	10000

UK

Parameters only in "under5m_low"-type of application imager config

Parameter name	Data type	Description
ExposureTime	Int *has limits	Time for the exposure The 2nd exposure time will be calculated based on the first one.
ExposureTimeRatio	Int *has limits	Ratio of long exposure time to short exposure time.
Channel	Int *has limits	Allowed values: 0: non-group use (like channel1 but additional GUI option) 1: channel1 2: channel2 3: channel3

Default values of the "under5m_low" mode parameters

Parameter name	Data type	Default value
ExposureTime	Int	1000
Channel	Int	0

Minimum and maximum values of the "under5m_low" mode parameters

Parameter name	Minimum value	Maximum value
ExposureTime	1	10000
Channel	0	3

Parameters only in "under5m_moderate"-type of application imager config

Parameter name	Data type	Description
ExposureTime	Int *has limits	Time for the long exposure The 2nd exposure time will be calculated based on the first one.
Channel	Int *has limits	Allowed values: 0: non-group use (like channel1) 1: channel1 2: channel2 3: channel3

Default values of the "under5m_moderate" mode parameters

Parameter name	Data type	Default value
ExposureTime	Int	1000
ExposureTimeRatio	Int	40
Channel	Int	0

Minimum and maximum values of the "under5m_moderate" mode parameters

Parameter name	Minimum value	Maximum value
ExposureTime	1	10000
ExposureTimeRatio	2	50
Channel	0	3

Parameters only in "under5m_high"-type of application imager config

Parameter name	Data type	Description
Channel	Int *has limits	Allowed values: 0: non-group use (like channel1 but additional GUI option) 1: channel1 2: channel2 3: channel3

Default values of the "under5m_high" mode parameters

Parameter name	Data type	Default value
Channel	Int	0

Minimum and maximum values of the "under5m_high" mode parameters

Parameter name	Minimum value	Maximum value
Channel	0	3

Parameters only in "upto30m_low"-type of application imager config

Parameter name	Data type	Description
ExposureTime	Int *has limits	Time for the long exposure
Channel	Int *has limits	Allowed values: 0: non-group use (like channel1) 1: channel1 2: channel2 3: channel3

UK

Default values of the "upto30m_low" mode parameters

Parameter name	Data type	Default value
ExposureTime	Int	1000
Channel	Int	0

Minimum and maximum values of the "upto30m_low" mode parameters

Parameter name	Minimum value	Maximum value
ExposureTime	1	10000
Channel	0	3

Parameters only in "upto30m_moderate"-type of application imager config

Parameter name	Data type	Description
ExposureTime	Int *has limits	Time for the long exposure The 2nd exposure time will be calculated based on the first one.
ExposureTimeRatio	Int *has limits	Ratio of long exposure time to short exposure time
Channel	Int *has limits	Allowed values: 0: non-group use (like channel1 but additional GUI option) 1: channel1 2: channel2 3: channel3

Default values of the "upto30m_moderate" mode parameters

Parameter name	Data type	Default value
ExposureTime	Int	1000
ExposureTimeRatio	Int	40
Channel	Int	0

Minimum and maximum values of the "upto30m_moderate" mode parameters

Parameter name	Minimum value	Maximum value
ExposureTime	1	10000
ExposureTimeRatio	2	50
Channel	0	3

Parameters only in "upto30m_high"-type of application imager config

Parameter name	Data type	Description
Channel	Int *has limits	Allowed values: 0: non-group use (like channel1 but additional GUI option) 1: channel1 2: channel2 3: channel3

Default values of the "upto30m_high" mode parameters

Parameter name	Data type	Default value
Channel	Int	0

Minimum and maximum values of the "upto30m_high" mode parameters

Parameter name	Minimum value	Maximum value
Channel	0	3

14.6.9 Image Settings and Filter Parameters

To set the spatial or temporal filter use the general “setter” method.

Parameters of spatial median, spatial mean and spatial bilateral filter

Parameter name	Data type	Description
MaskSize	Int	Allowed values: 0: 3x3 1: 5x5

UK

Parameters of temporal mean filter

Parameter name	Data type	Description
NumberOfImages	Int	Limit: 2..25