# Programmers Guide
# Interface description
# 3D Camera

UK

**O3X100**

706447 / 00 12 / 2018

# Contents

# 1. Preliminary note

This document describes the software interfaces of the O3X1xx 3D camera.

## 1.1 Symbols used

► Instructions

> Reaction, result

[…] Designation of keys and buttons

"…" Name of display text

→ Cross-reference

Important note
Non-compliance may result in malfunction or interference.

Information
Supplementary note

## 1.2 Safety instructions

Please read the operating instructions prior to set-up of the device. Ensure that the device is suitable for your application without any restrictions.

If the operating instructions or the technical data are not adhered to, personal injury and damage to property can occur.

## 1.3 Further documents

- Operating instructions
- Interface description

The documents can be downloaded at:
www.ifm.com

# 2. System requirements

## 2.1 Software

Windows 7 (32/64 bit), Windows 8.1 (32/64 bit), Windows 10 (32/64 bit)

## 2.2 Hardware and accessories

Hardware:

– Camera of the O3X1xx product family
– PC with x86 or x64 type processor
– Screen: min. 1024 x 768 pixels, 32 bit colour depth
– Ethernet interface

The Ethernet interface can be retrofitted with an USB-to-Ethernet adapter.

Accessories:

– Power supply 24 V, 1.6 A, min. peak current 2.4 A

You will find further information about available accessories at:
www.ifm.com

# 3. 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



# 4. XML-RPC Interface

In case the device should not be configured by the ifm Vision Assistant, 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.

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.

A detailed example is available on the web at:
https://github.com/ifm/ifm3d-examples/blob/master/file_io/ex-file_io.cpp

## 4.1 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 O3X1xx command:

POST /api/rpc/v1/com.ifm.efector/ HTTP/1.1<CR><LF>

Host: 192.168.0.69<CR><LF>

Content-Type: text/xml<CR><LF>

User-Agent: Python-xmlrpc/3.4<CR><LF>

Content-Length: 160<CR><LF>

<CR><LF>

<?xml version='1.0'?> <CR><LF>

<methodCall><CR><LF>

<methodName>getParameter</methodName><CR><LF>

<params><CR><LF>

<param><CR><LF>

<value><string>Name</string></value><CR><LF>

</param><CR><LF>

</params><CR><LF>

</methodCall><CR><LF>

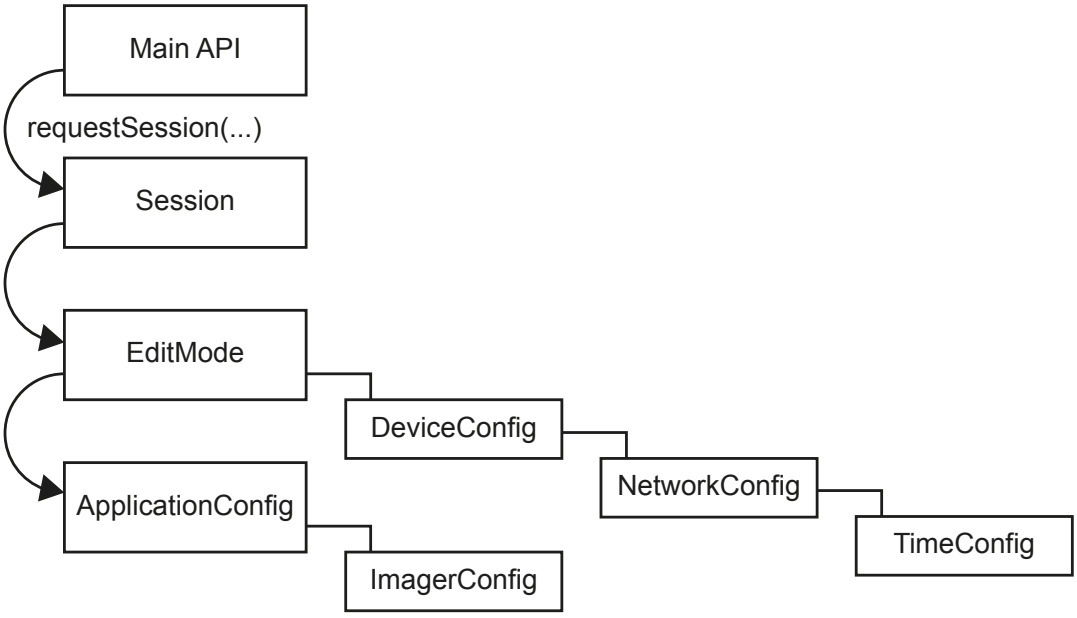
## 4.2 XML-RPC Objects

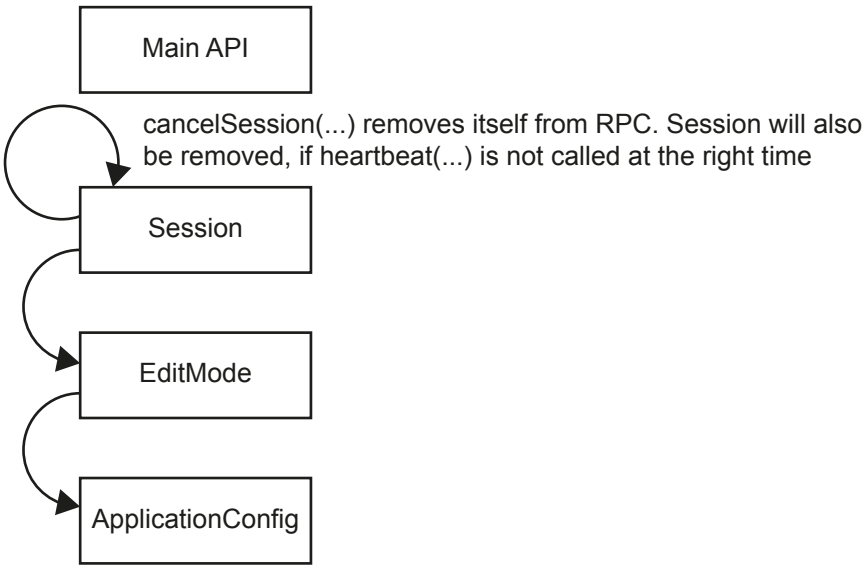
To communicate and to configure the device via XML-RPC the XML-RPC commands have to use different XML-RPC objects. Different commands need different XML-RPC objects (see XML-RPC command references).

The interface of O3X1xx 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:








### 4.2.1 Main Object

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

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

### 4.2.2 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.

### 4.2.3 EditMode Object

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

### 4.2.4 DeviceConfig Object

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

### 4.2.5 Device/NetworkConfig Object

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

### 4.2.6 TimeConfig Object

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

### 4.2.7 Application Config Object

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

### 4.2.8 Application/Imager Config Object

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

As there is only one imager config on the device, 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 ifm Vision Assistant draft:

- upTo02m_low
- upTo02m_moderate
- upTo03m_low
- upTo03m_moderate
- upTo07m_low
- upTo07m_moderate
- upTo15m_low
- upTo15m_moderate
- upTo30m_low
- upTo30m_moderate

# 5. Process Interface

The process interface is used during the normal operation mode to get operational data (e.g. 3D images, process values) from the O3X1xx.

## 5.1 Protocol structure

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<br>(e.g. <code> is a placeholder for code) | | |
| [ ] | Optional argument<br>(possible but not required) | | |

| Command | Description |
|---|---|
| <content> | It is the command (e.g. trigger the unit). |
| <ticket> | It is a character string fixed to 0000. The device will reply with the same ticket number. |
| <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. |

It is not possible to send commands.

## 5.2 Receiving Images

For receiving the image data a TCP/IP socket communication is established. The default port number is 50010. After opening the socket communication, the O3X1xx 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 | „0000“ |
| Start sequence | String "star" (4 bytes) |
| Normalised amplitude image<br>Output format: 32-bit floating point number | 1 image |
| Distance image<br>Output format: 32-bit floating point number. Unit: m. | 1 image |
| Combined XYZ image<br>Output format: 32-bit floating point number. Unit: m. | 1 image |
| Confidence image<br>Output format: 8-bit unsigned integer | 1 image |
| Grayscale image<br>Output format: 32-bit floating point number | 1 image |
| Stop sequence | String "stop" (4 bytes) |
| Ticket signature | <CR><LF> |

Only configured images will be displayed.

## 5.3 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.

### 5.3.1 Chunk type

| 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_HEIGTH | Image height in pixel | 4 |
| 0x0018 | PIXEL_FORMAT | Pixel format | 4 |
| 0x001C | TIME_STAMP | Time stamp in microseconds | 4 |
| 0x0020 | FRAME_COUNT | Frame count according to algorithm output | 4 |
| 0x0024 | STATUS_CODE | Errors on the device | 4 |

| Offset | Name | Description | Size [byte] |
|---|---|---|---|
| 0x0028 | TIME_STAMP_SEC | Timestamp seconds | 4 |
| 0x002C | TIME_STAMP_NSEC | Timestamp nanoseconds | 4 |
| 0x0030 | PIXEL_DATA | The pixel data in the given type and dimension of the image. Padded to 4-byte boundary. | 4 |

### 5.3.2 Available chunk types

| Constant | Value | Description |
|---|---|---|
| USERDATA | 0 | Undefined user data with arbitrary content |
| RADIAL_DISTANCE_IMAGE | 100 | 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.<br>Invalid PMD pixels (e.g. due to saturation) have a value of zero.<br>Data type: 32-bit floating point number<br>Unit: millimetres |
| NORM_AMPLITUDE_IMAGE | 101 | 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.<br>Invalid PMD pixels (e.g. due to saturation) have an amplitude value of 0.<br>Data type: 32-bit floating point number |
| AMPLITUDE_IMAGE | 103 | 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.<br>Invalid PMD pixels (e.g. due to saturation) have an amplitude value of 0.<br>Data type: 32-bit floating point number |
| 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.<br>Data type: 32-bit floating point number<br>Unit: millimetres |

| Constant | Value | Description |
|---|---|---|
| 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.<br>Data type: 32-bit floating point number<br>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.<br>Data type: 32-bit floating point number<br>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.<br>Data type: 32-bit floating point number (3x per pixel) |
| CONFIDENCE_IMAGE | 300 | See Additional Information for Image Data (→ 5.4) |

### 5.3.3 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 |

## 5.4 Additional Information for CONFIDENCE_IMAGE

| Bit | Value | Description |
|---|---|---|
| 0 | 1 = pixel invalid | Pixel invalid<br>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<br>Contributes to pixel validity: yes |
| 2 | 1 = bad A-B symmetry | A-B pixel symmetry<br>The A-B symmetry value of the four phase measurements is above threshold.<br>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.<br>Contributes to pixel validity: yes |
| 3 | 1 = amplitude below minimum amplitude threshold | Amplitude limits<br>The amplitude value is below minimum amplitude threshold.<br>Contributes to pixel validity: yes |
| 4 | 1 = exposure time | Exposure time indicator.<br>The bit is always set in single exposure mode. It is set in double exposure mode when the long exposure time is used.<br>Contributes to pixel validity: no |
| 5 | 1 = assignment error | Double Frequency assignment error.<br>The bit indicates a double frequency mismatch while calculating the final distance. Possible reasons are fast movement in the scene or disturbances due to straylight.<br>Contributes to pixel validity: yes |
| 6 | | Reserved |
| 7 | | Reserved |

# 6. XML-RPC Command Reference

## 6.1 Parameter API

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

- Device
- Network
- Application
- ImagerConfig
- Filter

### setParameter

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

### getParameter

| | |
|---|---|
| Method name | `getParameter` |
| Description | Returns the current value of the parameter. |
| Input parameters | Name of parameter: string |
| Output parameters | Value of parameter: string |

### getAllParameters

| | |
|---|---|
| Method name | `getAllParameters` |
| Description | Returns all parameters of the object in one data structure. |
| Input parameters | None |
| Output parameters | Struct (name contains the parameter name, value contains the stringified parameter value) |

### getAllParameterLimits

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

### Parameter string encoding

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

| Type | Stringified |
|---|---|
| bool | "true" / "false"<br>setParameter method also accepts "1"/"0", getter methods must always return "true"/"false" |
| int | decimal ( e.g "-1234" / "1234" )<br>Values should be in the range of int32 (-2^31 .. 2^31) |
| double | English floating point notation (optional with exponent)<br>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.



## 6.2 Main Object

### getParameter

| | |
|---|---|
| Method name | `getParameter` |
| Description | Getter for the device-global parameters.<br>This is an additional getter outside of edit sessions, so it is possible to read device information without login. |
| Input parameters | Name of a device parameter: string |
| Output parameters | Value of the requested parameter: string |

### getAllParameters

| | |
|---|---|
| Method name | `getAllParameters` |
| Description | Getter for the device-global parameters.<br>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", ... } )<br>*mandatory keys:<br>"IFM_Software"<br>"Linux"<br>"Main_Application"<br>"Algorithm_Version"<br>"Calibration_Version"<br>"Calibration_Device" |

### 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", ... } )<br>*mandatory keys:<br>"MACAddress"<br>"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 | `requestSession` |
| Description | Requests a session object for access to the configuration and for changing the device operating mode.<br>This blocks parallel editing and allows protection of editing with a password.<br>The ID could optionally be defined by the external system but it must be the defined format (32char "hex").<br>If it is called with only one parameter, the device will generate a session ID.<br>The session will start with a default timeout ("SessionTimeout" device parameter), the timeout can be extended by calling "heartbeat".<br>The device will stay in RUN mode.<br>If password is disabled on the device, the value given as password parameter is ignored. |
| Input parameters | 1. Password: string<br>2. Session ID: string |
| Output parameters | Session ID: string |

### reboot

| | |
|---|---|
| Method name | `reboot` |
| Description | Reboot system, parameter defines which mode/system will be booted. |
| Input parameters | Type of system that should be booted after shutdown: int<br>0: Productive mode<br>1: Recovery mode |
| Output parameters | Empty string (compatibility with classic XmlRPC-client) |

### systemCommand

| | |
|---|---|
| Method name | `systemCommand` |
| Description | Performs a generic command on the device. |
| Input parameters | 1. Command: string<br>2. Parameter: string |
| Output parameters | string |

### getTraceLogs

| | |
|---|---|
| Method name | `getTraceLogs` |
| Description | Returns entries from the internal log buffer of the device. It can contain informal, error or trace messages. |
| Input parameters | nLogs (Integer): max. number of logs to fetch from the IO manager. If the value is 0, all logs are fetched. |
| Output parameters | logs (Array of Strings): trace logs |

### getClientCompatibilityList

| | |
|---|---|
| Method name | `getClientCompatibilityList` |
| Description | The device must be able to define which type and version of operating program is compatible. |
| Input parameters | none |
| Output parameters | Array of strings:<br>• Each string contains: "[VendorID]-[OperatingProgramID],[major].[minor]" (IDs are 4 digit-hex, version dec).<br>• Each field could be a wildcard with '*'.<br>• E.g. "0001-0001,1.0" or "0001-*,1.*". |

### getUnitVectors

| | |
|---|---|
| Method name | `getUnitVectors` |
| Description | Returns a chunk containing the current unit vectors. |
| Input parameters | none |
| Output parameters | Unit vector chunk: binary/base64 |

### Trigger

| | |
|---|---|
| Method name | `trigger` |
| Description | Executes a software trigger. The trigger might be delayed in order to fulfill the pause time or maximum frame rate requirements. This delay takes place synchronously, i.e. this RPC returns after the trigger has been executed. |
| Input parameters | none |
| Output parameters | Empty string (compatibility with classic XmlRPC-client) |

## 6.3 Session Object

### heartbeat

| | |
|---|---|
| Method name | `heartbeat` |
| Description | Extends the life time of the edit session.<br>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 |



### cancelSession

| | |
|---|---|
| Method name | `cancelSession` |
| Description | Explicit stop of this session.<br>If an import or export is still being processed, the session is kept alive until the import or export has finished, although the method returns immediately. |
| 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<br>2. Flags describing which parts should be loaded:<br>0x0001: Includes configuration (Name, Description, Location, ...)<br>0x0002: Includes network configuration (IP, DHCP, ...)<br>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 | `importApplication` |
| Description | Imports an application config and creates a new application with it.<br>The name of the application should be based on the one stored in the exported config.<br>If the name should be unique, the sensor must generate a suffix in case of a naming conflict.<br>The device will get a new ID. If the config data contains a ID, it must be ignored.<br>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 | `setOperatingMode` |
| Description | Changes the operating mode of the device.<br>Setting this to "edit" will enable the "edit mode object" on RPC. |
| Input parameters | Mode: integer<br>0: Run mode<br>1: Edit mode |
| Output parameters | Empty string (compatibility with classic XmlRPC-client) |

The device will not distinguish between edit and run mode. The application will always behave as in run mode. As soon as a session is opened the application can immediately be edited.

After successfully calling "requestSession" the application object at URL "/api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/application/" is immediately available.

## 6.4 Edit Mode Object

The device will not distinguish between edit and run mode. The application will always behave as in run mode. As soon as a session is opened the application can immediately be edited.

### factoryReset

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



### changeNameAndDescription

| | |
|---|---|
| Method name | `changeNameAndDescription` |
| Description | Changes the name and the description of the application. |
| Input parameters | 1. Application index :int<br>2. New name of the application: string (utf8, max. 64 character)<br>3. New description of the application: string (utf8, max. 500 character) |
| Output parameters | Empty string (compatibility with classic XmlRPC-client) |

## 6.5 Device Config Object

### activatePassword

| | |
|---|---|
| Method name | `activatePassword` |
| Description | Sets a password and activates it for the next edit session.<br>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.<br>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.<br>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). |
| SessionTimeout | Int<br>*has limits | Number of seconds which a session stays before a call to "heartbeat" method is needed. |
| IPAddressConfig | Int | readonly: The ifm Vision Assistant 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.<br>Allowed values:<br>0: Static (IP address explicitly defined inside the device)<br>1: DHCP (using a DHCP server in the network)<br>2: LinkLocal (configured to DHCP, but no server which provided an address)<br>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)<br>Always returns 0 ("run mode"). Parameter is present for compatibility reasons. |
| 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<br>Unit: seconds | readonly: The timestamp is the current system time (UTC) in seconds since 1.1.1970. |
| TemperatureIllu | Double<br>Unit: celsius | readonly: Temperature measured in the device. |

*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 | Default value |
|---|---|---|
| Name | String (utf8) | "New sensor" |
| Description | String (utf8) | "" |
| SessionTimeout | Int<br>*has limits | 30 |
| IPAddressConfig | Int | 0 |
| PasswordActivated | Bool | false |
| OperatingMode | Int | 0 |

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 |
|---|---|---|
| SessionTimeout | 5 | 300 |

## 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.<br>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) |

### Parameters of Device/Network Config

Methods for parameter access are defined here:

| Parameter name | Data type | Description |
|---|---|---|
| StaticIPv4Address | String (utf8) | IPv4 address of the device. Only used if DHCP is disabled and if there is no temporary IP address set via discovery.<br>Only numeric IPv4 addresses in quad-dotted notation are allowed (e.g. "192.168.0.69") |
| StaticIPv4SubNetMask | String (utf8) | IPv4 network mask of the device.<br>Only numeric IPv4 addresses in quad-dotted notation are allowed (e.g. "255.255.255.0"). |
| StaticIPv4Gateway | String (utf8) | IPv4 gateway of the device.<br>Only numeric IPv4 addresses in quad-dotted notation are allowed (e.g. "192.168.0.69"). |
| UseDHCP | Bool | Selects whether the network should be configured via DHCP. |

| Parameter name | Data type | Description |
|---|---|---|
| MACAddress | String | Read only: MAC-Address of the device (format: "xx:xx:xx:xx:xx:xx"). |

### Default values of Device/Network Config parameters

The default values of the Device/Network configuration parameters are:

| Parameter name | Data type | Default value |
|---|---|---|
| StaticIPv4Address | String (utf8) | 192.168.0.69 |
| StaticIPv4SubNetMask | String (utf8) | 255.255.255.0 |
| StaticIPv4Gateway | String (utf8) | 192.168.0.201 |
| UseDHCP | Bool | false |
| MACAddress | String | - |

The parameter "MACAdress" has no defined default values, because they are device-dependent.

## 6.7 Time Config Object

### setCurrentTime

| Method name | `setCurrentTime` |
|---|---|
| Description | Sets the current system time. |
| Input parameters | Timestamp: int (UTC, seconds since 1.1.1970) |
| Output parameters | Empty string (compatibility with classic XmlRPC-client) |

### saveAndActivateConfig

| Method name | `saveAndActivateConfig` |
|---|---|
| Description | Save and immediately apply the current time configuration (might lead to jump in the system time). |
| Input parameters | none |
| Output parameters | Empty string (compatibility with classic XmlRPC-client) |

### Parameters of Time Config

Methods for parameter access are defined here:

| Parameter name | Data type | Description |
|---|---|---|
| NTPServers | String (utf8) | List of NTP servers which should be used for system time synchronization.<br>The server entries have to be separated with commas and/or whitespaces. Each server entry must be a numeric IPv4 address in dot notation (e.g. "192.168.0.69"). |
| WaitSyncTries | Int | Number of attempts to initially synchronize with a NTP server. Each attempt waits 10 seconds for a server response. |
| SynchronizationActivated | Bool | Enables synchronization of system time via NTP. |
| StartingSynchronization | Bool | Read only: Returns "true" during the initial NTP synchronization, which may result in a jump in the system time. Returns "false" afterwards. |

| Parameter name | Data type | Description |
|---|---|---|
| Syncing | Bool | Read only: Returns "true" if the system time is synchronized to a NTP server (i.e. if at least one of the three most recent synchronization attempts was successful). |
| CurrentTime | Int | Read only: Returns the current system time (UTC, seconds since 1.1.1970). |
| Stats | String | Read only: Returns a human readable string containg the synchronization state and accuracy for all NTP servers. |



### Default values of Time Config parameters

The default values of the time configuration parameters are:

| Parameter name | Data type | Default value |
|---|---|---|
| WaitSyncTries | Int | 2 |
| SynchronizationActivated | Bool | false |

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 Time Config parameters

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

| Parameter name | Minimum value | Maximum value |
|---|---|---|
| WaitSyncTries | 1 | 6 |

## 6.8 Application Config Object

### save

| Method name | `save` |
|---|---|
| Description | Stores current configuration in persistent memory.<br>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) |

### Validate

| Method name | `validate` |
|---|---|
| Description | Validates the application. This means it checks if the application can be activated.<br>The device should do this check, while activating an application. |
| Input parameters | none |
| Output parameters | Array of fault structs (Id: int, Text: string) |
| Fault scenarios | none |

### 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) |

## discardUnsavedChanges

| | |
|---|---|
| Method name | `discardUnsavedChanges` |
| Description | Resets all changed parameters of the application object and all child objects to their persistent values, i.e. the values at the time when "save" has been called last or when the session has been opened. |
| Input parameters | none |
| Output parameters | Empty string (compatibility with classic XmlRPC-client) |

## 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<br>*has limits | Selects how images are generated in the application.<br>Allowed values:<br>1: free run<br>2: software trigger |
| OutputDistanceImage | Bool | Output the radial distance image. |
| OutputAmplitudeImage | Bool | Output the amplitude image. |
| OutputGrayscaleImage | Bool | Output the grayscale image if supported by the selected exposure mode. See the description of the exposure modes whether a particular mode also provides a grayscale image. |
| OutputConfidenceImage | Bool | Output the confidence image. |
| OutputXYZImage | Bool | Output the point cloud image. |

*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 | Default value |
|---|---|---|
| Name | String (utf8) | "new application" |
| Description | String (utf8) | "" |
| TriggerMode | Int<br>*has limits | 1 |
| OutputDistanceImage | Bool | true |
| OutputAmplitudeImage | Bool | true |
| OutputGrayscaleImage | Bool | false |
| OutputConfidenceImage | Bool | false |
| OutputXYZImage | Bool | false |

### Minimum and maximum values of application parameters

The minimum and maximum values of application parameters are:

| Parameter name | Minimum value | Maximum value |
|---|---|---|
| TriggerMode | 1 | 2 |

## 6.9 Application/Imager Config Object

### changeType

| | |
|---|---|
| Method name | `changeType` |
| Description | Changes the type of imager configuration.<br>This changes setting of available parameters and might also change available RPC methods. |
| Input parameters | 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 |
|---|---|---|
| FrameRate | Double<br>*has limits | Target frame rate in frames per second for free run mode. |
| ExposureTime | Int<br>*has limits | Exposure time in microseconds. |
| SpatialFilterType | Int<br>*has limits | A filter for amplitude and distance images. Allowed values:<br>0: off<br>1: 3x3 median filter |
| TemporalFilterType | Int<br>*has limits | A filter for consecutive amplitude and distance images. Allowed values:<br>0: off<br>1: adaptive exponential filter |
| MinimumAmplitude | Double<br>*has limits | Minimum amplitude. |
| SymmetryThreshold | Double<br>*has limits | Symmetry threshold. |
| Type | String | Read only: Type of imager configuration, see changeType Method. For available exposure modes see (→ 6.10). |
| MaxAllowedFrameRate | Double | Read only: maximum allowed frame rate for the current application parameters. |

*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 | Default value |
|---|---|---|
| FrameRate | Double | 5.0 |
| ExposureTime | Int | 1000 |
| SpatialFilterType | Int | 0 |
| TemporalFilterType | Int | 0 |
| MinimumAmplitude | Double | 42 |
| SymmetryThreshold | Double | 0.4 |

## Minimum and maximum values of common imager config parameters

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

| Parameter name | Minimum value | Maximum value |
|---|---|---|
| FrameRate | 0.0167 | 30 |
| ExposureTime | depends on exposure mode | depends on exposure mode |
| SpatialFilterType | 0 | 1 |
| TemporalFilterType | 0 | 1 |
| MinimumAmplitude | 0 | 10000 |
| SymmetryThreshold | 0 | 1000 |

## Parameters only in double exposure modes of application imager config

| Parameter name | Data type | Description |
|---|---|---|
| ExposureTimeRatio | Double<br>*has limits | Ratio of long exposure time to short exposure time. Only available in double exposure modes. |

## Default values of double exposure mode parameters

| Parameter name | Data type | Default value |
|---|---|---|
| ExposureTimeRatio | Double | 40 |

## Minimum and maximum values of double exposure mode parameters

| Parameter name | Minimum value | Maximum value |
|---|---|---|
| ExposureTimeRatio | 2 | 50 |

## Parameters only in exposure modes with grayscale image of application imager config

| Parameter name | Data type | Description |
|---|---|---|
| ExposureTimeGrayscale | Int<br>*has limits | Exposure time in microseconds for the grayscale image. Only available in exposure modes with grayscale image. |

## Default values of exposure mode parameters with grayscale image

| Parameter name | Data type | Default value |
|---|---|---|
| ExposureTimeGrayscale | Int | 1000 |

## Minimum and maximum values of exposure mode parameters with grayscale image

| Parameter name | Minimum value | Maximum value |
|---|---|---|
| ExposureTimeGrayscale | 1 | 5000 |

## 6.10 Exposure modes

The following exposure modes are available:

- upTo02m_low
- upTo02m_moderate
- upTo03m_low
- upTo03m_moderate
- upTo07m_low
- upTo07m_moderate
- upTo15m_low
- upTo15m_moderate
- upTo30m_low
- upTo30m_moderate



The capture mode consists of:

| Section | Description |
|---|---|
| upTo30m_ | • Length of the unambiguous range<br>• 1 measurement frequency: unambiguous range < "7m"<br>• 2 measurement frequencies: unambiguous range >= "7m" |
| _moderate | • 2 exposure times |
| _low | • 1 exposure time |

The capture mode "upTo30m_moderate" is preset.