



Programming manual
Interface description

3D camera

O3X120

O3X130

GB

Contents

1	Preliminary note	3
1.1	Symbols used	3
1.2	Legal and copyright information	3
2	Safety instructions	4
3	System requirements	5
3.1	Required ports	5
4	XML-RPC interface	6
4.1	Sample command	6
4.2	Objects	7
5	Process interface	10
5.1	Protocol structure	10
5.2	Receiving images	10
5.3	Image data	11
5.3.1	Chunk type	11
5.3.2	Available chunk types	11
5.3.3	Pixel format	13
5.4	Additional information for CONFIDENCE_IMAGE	13
6	XML-RPC command reference	14
6.1	Parameter API	14
6.2	Main object	15
6.3	Session object	17
6.4	Edit mode object	19
6.5	Device config object	20
6.6	Device/Network config object	21
6.7	Time config object	22
6.8	Application config object	23
6.9	Application/Imager config object	25
6.10	Exposure modes	28

1 Preliminary note

You will find instructions, technical data, approvals and further information using the QR code on the unit / packaging or at www.ifm.com.

1.1 Symbols used

- ✓ Requirement
- ▶ 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 Legal and copyright information

© All rights reserved by ifm electronic gmbh. No part of these instructions may be reproduced and used without the consent of ifm electronic gmbh.

All product names, pictures, companies or other brands used on our pages are the property of the respective rights owners.

2 Safety instructions

Please read the operating instructions prior to set-up of the device. The device must be suitable for the application without any restrictions.

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

3 System requirements

Software

The following software is required for operation.

- Operating system: Windows 10 (32/64 bits)
- ifm Vision Assistant software: 2.5.23 or higher



Other versions of the software or firmware may contain changed or new functions which are not described in this software manual.

Hardware

The following hardware is required for operation:

- Hard disk: min. 1 GB free memory space
- Monitor: Resolution of min. 1024x768 pixels, 32-bit colour depth



▷ More information about available accessories at: www.ifm.com

3.1 Required ports

The following ports are required for the device 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 is not configured via the ifm Vision Assistant, the XML-RPC interface can be used instead.



General information about XML-RPC can be found on the website <http://xmlrpc.com/spec.md>

To send a command via the XML-RPC interface the command has to be in a special layout. Every command which is sent via the XML-RPC interface must end with carriage return <CR> and linefeed <LF>.

Several commands 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 command

All following XML-RPC commands 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 a 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 Objects

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

The interface of the device 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 may be necessary to send heartbeats to prevent a session timeout.

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

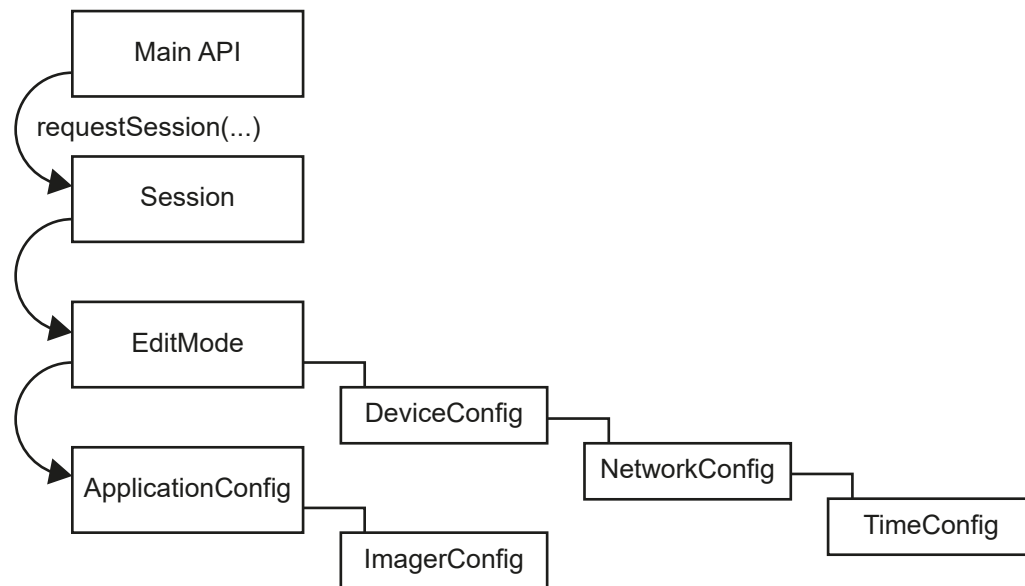


Fig. 1: XML-RPC objects: requestSession

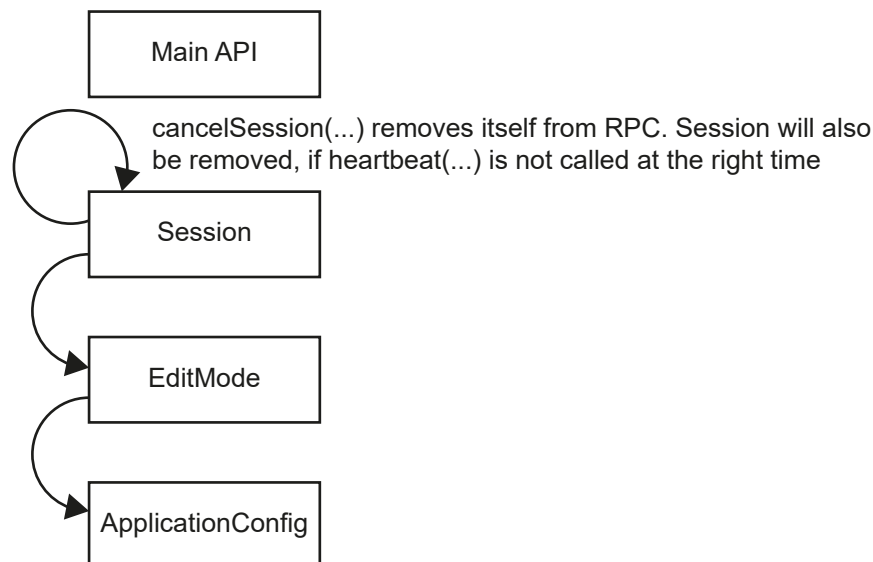


Fig. 2: XML-RPC objects: cancelSession

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.

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/
```

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/
```

TimeConfig Object

Object URI e.g.:

```
/api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/  
device/time/
```

Application Config Object

Object URI e.g.:

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

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 determined as "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:

- 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 from the device (e.g., 3D images, process values).

5.1 Protocol structure

This protocol corresponds 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)		

Tab. 1: Protocol structure

Command	Description
<content>	This is the command (e.g., trigger the unit).
<ticket>	This is a character string fixed to 0000. The device replies with the same ticket number.
<length>	This 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.

Tab. 2: Command description



It is not possible to send commands.

5.2 Receiving images

For receiving image data a TCP/IP socket communication is established. The default port number is 50010. After opening the socket communication, the device automatically sends the data through this socket to the TCP/IP client (PC). For this, the device must be in Free run mode.

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

Component	Description
Ticket	0000
Start sequence	String "star" (4 bytes)
Normalised amplitude image Output format: 32-bit floating point number	1 image
Distance image Output format: 32-bit floating point number. Unit: m	1 image
Combined XYZ image Output format: 32-bit floating point number. Unit: m	1 image

Component	Description
Confidence image Output format: 8-bit unsigned integer	1 image
Grayscale image Output format: 32-bit floating point number	1 image
Stop sequence	String " stop " (4 bytes)
Ticket signature	<CR><LF>

Tab. 3: Receiving images



Only configured images are displayed.

5.3 Image data

There is a separate chunk for each image. The chunk is part of the response frame data of the process interface.

The header of each chunk contains different kinds of information which are separated into bytes. This information contains the kind of image which is 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_HEIGHT	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 in the device	4
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

Tab. 4: Chunk type

5.3.2 Available chunk types

Constant	Value	Description
USERDATA	0	Undefined user data with arbitrary content.

Constant	Value	Description
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 device 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 device housing. Invalid PMD pixels (e.g., due to saturation) have a value of zero. Data type: 32-bit floating point number Unit: millimetre
NORM_AMPLITUDE_IMAGE	101	Each pixel of the normalised amplitude image denotes the raw amplitude (see amplitude image below for further explanation), normalised to exposure time. Furthermore, vignetting effects are compensated, 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. Invalid PMD pixels have an amplitude value "0" (e.g., due to saturation). Data type: 32-bit floating point number
AMPLITUDE_IMAGE	103	Each pixel of the amplitude matrix denotes the amount of modulated light, which is reflected by the appropriate object (the light from the 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. 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. Invalid PMD pixels have an amplitude value "0" (e.g., due to saturation). Data type: 32-bit floating point number
RADIAL_DISTANCE_NOISE	105	If the application parameter "OutputDistanceNoiseImage" is set to "true", the payload data will contain an image chunk with the chunk value "105". Data type: 16-unsigned integer Unit: 0.1 millimetre
CARTESIAN_X_COMPONENT	200	The X matrix denotes the X component of the Cartesian coordinate of a PMD 3D measurement. The origin of the device coordinate system is in the centre of the front glass, if the extrinsic parameters are all set to "0". Data type: 32-bit floating point number Unit: millimetre
CARTESIAN_Y_COMPONENT	201	The Y matrix denotes the Y component of the Cartesian coordinate of a PMD 3D measurement. The origin of the device coordinate system is in the centre of the front glass, if the extrinsic parameters are all set to "0". Data type: 32-bit floating point number Unit: millimetre
CARTESIAN_Z_COMPONENT	202	The Z matrix denotes the Z component of the Cartesian coordinate of a PMD 3D measurement. The origin of the device coordinate system is in the centre of the front glass, if the extrinsic parameters are all set to "0". Data type: 32-bit floating point number Unit: millimetre
CARTESIAN_ALL	203	CARTESIAN_X_COMPONENT CARTESIAN_Y_COMPONENT CARTESIAN_Z_COMPONENT
UNIT_VECTOR_ALL	223	The unit vector matrix contains 3 values for each PMD pixel (ex, ey, ez). 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)

Constant	Value	Description
CONFIDENCE_IMAGE	300	See Additional Information for Image Data. (→ Additional information for CONFIDENCE_IMAGE □ 13)

Tab. 5: Available chunk types

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
FORMAT_16U2	9	2x 16-bit unsigned integer
FORMAT_32F3	10	Vector with 3x32-bit floating point number
FORMAT_12U	11	12-bit unsigned integer (nested)

Tab. 6: Pixel format

5.4 Additional information for CONFIDENCE_IMAGE

Bit	Value	Description
0	1 : pixel invalid	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. This symmetry value is used to detect motion artefacts. Noise or PMD interference may also contribute (e.g. due to strong ambient light or very short integration times). 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	1 : exposure time	Exposure time indicator. The bit is always set in single exposure mode. It is set in double exposure mode when the long exposure time is used. Contributes to pixel validity: no
5	1 : assignment error	Double Frequency assignment error. 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. Contributes to pixel validity: yes
6		Reserved
7		Reserved

Tab. 7: Additional information for CONFIDENCE_IMAGE

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 2. New value: string
Output parameters	Empty string (compatibility with classic XmlRPC client)

Tab. 8: setParameter

getParameter

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

Tab. 9: getParameter

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)

Tab. 10: getAllParameters

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) E.g.: <pre>{ "ExposureTime1": { "min": "123", "max": "432" }, "ExposureTime2": { "min": "123", "max": "432" }} </pre>

Tab. 11: getAllParameterLimits

Parameter string encoding

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

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

Tab. 12: Parameter string encoding



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

6.2 Main object

getParameter

Method name	getParameter
Description	Getter for the global/general device parameters. 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

Tab. 13: getParameter

getAllParameters

Method name	getAllParameters
Description	Getter for the global/general device parameters. 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)

Tab. 14: getAllParameters

getSWVersion

Method name	getSWVersion
Description	Returns version information of all software components.
Input parameters	None
Output parameters	Struct of strings e.g. <code>{ " IFM_Software ": " 0.01.07 ", " Frontend ": " 01.05.02 ", ... }</code> Mandatory keys: IFM_Software Linux Main_Application Algorithm_Version Calibration_Version Calibration_Device

Tab. 15: getSWVersion

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 ", ... } Mandatory keys: MACAddress Mainboard

Tab. 16: getHWInfo

getApplicationList

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

Tab. 17: getApplicationList

requestSession

Method name	requestSession
Description	Requests a session object for access to the configuration and for changing the operating mode of the device. This blocks parallel editing and allows protection of editing with a password. The ID could optionally be defined by the external system, but it must be the defined format (32char "hex"). If it is called with only one parameter, the device generates a session ID. The session starts with a default timeout ("SessionTimeout" device parameter). The timeout can be extended by activating "heartbeat". The device stays in RUN mode. If the password is disabled in the device, the value given as password parameter is ignored.
Input parameters	1. Password: string 2. Session ID: string
Output parameters	Session ID: string

Tab. 18: requestSession

reboot

Method name	reboot
Description	Reboot system. The parameter defines which mode is booted.
Input parameters	Type of mode that should be booted after shutdown [int]: 0 : Productive mode 1 : Recovery mode
Output parameters	Empty string (compatibility with classic XmlRPC client)

Tab. 19: reboot

systemCommand

Method name	systemCommand
-------------	---------------

Description	Performs a generic command in the device.
Input parameters	1. Command: string 2. Parameter: string
Output parameters	string

Tab. 20: systemCommand

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

Tab. 21: getTraceLogs

getClientCompatibilityList

Method name	getClientCompatibilityList
Description	The device defines which type and version of operating program is compatible.
Input parameters	None
Output parameters	Array of strings: - Each string contains: "[VendorID]-[OperatingProgramID],[major].[minor]" (IDs are 4 digit-hex, version dec). - Each field could be a wildcard with "*". - E.g. "0001-0001,1.0" or "0001-*,1.*".

Tab. 22: getClientCompatibilityList

getUnitVectors

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

Tab. 23: getUnitVectors

Trigger

Method name	Trigger
Description	Executes a software trigger. The trigger might be delayed fulfilling the pause time or maximum frame rate requirements. This delay takes place synchronously. The RPC returns after the trigger has been executed.
Input parameters	none
Output parameters	Empty string (compatibility with classic XmlRPC client)

Tab. 24: Trigger

6.3 Session object

heartbeat

Method name	heartbeat
-------------	-----------

Description	Extends the lifetime of the edit session. If the given value is outside the range of "SessionTimeout", the saved default timeout is used.
Input parameters	Requested timeout interval until the next heartbeat, in seconds: int
Output parameters	The used timeout interval, in seconds: int

Tab. 25: heartbeat

cancelSession

Method name	cancelSession
Description	Explicit stop of this session. If an import or export is still being processed, the session is kept alive until the import or export is finished, although the method returns immediately.
Input parameters	None
Output parameters	Empty string (compatibility with classic XmlRPC client)

Tab. 26: cancelSession

exportConfig

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

Tab. 27: exportConfig

importConfig

Method name	importConfig
Description	Imports the 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)

Tab. 28: importConfig

exportApplication

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

Tab. 29: exportApplication

importApplication

Method name	importApplication
-------------	-------------------

Description	Imports an application config and creates a new application with it. The name of the application should be based on the one stored in the exported config. If the name should be unique, the device must generate a suffix in case of a naming conflict. The device gets a new ID. If the config data contains an ID, it must be ignored. The device puts the new application on the first free index.
Input parameters	Application config as one data BLOB: binary/base64
Output parameters	Index of new application

Tab. 30: importApplication

setOperatingMode

Method name	setOperatingMode
Description	Changes the operating mode of the device. Setting this to "edit" enables the "edit mode object" on RPC.
Input parameters	Mode [int]: 0 : Run mode 1 : Edit mode
Output parameters	Empty string (compatibility with classic XmlRPC client)

Tab. 31: setOperatingMode



The device does not distinguish between Edit and Run mode. The application always behaves as if in Run mode. As soon as a session is opened, the application can immediately be edited.

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

6.4 Edit mode object



The device does not distinguish between Edit and Run mode. The application always behaves as if 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)

Tab. 32: factoryReset

changeNameAndDescription

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

Tab. 33: changeNameAndDescription

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)

Tab. 34: activatePassword

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)

Tab. 35: disablePassword

save

Method name	save
Description	Stores the current configuration in the persistent memory. If this is not called after changing device parameters (via set-Parameter), changes are lost on reboot.
Input parameters	None
Output parameters	Empty string (compatibility with classic XmlRPC client)

Tab. 36: save

Parameters of Device config

Methods for parameter access are defined next.

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	Duration of a session in seconds before a call to "heartbeat" method is needed. The parameter has limits and is listed in the reply of the getAllParameterLimits method.
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 conditions. 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) Always returns 0 ("Run mode"). Parameter is present for compatibility reasons.

Parameter name	Data type	Description
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: seconds	readonly: The timestamp is the current system time (UTC) in seconds since 1.1.1970.
TemperatureIllu	Double Unit: celsius	readonly: Temperature measured in the device.

Tab. 37: Parameter access

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 The parameter has limits and is listed in the reply of the getAllParameterLimits method.	30
IPAddressConfig	Int	0
PasswordActivated	Bool	false
OperatingMode	Int	0

Tab. 38: Default values of device configuration parameters



For all other device config parameters there are no defined default values. Possible reasons for this are that 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

Tab. 39: Minimum and maximum values of Device Config parameters

6.6 Device/Network config object

saveAndActivateConfig

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

Tab. 40: saveAndActivateConfig

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. Only numeric IPv4 addresses in dotted quad notation are allowed (e.g. "192.168.0.69").
StaticIPv4SubNetMask	String (utf8)	IPv4 network mask of the device. Only numeric IPv4 addresses in dotted quad notation are allowed (e.g. "255.255.255.0").
StaticIPv4Gateway	String (utf8)	IPv4 gateway of the device. Only numeric IPv4 addresses in dotted quad notation are allowed (e.g. "192.168.0.69").
UseDHCP	Bool	Selects whether the network should be configured via DHCP.
MACAddress	String	Read only: MAC-Address of the device. Format: "xx:xx:xx:xx:xx:xx".

Tab. 41: Methods for parameter access

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	-

Tab. 42: Default values of Device/Network Config parameters



The parameter "MACAddress" has no 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)

Tab. 43: setCurrentTime

saveAndActivateConfig

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

Tab. 44: saveAndActivateConfig

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 synchronisation. The server entries must be separated with commas or whitespaces. Each server entry must be a numeric IPv4 address in dotted quad notation (e.g., "192.168.0.69").
WaitSyncTries	Int	Number of attempts to initially synchronise with a NTP server. Each attempt waits 10 seconds for a server response.
SynchronisationActivated	Bool	Enables synchronisation of system time via NTP.
StartingSynchronisation	Bool	Read only: Returns "true" during the initial NTP synchronisation, which may result in a jump in the system time. Returns "false" afterwards.
Syncing	Bool	Read only: Returns "true" if the system time is synchronised to a NTP server, if at least one of the three most recent synchronisation 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 containing the synchronisation state and accuracy for all NTP servers.

Tab. 45: Parameters of Time config

Default values of Time Config parameters

The default values of the time configuration parameters are:

Parameter name	Data type	Default value
WaitSyncTries	Int	2
SynchronisationActivated	Bool	false

Tab. 46: Default values of Time config



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

Tab. 47: Minimum and maximum values of Time Config parameters

6.8 Application config object

save

Method name	save
Description	Stores the current configuration in the persistent memory. This is also possible if the application is not yet in a status where it can be activated.
Input parameters	None
Output parameters	Empty string (compatibility with classic XmlRPC client)

Tab. 48: save

validate

Method name	validate
-------------	----------

Description	Validates the application. This means it checks if the application can be activated. 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

Tab. 49: validate

forceTrigger

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

Tab. 50: forceTrigger

discardUnsavedChanges

Method name	discardUnsavedChanges
Description	Resets all changed parameters of the application object and all child objects to their persistent values. This applies to 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)

Tab. 51: discardUnsavedChanges

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	Selects how images are generated in the application. Allowed values: 1 : free run 2 : software trigger The parameter has limits and is listed in the reply of the getAllParameterLimits method.
OutputDistanceImage	Bool	Outputs the radial distance image.
OutputAmplitudeImage	Bool	Outputs the amplitude image.
OutputGrayscaleImage	Bool	Outputs the grayscale image if supported by the selected exposure mode. See in the description of the exposure modes whether a particular mode also provides a grayscale image.
OutputConfidenceImage	Bool	Outputs the confidence image.
OutputXYZImage	Bool	Outputs the point cloud image.
OutputDistanceNoiseImage	Bool	Outputs the distance noise image.

Tab. 52: Parameters of application

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 The parameter has limits and is listed in the reply of the getAllParameterLimits method.	1
OutputDistanceImage	Bool	true
OutputAmplitudeImage	Bool	true
OutputGrayscaleImage	Bool	false
OutputConfidenceImage	Bool	false
OutputXYZImage	Bool	false
OutputDistanceNoiseImage	Bool	false

Tab. 53: Default values of application parameters

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
OutputDistanceNoiseImage	0	65535

Tab. 54: minimum and maximum values of application parameters

6.9 Application/Imager config object

changeType

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

Tab. 55: changeType

availableTypes

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

Tab. 56: availableTypes

Parameters of all types of application imager config

Methods for parameter access are defined here:

Parameter name	Data type	Description
FrameRate	Double	Target frame rate in frames per second for free run mode. The parameter has limits and is listed in the reply of the getAllParameterLimits method.

Parameter name	Data type	Description
ExposureTime	Int	Exposure time in microseconds. The parameter has limits and is listed in the reply of the <code>getAllParameterLimits</code> method.
SpatialFilterType	Int	A filter for amplitude and distance images. Allowed values: 0 : off 1 : 3x3 median filter The parameter has limits and is listed in the reply of the <code>getAllParameterLimits</code> method.
TemporalFilterType	Int	A filter for consecutive amplitude and distance images. Allowed values: 0 : off 1 : adaptive exponential filter The parameter has limits and is listed in the reply of the <code>getAllParameterLimits</code> method.
MinimumAmplitude	Double	Minimum amplitude. The parameter has limits and is listed in the reply of the <code>getAllParameterLimits</code> method.
SymmetryThreshold	Double	Symmetry threshold. The parameter has limits and is listed in the reply of the <code>getAllParameterLimits</code> method.
Type	String	Read only: Type of imager configuration, see <code>changeType</code> Method. Available exposure modes. (→ Exposure modes □ 28)
MaxAllowedFrameRate	Double	Read only: maximum allowed frame rate for the current application parameters.
EnableStraylightCorrection	Bool	Enables straylight correction.
AbsDistStraylightThreshold	Float	Absolute distance threshold for invalidating pixels due to straylight correction. Unit: [m]
RelAmpStraylightThreshold	Float	Relative amplitude threshold for invalidating pixels due to straylight correction.
EnableNoiseEstimation	Bool	Enables noise estimation.
DistNoiseThreshold	Float	Threshold for distance noise. "0": threshold is disabled Unit: [m]
CompensateAmbientLightDrift	Bool	Enables ambient light drift compensation.

Tab. 57: Parameters of all types of application imager config

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
EnableStraylightCorrection	Bool	false
AbsDistStraylightThreshold	Float	0.05
RelAmpStraylightThreshold	Float	0.2
EnableNoiseEstimation	Bool	false

Parameter name	Data type	Default value
DistNoiseThreshold	Float	0
CompensateAmbientLightDrift	Bool	false

Tab. 58: Default values of common imager config parameters

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
AbsDistStraylightThreshold	0	1
RelAmpStraylightThreshold	0	1
DistNoiseThreshold	0	100

Tab. 59: Minimum and maximum values of common imager config parameters

Parameters only in double exposure modes of application imager config

Parameter name	Data type	Description
ExposureTimeRatio	Double	Ratio of long exposure time to short exposure time. Only available in double exposure modes. The parameter has limits and is listed in the reply of the getAllParameterLimits method.

Tab. 60: Parameters only in double exposure modes of application imager config

Default values of double exposure mode parameters

Parameter name	Data type	Default value
ExposureTimeRatio	Double	40

Tab. 61: Default values of double exposure mode parameters

Minimum and maximum values of double exposure mode parameters

Parameter name	Minimum value	Maximum value
ExposureTimeRatio	2	50

Tab. 62: Minimum and maximum values of double exposure mode parameters

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

Parameter name	Data type	Description
ExposureTimeGrayscale	Int	Exposure time in microseconds for the grayscale image. Only available in exposure modes with grayscale image. The parameter has limits and is listed in the reply of the getAllParameterLimits method.

Tab. 63: Parameters only in exposure modes with grayscale image of application imager config

Default values of exposure mode parameters with grayscale image

Parameter name	Data type	Default value
ExposureTimeGrayscale	Int	1000

Tab. 64: Default values of exposure mode parameters with grayscale image

Minimum and maximum values of exposure mode parameters with grayscale image

Parameter name	Minimum value	Maximum value
ExposureTimeGrayscale	1	5000

Tab. 65: Minimum and maximum values exposure mode parameters with grayscale image

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

Capture mode

The capture mode consists of:

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

Tab. 66: Capture mode

Table index

Tab. 1	Protocol structure	10
Tab. 2	Command description	10
Tab. 3	Receiving images	10
Tab. 4	Chunk type	11
Tab. 5	Available chunk types	11
Tab. 6	Pixel format	13
Tab. 7	Additional information for CONFIDENCE_IMAGE	13
Tab. 8	setParameter	14
Tab. 9	getParameter	14
Tab. 10	getAllParameters	14
Tab. 11	getParameterLimits	14
Tab. 12	Parameter string encoding	15
Tab. 13	getParameter	15
Tab. 14	getAllParameters	15
Tab. 15	getSWVersion	15
Tab. 16	getHWInfo	16
Tab. 17	getApplicationList	16
Tab. 18	requestSession	16
Tab. 19	reboot	16
Tab. 20	systemCommand	16
Tab. 21	getTraceLogs	17
Tab. 22	getClientCompatibilityList	17
Tab. 23	getUnitVectors	17
Tab. 24	Trigger	17
Tab. 25	heartbeat	17
Tab. 26	cancelSession	18
Tab. 27	exportConfig	18
Tab. 28	importConfig	18
Tab. 29	exportApplication	18
Tab. 30	importApplication	18
Tab. 31	setOperatingMode	19
Tab. 32	factoryReset	19
Tab. 33	changeNameAndDescription	19
Tab. 34	activatePassword	20
Tab. 35	disablePassword	20
Tab. 36	save	20
Tab. 37	Parameter access	20
Tab. 38	Default values of device configuration parameters	21
Tab. 39	Minimum and maximum values of Device Config parameters	21
Tab. 40	saveAndActivateConfig	21
Tab. 41	Methods for parameter access	22
Tab. 42	Default values of Device/Network Config parameters	22
Tab. 43	setCurrentTime	22

Tab. 44	saveAndActivateConfig	22
Tab. 45	Parameters of Time config	23
Tab. 46	Default values of Time config	23
Tab. 47	Minimum and maximum values of Time Config parameters	23
Tab. 48	save	23
Tab. 49	validate	23
Tab. 50	forceTrigger	24
Tab. 51	discardUnsavedChanges	24
Tab. 52	Parameters of application	24
Tab. 53	Default values of application parameters	25
Tab. 54	minimum and maximum values of application parameters	25
Tab. 55	changeType	25
Tab. 56	availableTypes	25
Tab. 57	Parameters of all types of application imager config	25
Tab. 58	Default values of common imager config parameters	26
Tab. 59	Minimum and maximum values of common imager config parameters	27
Tab. 60	Parameters only in double exposure modes of application imager config	27
Tab. 61	Default values of double exposure mode parameters	27
Tab. 62	Minimum and maximum values of double exposure mode parameters	27
Tab. 63	Parameters only in exposure modes with grayscale image of application imager config	27
Tab. 64	Default values of exposure mode parameters with grayscale image	28
Tab. 65	Minimum and maximum values exposure mode parameters with grayscale image ..	28
Tab. 66	Capture mode	28