# RF627 Smart

# Contents

# Service protocol

# Profile Format v1

https://docs.google.com/spreadsheets/d/1prJy1ptW2sf4R1WGm6t-sBOw8nv3BVj3i5KBPV6J-ng/edit#gid=199004804

# WebAPI

By simple WebAPI, the user can get information about the device, read or write the parameter value. In addition, the device can run some commands via WebAPI. For a complete list of commands supported by this access, see the command description. The structure of the returned responses is presented in the command description section. The WebAPI samples use the factory IP address of the device and are presented as they should be typed in the browser address bar. If the IP address of the device has been changed, then the current IP address of the device should be used.

## General Device Information

- **/hello** - getting general information on the device in JSON format.
    - GET:
        - 192.168.1.30/hello
- **/api/v1/config/commands** - getting the list of commands supported by the device. The formalized description will contain the command name, Web API accessibility, the command ID, and access mode.
    - GET:
        - 192.168.1.30/api/v1/config/commands
- **/api/v1/config/returnCodes** - getting a textual description of operation results codes and errors returned by the device.
    - GET:
        - 192.168.1.30/api/v1/config/returnCodes

## Reading and Writing Device parameters

- **/api/v1/config/params** - getting general information about all device parameters in JSON format. The formalized description of the parameter will contain its name, type, access mode, index in the parameter array, offset for binary data, parameter data size, current value, default value, minimum and maximum values, parameter value step, maximum number of elements - for arrays.
    - GET:
        - 192.168.1.30/api/v1/config/params
- **/api/v1/config/params/values** - reading and writing device parameters. For reading, you can request specific parameters by name or index. To write a parameter, it is necessary to form a “PUT” request with the following parameters “parameter_name:value”.
    - GET:
        - 192.168.1.30/api/v1/config/params/values
        - 192.168.1.30/api/v1/config/params/values?name=fact_general_hardwareVer&index=120
    - PUT:
        - 192.168.1.30/api/v1/config/params/values?user_sensor_framerate=100&user_sensor_exposure1=100000

## Saving and Restoring Settings, Rebooting the Device

- **/api/v1/config/params/save** - saving the current values of the device parameters in the non-volatile memory of the device in the user area. The saved values will be used the next time the device is turned on.
    - GET:
        - 192.168.1.30/api/v1/config/params/save
- **/api/v1/config/params/restore/save** - saving the current values of the device parameters in the recovery area. These parameters will be applied if the parameters from the user area are damaged.
    - GET:
        - 192.168.1.30/api/v1/config/params/restore/save
- **/api/v1/config/params/restore/load** - loading device parameters from the recovery area. The loaded values will be written to the user area, the device will automatically reboot.
    - GET:
        - 192.168.1.30/api/v1/config/params/restore/load
- **/api/v1/reboot** - reboot the device. The parameters will be loaded from the user area (if they are not damaged).
    - GET:
        - 192.168.1.30/api/v1/reboot

## Getting Information from the Device Log File

- **/api/v1/log** - getting a log file of the device operation with a full description of the records.
    - GET:
        - 192.168.1.30/api/v1/log
- **/api/v1/log/content** - getting a log file of the device operation in a shortened, easy-to-read form.
    - GET:
        - 192.168.1.30/api/v1/log/content

## Authorization

- **/api/v1/authorization** - authorization as a manufacturer allows to edit the factory parameters of the device. Using the “GET” request, you need to get a token for which a key must be generated. The key must be sent to the device in the “PUT” request.
    - GET:
        - 192.168.1.30/api/v1/authorization
    - PUT:
        - 192.168.1.30/api/v1/authorization?key=230d84e16c0dae529098f1f1bb4debb3a6db3c870c4699245e651c06b714deb35a4d0a43a99f5ea0cc771a0e189c190a

## Profile Request

- **/api/v1/profile/capture** - request for making measurements (obtaining a profile). It is available only in “Software, external” and “Software, internal” modes.
    - GET:
        - 192.168.1.30/api/v1/profile/capture - request for 1 measurement;
        - 192.168.1.30/api/v1/profile/capture?count=100 - request for 100 measurements.

## Smart

- **/api/v1/smart/description** - getting a description of block groups, data types of the “Smart” module and an array of blocks implemented in this firmware.
    - GET:
        - 192.168.1.30/api/v1/smart/description
- **/api/v1/smart/graph/results** - getting the results of the graph blocks operation and the profile on which the calculation was performed.
    - GET:
        - 192.168.1.30/api/v1/smart/graph/results
- **/api/v1/smart/block/read** - getting a list of graph blocks with their parameters.
    - GET:
        - 192.168.1.30/api/v1/smart/block/read

## Data Exchange with the Periphery

- **/api/v1/periphery** - data transfer (sending and receiving) to and from the periphery.
    - GET:
        - 192.168.1.30/api/v1/periphery?interface=usart0&count=14&timeout=100000
    - PUT:
        - 192.168.1.30/api/v1/periphery?interface=usart0&payload=[20, 0x86]&wait_answer=true&answer_timeout=100000

## Calibration File

- **/api/v1/calibration/info** - requesting information about the device calibration file.
    - GET:
        - 192.168.1.30/api/v1/calibration/info

# Device Parameters

## Description of Device Parameters

All device parameters are divided into two groups: “user” and “factory”. Most of the user parameters are available for writing by the user; changing them will not lead to a failure in the device operation. The parameters of the “factory” group are available for writing after authorization on the device, their writing may lead to incorrect operation or failure in the device operation.

The values of all device parameters can be read using the service protocol or WebAPI. Parameters that cannot be written without the manufacturer's permission have the "awrite" access type. If the minimum value is not specified for a parameter, then the minimum value corresponds to the minimum value of the parameter type. If the maximum value is not specified, then it corresponds to the maximum value of the parameter type.

Each device parameter is described by the following elements:

- Access - determines the parameter accessibility for reading and writing operations, possible values:
    - read - the parameter is read-only (writing is physically impossible or impractical);
    - write - the parameter is available for both reading and writing by the user;
    - awrite – the parameter is readable, the parameter can be edited (written) only after authorization as “manufacturer”.
- Type - parameter data type, possible values:
    - uint32_t - 32-bit unsigned integer;
    - uint64_t - 64-bit unsigned integer;
    - int32_t - 32-bit signed integer;
    - uint64_t - 64-bit signed integer;
    - float_t - 32-bit floating-point number;
    - double_t - 64-bit floating-point number;
    - u32_arr_t - array of numbers of type “uint32_t”;
    - u64_arr_t - array of numbers of type “uint64_t”;
    - i32_arr_t - array of numbers of type “int32_t”;
    - i64_arr_t - array of numbers of type “int64_t”;
    - flt_array_t - array of numbers of type “float_t”;
    - dbl_array_t - array of numbers of type “double_t”;
    - string_t - string ending with 0x00, maximum string length is specified in the parameter description.
- Min value - minimum value of the parameter, input of value less than this value is not allowed. If the minimum value is not specified, then it is determined by the type of the parameter.
- Max value - maximum value of the parameter, input of value greater than this value is not allowed. If the maximum value is not specified, then it is determined by the type of the parameter.
- Step - step by which the parameter value is allowed to be changed. Values which don’t match the step will not be set. If the step is not specified, then any parameter change is allowed.

- Enum - enumeration of valid parameter values. Values which don't match the enumeration will not be set.
- Default value - default value of the parameter, set by the manufacturer or after turning on the device (depending on the parameter).
- Units - units of measurement of the parameter value; if not specified - it is not applicable to the parameter.

# Parameter Group "User"

## Section "General Parameters"

- **user_general_deviceState** - current state of the device - combination of enumeration values. The device changes this setting when initializing equipment, transmitting important data over the network (such as firmware), updating the firmware, and in other cases. In all modes, except DEV_STATE_NORMAL, the device can stop transferring profiles and other data not related to the current mode of operation.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_general_deviceState",
        "type": "uint32_t",
        "access": "read_only",
        "index": 48,
        "offset": 716,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 524288,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
```

```
                "value": 0,
                "key": "DEV_STATE_NORMAL",
                "label": "Normal operation"
        },
        {
                "value": 1,
                "key": "DEV_STATE_CALIB_FILE_RCV",
                "label": "Receiving of the calibration file"
        },
        {
                "value": 2,
                "key": "DEV_STATE_CALIB_FILE_SND",
                "label": "Sending of the calibration file"
        },
        {
                "value": 4,
                "key": "DEV_STATE_CALIB_FILE_SAVE",
                "label": "Saving of the calibration file"
        },
        {
                "value": 8,
                "key": "DEV_STATE_FLASH_ERASE",
                "label": "Erasing of the flash memory"
        },
        {
                "value": 16,
                "key": "DEV_STATE_FLASH_SEC_RCV",
                "label": "Receiving of the flash sector data"
        },
        {
                "value": 32,
                "key": "DEV_STATE_FLASH_SEC_SND",
                "label": "Sending of the flash sector data"
        },
        {
                "value": 64,
                "key": "DEV_STATE_FLASH_SEC_SAVE",
                "label": "Saving of the flash sector"
        },
        {
                "value": 128,
                "key": "DEV_STATE_FIRMWARE_RCV",
                "label": "Receiving of the firmware data"
        },
        {
                "value": 256,
                "key": "DEV_STATE_FIRMWARE_SND",
                "label": "Sending of the firmware data"
        },
        {
                "value": 512,
                "key": "DEV_STATE_FIRMWARE_SAVE",
                "label": "Saving of the firmware"
        },
        {
                "value": 2048,
                "key": "DEV_STATE_DUMP_DOWNLOAD",
                "label": "Downloading dump data"
        },
        {
                "value": 4096,
                "key": "DEV_STATE_ETH_EXCESS",
                "label": "Required connection speed exceeds current"
```

```
                },
                {
                        "value": 524288,
                        "key": "DEV_STATE_HARDWARE_INIT",
                        "label": "Initializing of the device hardware"
                }
        ]
}
```

**Note:**

- DEV_STATE_NORMAL - device is operating normally (under normal operating conditions);
- DEV_STATE_CALIB_FILE_RCV - device receives calibration file;
- DEV_STATE_CALIB_FILE_SND - device sends calibration file;
- DEV_STATE_CALIB_FILE_SAVE - device saves calibration file to internal flash drive;
- DEV_STATE_FIRMWARE_RCV - device receives firmware;
- DEV_STATE_FIRMWARE_SND - device sends firmware;
- DEV_STATE_FIRMWARE_SAVE - device saves firmware file to internal flash drive;
- DEV_STATE_ETH_INIT - device initializes hardware and software for ethernet connection;
- DEV_STATE_DUMP_DOWNLOAD - device downloads data of profile dump;
- DEV_STATE_ETH_EXCESS - required connection speed exceeds the current value for the ethernet connection;
- DEV_STATE_HARDWARE_INIT - device initializes hardware.

- **user_general_deviceName** – user scanner name. It appears on the scanner web page and can be used for quick identification of scanners.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: String,
        defaultValue: String,
        maxLen: Number (uint32_t),
}
```

**JSON example**

```
{
        "name": "user_general_deviceName",
        "type": "string_t",
        "access": "write",
        "index": 49,
        "offset": 720,
        "size": 128,
```

```
        "value": "2D laser scanner",
        "maxLen": 128,
        "defaultValue": "2D laser scanner"
}
```

- **user_general_logSaveEnabled** - permission to save a log file automatically after booting the device and after important events. When this option is enabled, it slightly (~ 100 ms) increases boot time of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_general_logSaveEnabled",
        "type": "uint32_t",
        "access": "write",
        "index": 50,
        "offset": 848,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_general_logSize** - current size (number of records) of the device log file.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_general_logSize",
        "type": "uint32_t",
        "access": "read_only",
        "index": 51,
        "offset": 852,
        "size": 4,
        "value": 141,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0
}
```

## Section “System Monitor”

- **user_sysMon_fpgaTemp** - current FPGA temperature (internal processor module) of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_fpgaTemp",
        "type": "float_t",
```

```
        "access": "read_only",
        "index": 52,
        "offset": 856,
        "size": 4,
        "value": 51.9866943359375,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```


- **user_sysMon_paramsChanged** - device settings have been changed but not saved.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
}
```

**JSON example**


```
{
        "name": "user_sysMon_paramsChanged",
        "type": "uint32_t",
        "access": "read_only",
        "index": 53,
        "offset": 860,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 7,
        "step": 0,
        "defaultValue": 0
}
```


- **user_sysMon_tempSens00** - current temperature inside the housing of the device, measured by the sensor with address 00.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
```

```
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens00",
        "type": "float_t",
        "access": "read_only",
        "index": 54,
        "offset": 864,
        "size": 4,
        "value": 32.75,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens00Max** - maximum temperature recorded by the sensor with address 00.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens00Max",
        "type": "float_t",
        "access": "read_only",
        "index": 55,
        "offset": 868,
        "size": 4,
        "value": 32.9,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens00Min** - minimum temperature recorded by the sensor with address 00.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens00Min",
        "type": "float_t",
        "access": "read_only",
        "index": 56,
        "offset": 872,
        "size": 4,
        "value": 0.9,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens01** - current temperature inside the housing of the device, measured by the sensor with address 01.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens01",
```

```
        "type": "float_t",
        "access": "read_only",
        "index": 57,
        "offset": 876,
        "size": 4,
        "value": -100,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```


- **user_sysMon_tempSens01Max** - maximum temperature recorded by the sensor with address 01.

**Object description**


```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```


**JSON example**


```
{
        "name": "user_sysMon_tempSens01Max",
        "type": "float_t",
        "access": "read_only",
        "index": 58,
        "offset": 880,
        "size": 4,
        "value": 0,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```


- **user_sysMon_tempSens01Min** - minimum temperature recorded by the sensor with address 01.

**Object description**


```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
```

```
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens01Min",
        "type": "float_t",
        "access": "read_only",
        "index": 59,
        "offset": 884,
        "size": 4,
        "value": -100,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens10** - current temperature inside the housing of the device, measured by the sensor with address 10.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens10",
        "type": "float_t",
        "access": "read_only",
        "index": 60,
        "offset": 888,
        "size": 4,
        "value": -100,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
```

```
        "units": "°C"
}
```

- **user_sysMon_tempSens10Max** - maximum temperature recorded by the sensor with address 10.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens10Max",
        "type": "float_t",
        "access": "read_only",
        "index": 61,
        "offset": 892,
        "size": 4,
        "value": 0,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens10Min** - minimum temperature recorded by the sensor with address 10.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens10Min",
        "type": "float_t",
        "access": "read_only",
        "index": 62,
        "offset": 896,
        "size": 4,
        "value": -100,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens11** - current temperature inside the housing of the device, measured by the sensor with address 11.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens11",
        "type": "float_t",
        "access": "read_only",
        "index": 63,
        "offset": 900,
        "size": 4,
        "value": -100,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens11Max** - maximum temperature recorded by the sensor with address 11.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens11Max",
        "type": "float_t",
        "access": "read_only",
        "index": 64,
        "offset": 904,
        "size": 4,
        "value": 0,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```

- **user_sysMon_tempSens11Min** - minimum temperature recorded by the sensor with address 11.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sysMon_tempSens11Min",
        "type": "float_t",
        "access": "read_only",
        "index": 65,
        "offset": 908,
        "size": 4,
        "value": -100,
```

```
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```


- **user_sysMon_cmosSensorTemp** - current temperature of CMOS sensor (the main one, if there are several of them in the system).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_sysMon_cmosSensorTemp",
        "type": "float_t",
        "access": "read_only",
        "index": 65,
        "offset": 908,
        "size": 4,
        "value": -100,
        "min": -100,
        "max": 150,
        "step": 0,
        "defaultValue": 0,
        "units": "°C"
}
```


## Section “Sensor”

- **user_sensor_syncSource** - source of synchronization of measurements.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
```

```
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_sensor_syncSource",
        "type": "uint32_t",
        "access": "write",
        "index": 66,
        "offset": 912,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 3,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "SYNC_INTERNAL",
                        "label": "Internal"
                },
                {
                        "value": 1,
                        "key": "SYNC_EXTERNAL",
                        "label": "External"
                },
                {
                        "value": 2,
                        "key": "SYNC_SOFTWARE_EXT",
                        "label": "Software, external"
                },
                {
                        "value": 3,
                        "key": "SYNC_SOFTWARE",
                        "label": "Software, internal"
                }
        ]
}
```

**NOTE:**

- SYNC_INTERNAL - starting measurements with the internal generator of the device;
- SYNC_EXTERNAL - starting measurements with the external source (trigger - described in a separate section);
- SYNC_SOFTWARE_EXT - software request for measurements via service protocol (a separate command), the start of measurements is synchronized with external source (trigger - described in a separate section);

- SYNC_SOFTWARE - starting measurements by software request via service protocol (separate command).

- **user_sensor_framerate** - frame rate of CMOS sensor, which sets the measurement frequency. The value to be written must not exceed the value of the parameter “user_sensor_maxFramerate”.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sensor_framerate",
        "type": "uint32_t",
        "access": "write",
        "index": 67,
        "offset": 916,
        "size": 4,
        "value": 490,
        "min": 1,
        "max": 100000,
        "step": 0,
        "defaultValue": 490,
        "units": "Hz"
}
```

- **user_sensor_maxFramerate** - maximum possible frame rate (measurement frequency) for the current operating mode.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
```

```
}
```

**JSON example**

```
{
        "name": "user_sensor_maxFramerate",
        "type": "uint32_t",
        "access": "write",
        "index": 68,
        "offset": 920,
        "size": 4,
        "value": 490,
        "min": 1,
        "max": 100000,
        "step": 0,
        "defaultValue": 490,
        "units": "Hz"
}
```

- **user_sensor_exposureControl** - control method of sensor exposure.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_sensor_exposureControl",
        "type": "uint32_t",
        "access": "write",
        "index": 69,
        "offset": 924,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 3,
        "step": 0,
        "defaultValue": 1,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "EXPOSE_AUTO",
```

```
                        "label": "Auto"
                },
                {
                        "value": 1,
                        "key": "EXPOSE_FIXED",
                        "label": "Fixed"
                },
                {
                        "value": 2,
                        "key": "EXPOSE_ADJUST",
                        "label": "Adjust"
                },
                {
                        "value": 3,
                        "key": "EXPOSE_MULTI_2",
                        "label": "2 exposures"
                },
                {
                        "value": 4,
                        "key": "EXPOSE_MULTI_3",
                        "label": "3 exposures"
                }
        ]
}
```


**NOTE:**

- EXPOSE_AUTO - automatic exposure control based on profile analysis;
- EXPOSE_FIXED - exposure time is set by the user;
- EXPOSE_ADJUST - exposure is automatically selected by the device, when the value “TRUE” is assigned to the parameter “user_sensor_exposureAdjust”, the value of this parameter will be automatically changed to “FALSE” after the end of the selection;
- EXPOSE_MULTI_2 - mode with 2 exposures, used to obtain a profile of surfaces with different types of reflection;
- EXPOSE_MULTI_3 - mode with 3 exposures, used to obtain a profile of surfaces with different types of reflection;

- **user_sensor_exposure1** - exposure time in the modes EXPOSE_AUTO and EXPOSE_FIXED.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_sensor_exposure1",
        "type": "uint32_t",
        "access": "write",
        "index": 70,
        "offset": 928,
        "size": 4,
        "value": 63900,
        "min": 3000,
        "max": 300000000,
        "step": 100,
        "defaultValue": 300000,
        "units": "ns"
}
```


- **user_sensor_exposure2** - exposure time №2 in the mode EXPOSE_MULTI_2.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_sensor_exposure2",
        "type": "uint32_t",
        "access": "write",
        "index": 71,
        "offset": 932,
        "size": 4,
        "value": 63900,
        "min": 3000,
        "max": 300000000,
        "step": 100,
        "defaultValue": 300000,
        "units": "ns"
}
```


- **user_sensor_exposure3** - exposure time №3 in the mode EXPOSE_MULTI_3.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_sensor_exposure3",
        "type": "uint32_t",
        "access": "write",
        "index": 72,
        "offset": 936,
        "size": 4,
        "value": 63900,
        "min": 3000,
        "max": 300000000,
        "step": 100,
        "defaultValue": 300000,
        "units": "ns"
}
```

- **user_sensor_exposureAdjust** - start of automatic selection of exposure in the mode “Adjust”, which is performed when the value “TRUE” is assigned to this parameter; the value of this parameter will be automatically changed to “FALSE” after the end of the selection.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_sensor_exposureAdjust",
        "type": "uint32_t",
        "access": "write",
        "index": 144,
        "offset": 1404,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```


- **user_sensor_maxExposure** - maximum exposure time in the current operating mode of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_sensor_maxExposure",
        "type": "uint32_t",
        "access": "read_only",
        "index": 73,
        "offset": 940,
        "size": 4,
        "value": 70000,
        "min": 3000,
        "max": 3000000000,
        "step": 0,
        "defaultValue": 1443298,
        "units": "ns"
```

```
}
```

- **user_sensor_defectivePixels** - array of coordinates [X1, Y1, X2, Y2, ... X15, Y15] of defective pixels of sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                …
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                …
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_sensor_defectivePixels",
        "type": "u32_arr_t",
        "access": "write",
        "index": 74,
        "offset": 944,
        "size": 128,
        "value": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        "defaultValue": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        "min": 0,
        "max": 4096,
        "step": 0,
        "maxCount": 32
}
```

- **user_sensor_doubleSpeedEnabled** – enabling/disabling the double frame rate mode. Enabling this mode allows to almost double the measurement frequency (profiles per second) due to reducing the accuracy along the Z axis.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_sensor_doubleSpeedEnabled",
        "type": "uint32_t",
        "access": "write",
        "index": 144,
        "offset": 1404,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_sensor_edrType** - enabling/disabling extended dynamic range. Allows to get a high-quality profile of both light and dark surfaces.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
```

```
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_sensor_edrType",
        "type": "uint32_t",
        "access": "write",
        "index": 145,
        "offset": 1408,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 2,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "EDR_DISABLE",
                        "label": "EDR disabled"
                },
                {
                        "value": 1,
                        "key": "EDR_COLUMN",
                        "label": "Column EDR"
                },
                {
                        "value": 2,
                        "key": "EDR_PIECEWISE",
                        "label": "Piecewise linear EDR"
                }
        ]
}
```

**NOTE:**

- EDR_DISABLE - extended dynamic range mode is disabled;
- EDR_COLUMN - different exposure modes for odd and even columns;
- EDR_PIECEWISE - the mode with piecewise linear response of the sensor.

- **user_sensor_edrColumnDivider** - exposure time divider for odd columns. This parameter controls the sensitivity to very bright areas of the profile in extended dynamic range mode EDR_COLUMN.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_sensor_edrColumnDivider",
        "type": "uint32_t",
        "access": "write",
        "index": 146,
        "offset": 1412,
        "size": 4,
        "value": 2,
        "min": 2,
        "max": 32,
        "step": 0,
        "defaultValue": 2
}
```

## Section “Region of Interest”

- **user_roi_enabled** - enabling/disabling the “region of interest” mode. This mode allows to increase frame rate of the sensor and hence the measurement rate.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_roi_enabled",
        "type": "uint32_t",
        "access": "write",
        "index": 75,
        "offset": 1072,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```


- **user_roi_active** - indicates the state of the “region of interest” mode during automatic positioning of the measurement area. In auto mode, if a profile is not found, the parameter value becomes FALSE, and when a profile is found, the parameter value automatically becomes TRUE. In manual mode, this parameter is always TRUE.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_roi_enabled",
        "type": "uint32_t",
```

```
        "access": "write",
        "index": 75,
        "offset": 1072,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```


- **user_roi_mode** - the mode of control of the measurement area position.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_roi_posMode",
        "type": "uint32_t",
        "access": "write",
        "index": 77,
        "offset": 1080,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
```

```
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "ROI_POSITION_MANUAL",
                        "label": "Manual"
                },
                {
                        "value": 1,
                        "key": "ROI_POSITION_AUTO",
                        "label": "Auto"
                },
                {
                        "value": 2,
                        "key": "ROI_MODE_AUTOSCAN",
                        "label": "Auto-scan"
                }       ]
}
```


**NOTE:**

- ROI_POSITION_MANUAL - the position is set by the user;
- ROI_POSITION_AUTO - automatic position control with keeping the profile in the center. When the profile is lost, the scanner switches to the operating mode without the region of interest (work in the entire range, the frame rate is decreased to standard), when the profile is found, it automatically switches to the region of interest with the increase of the frame rate.
- ROI_MODE_AUTOSCAN - automatic position control with keeping the profile in the center. When a profile is lost, the scanner switches to the mode of scanning the working range by the region of interest (the frame rate is not decreased), when the profile is found, it automatically switches to keeping the profile within the region of interest.

- **user_roi_pos** - current position of the upper boundary of “the region of interest” in lines of sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_roi_pos",
```

```
        "type": "uint32_t",
        "access": "write",
        "index": 78,
        "offset": 1084,
        "size": 4,
        "value": 208,
        "min": 0,
        "max": 1280,
        "step": 0,
        "defaultValue": 100,
        "units": "lines"
}
```


- **user_roi_maxPos** - maximum position of the upper boundary of “the region of interest” in the current operating mode of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_roi_maxPos",
        "type": "uint32_t",
        "access": "read_only",
        "index": 79,
        "offset": 1088,
        "size": 4,
        "value": 416,
        "min": 0,
        "max": 1280,
        "step": 0,
        "defaultValue": 1180,
        "units": "lines"
}
```


- **user_roi_size** - determines the size of “the region of interest” in which the profile is searched and processed.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
```

```
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_roi_size",
        "type": "uint32_t",
        "access": "write",
        "index": 148,
        "offset": 1480,
        "size": 4,
        "value": 64,
        "min": 0,
        "max": 488,
        "step": 8,
        "defaultValue": 64,
        "units": "lines"
}
```

- **user_roi_reqProfSize** - minimum required number of profile points to enable “the region of interest" in the mode ROI_POSITION_AUTO.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_roi_reqProfSize",
        "type": "uint32_t",
        "access": "write",
        "index": 80,
        "offset": 1092,
        "size": 4,
        "value": 320,
        "min": 0,
        "max": 1280,
        "step": 64,
        "defaultValue": 320,
```

```
        "units": "points"
}
```

- **user_roi_zsmr** - the position of the upper boundary of the region in which the profile is searched and processed.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_roi_zsmr",
        "type": "float_t",
        "access": "read_only",
        "index": 81,
        "offset": 1096,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 0,
        "units": "mm"
}
```

- **user_roi_zemr** - the position of the lower boundary of the region in which the profile is searched and processed.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_roi_zemr",
        "type": "float_t",
        "access": "read_only",
        "index": 82,
        "offset": 1100,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 0,
        "units": "mm"
}
```


## Section “Ethernet network”

- **user_network_speed** - current speed of Ethernet connection. The connection speed is changed according to the value of this parameter. If autonegotiation of connection speed is enabled, the value of this parameter is ignored.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_network_speed",
        "type": "uint32_t",
        "access": "write",
        "index": 83,
        "offset": 1104,
        "size": 4,
        "value": 1000,
        "min": 10,
        "max": 1000,
        "step": 0,
```

```
        "defaultValue": 1000,
        "valuesEnum": [
                {
                        "value": 10,
                        "key": "LINK_SPEED_10MBIT",
                        "label": "10"
                },
                {
                        "value": 100,
                        "key": "LINK_SPEED_100MBIT",
                        "label": "100"
                },
                {
                        "value": 1000,
                        "key": "LINK_SPEED_1GBIT",
                        "label": "1000"
                }
        ],
        "units": "Mbps"
}
```


**NOTE:**

- LINK_SPEED_10MBIT - the connection speed is 10 Mbps, currently not used;
- LINK_SPEED_100MBIT - the connection speed is 100 Mbps;
- LINK_SPEED_1GBIT - the connection speed is 1000 Mbps.

- **user_network_requiredSpeed** - the required speed of Ethernet connection in the current operating mode of the device. Depends on the number of profiles per second, the number of points in the profile, etc.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_network_requiredSpeed",
        "type": "uint32_t",
        "access": "read_only",
        "index": 84,
        "offset": 1108,
        "size": 4,
        "value": 80,
```

```
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 1,
        "units": "Mbps"
}
```

- **user_network_autoNeg** – enabling/disabling autonegotiation of Ethernet connection speed.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_network_autoNeg",
        "type": "uint32_t",
        "access": "write",
        "index": 85,
        "offset": 1112,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 1,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_network_ip** – IP address of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_network_ip",
        "type": "u32_arr_t",
        "access": "write",
        "index": 86,
        "offset": 1116,
        "size": 16,
        "value": [
                192,
                168,
                1,
                30
        ],
        "defaultValue": [
                192,
                168,
                1,
                30
        ],
        "min": 0,
        "max": 255,
        "step": 0,
        "maxCount": 4
}
```

- **user_network_mask** - subnet mask for the device.

**Object description**

```
{
        name: String,
```

```
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_network_mask",
        "type": "u32_arr_t",
        "access": "write",
        "index": 87,
        "offset": 1132,
        "size": 16,
        "value": [
                255,
                255,
                255,
                0
        ],
        "defaultValue": [
                255,
                255,
                255,
                0
        ],
        "min": 0,
        "max": 255,
        "step": 0,
        "maxCount": 4
}
```

- **user_network_gateway** - gateway address.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
```

```
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_network_gateway",
        "type": "u32_arr_t",
        "access": "write",
        "index": 88,
        "offset": 1148,
        "size": 16,
        "value": [
                192,
                168,
                1,
                1
        ],
        "defaultValue": [
                192,
                168,
                1,
                1
        ],
        "min": 0,
        "max": 255,
        "step": 0,
        "maxCount": 4
}
```

- **user_network_hostIP** - IP address of the device receiving profiles and calculation results via UDP.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
```

```
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_network_hostIP",
        "type": "u32_arr_t",
        "access": "write",
        "index": 89,
        "offset": 1164,
        "size": 16,
        "value": [
                192,
                168,
                1,
                2
        ],
        "defaultValue": [
                192,
                168,
                1,
                2
        ],
        "min": 0,
        "max": 255,
        "step": 0,
        "maxCount": 4
}
```

- **user_network_hostPort** - port number of the device receiving profiles and calculation results via UDP.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_network_hostPort",
        "type": "uint32_t",
        "access": "write",
        "index": 90,
        "offset": 1180,
        "size": 4,
        "value": 50001,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 50001
}
```


- **user_network_webPort** - port number to access the web page.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "user_network_webPort",
        "type": "uint32_t",
        "access": "write",
        "index": 91,
        "offset": 1184,
        "size": 4,
        "value": 80,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 80
}
```


- **user_network_servicePort** - port number for the service protocol.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
```

```
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_network_servicePort",
        "type": "uint32_t",
        "access": "write",
        "index": 92,
        "offset": 1188,
        "size": 4,
        "value": 50011,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 50011
}
```

## Section “Data Stream”

- **user_streams_udpEnabled** - enabling/disabling the UDP data stream (sending to IP address specified by the parameter “user_network_hostIP” and to the port specified by the parameter “user_network_hostPort”).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_streams_udpEnabled",
        "type": "uint32_t",
        "access": "write",
        "index": 88,
        "offset": 1112,
        "size": 4,
```

```
        "value": 1,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```


- **user_streams_format** - data transfer formats.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_streams_format",
        "type": "uint32_t",
        "access": "write",
        "index": 89,
        "offset": 1116,
        "size": 4,
        "value": 17,
        "min": 16,
        "max": 17,
        "step": 0,
        "defaultValue": 17,
        "valuesEnum": [
                {
                        "value": 16,
                        "key": "DATA_FORMAT_RAW_PROFILE",
```

```
                        "label": "RAW"
                },
                {
                        "value": 17,
                        "key": "DATA_FORMAT_PROFILE",
                        "label": "Profile"
                }
        ]
}
```


**NOTE:**

- DATA_FORMAT_RAW_PROFILE – the position of profile points is transferred as uncalibrated data, in subpixel values. The format is used for debugging, allows to visually match the profile and the image formed by the CMOS sensor;
- DATA_FORMAT_PROFILE - the value of the coordinates of profile points is transferred in millimeters, the main format of data transfer by the scanner.

- **user_streams_pointsCount** - the number of points of profile counted and transferred by the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_streams_pointsCount",
        "type": "uint32_t",
        "access": "write",
        "index": 147,
        "offset": 1416,
        "size": 4,
        "value": 648,
        "min": 648,
        "max": 1296,
        "step": 648,
        "defaultValue": 648,
        "units": "points"
}
```

- **user_streams_includeIntensity** – enabling/disabling transmission of the point brightness of the profile. The brightness values are transmitted after the profile data in the format 1 byte per point, 0 – black color... 255 – white color.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_streams_includeIntensity",
        "type": "uint32_t",
        "access": "write",
        "index": 90,
        "offset": 1120,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                "label": "true"
                }
        ]
}
```

- **user_streams_udpPacketsCounter** - internal counter for sending UDP packets with profiles. It can be used to control the loss of packets with profiles.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_streams_udpPacketsCounter",
        "type": "uint32_t",
        "access": "read_only",
        "index": 91,
        "offset": 1124,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0
}
```

## Section “Profile Processing”

- **user_processing_intensityClipping** - clipping threshold of the level of image signal received from the sensor. The vertical sliding window of the function has a size of 5 points. If the window has (at least one) brightness value above the threshold, then the brightness of the central pixel in the window does not change. If all the brightness values in the window are below the threshold, then the brightness value of the central pixel in the window is assigned 0 value. This allows to obtain a Gaussian distribution of the signal for bright pixels.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_processing_intensityClipping",
        "type": "uint32_t",
        "access": "write",
        "index": 92,
        "offset": 1128,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 100,
        "step": 1,
        "defaultValue": 1,
        "units": "%"
}
```


- **user_processing_threshold** - determines the profile point detection threshold. If this parameter has low values, a profile with a low brightness signal is detected, but this can lead to false detections on surfaces with glare and reflections. If this parameter has high values, a high brightness signal is required to detect the profile, but this allows to get a clear profile without artifacts.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_processing_threshold",
        "type": "uint32_t",
        "access": "write",
        "index": 92,
        "offset": 1128,
        "size": 4,
        "value": 2,
        "min": 0,
        "max": 100,
        "step": 1,
        "defaultValue": 2,
        "units": "%"
}
```


- **user_processing_profPerSec** - the number of processed profiles per second.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_processing_profPerSec",
        "type": "uint32_t",
        "access": "read_only",
        "index": 93,
        "offset": 1132,
        "size": 4,
        "value": 490,
        "min": 0,
        "max": 100000,
        "step": 0,
        "defaultValue": 490,
        "units": "pps"
}
```

- **user_processing_medianMode** - enabling and setting median filter width. The median filter removes random outliers and fills in the gaps in the profile up to half of the filter size.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_processing_medianMode",
        "type": "uint32_t",
        "access": "write",
        "index": 94,
        "offset": 1136,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 15,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "OFF",
                        "label": "Off"
                },
                {
                        "value": 3,
                        "key": "3",
                        "label": "3"
                },
                {
                        "value": 5,
                        "key": "5",
                        "label": "5"
                },
                {
                        "value": 7,
                        "key": "7",
                        "label": "7"
                },
                {
                        "value": 9,
                        "key": "9",
                        "label": "9"
                },
                {
                        "value": 11,
                        "key": "11",
                        "label": "11"
                },
                {
                        "value": 13,
                        "key": "13",
                        "label": "13"
                },
                {
                        "value": 15,
                        "key": "15",
                        "label": "15"
                }
        ]
}
```


**NOTE:**

- 0 - filter is disabled;
- 3 - filter is enabled, the size of the sliding window of the filter is 3 points;
- 5 - filter is enabled, the size of the sliding window of the filter is 5 points;

- 7 - filter is enabled, the size of the sliding window of the filter is 7 points;
- 9 - filter is enabled, the size of the sliding window of the filter is 9 points;
- 11 - filter is enabled, the size of the sliding window of the filter is 11 points;
- 13 - filter is enabled, the size of the sliding window of the filter is 13 points;
- 15 - filter is enabled, the size of the sliding window of the filter is 15 points.

- **user_processing_bilateralMode** - enabling and setting bilateral filter width. Bilateral filter allows to smooth the values of the profile points, while preserving its abrupt changes.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_processing_bilateralMode",
        "type": "uint32_t",
        "access": "write",
        "index": 95,
        "offset": 1140,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 15,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "OFF",
                        "label": "Off"
                },
                {
                        "value": 3,
                        "key": "3",
                        "label": "3"
                },
                {
```

```
                        "value": 5,
                        "key": "5",
                        "label": "5"
                },
                {
                        "value": 7,
                        "key": "7",
                        "label": "7"
                },
                {
                        "value": 9,
                        "key": "9",
                        "label": "9"
                },
                {
                        "value": 11,
                        "key": "11",
                        "label": "11"
                },
                {
                        "value": 13,
                        "key": "13",
                        "label": "13"
                },
                {
                        "value": 15,
                        "key": "15",
                        "label": "15"
                }
        ]
}
```


**NOTE:**

- 0 - filter is disabled;
- 3 - filter is enabled, the size of the sliding window of the filter is 3 points;
- 5 - filter is enabled, the size of the sliding window of the filter is 5 points;
- 7 - filter is enabled, the size of the sliding window of the filter is 7 points;
- 9 - filter is enabled, the size of the sliding window of the filter is 9 points;
- 11 - filter is enabled, the size of the sliding window of the filter is 11 points;
- 13 - filter is enabled, the size of the sliding window of the filter is 13 points;
- 15 - filter is enabled, the size of the sliding window of the filter is 15 points.

- **user_processing_peakMode** - peak selection mode for detecting the brightness peak in the column of the CMOS sensor to obtain the profile point. The mode avoids to ignore reflections and highlights.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
```

```
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_processing_peakMode",
        "type": "uint32_t",
        "access": "write",
        "index": 96,
        "offset": 1144,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 5,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "PEAK_MODE_INTENSITY",
                        "label": "Max intensity"
                },
                {
                        "value": 1,
                        "key": "PEAK_MODE_FIRST",
                        "label": "First"
                },
                {
                        "value": 2,
                        "key": "PEAK_MODE_LAST",
                        "label": "Last"
                },
                {
                        "value": 3,
                        "key": "PEAK_MODE_NUMBER_2",
                        "label": "Prefer #2"
                },
                {
                        "value": 4,
                        "key": "PEAK_MODE_NUMBER_3",
                        "label": "Prefer #3"
                },
                {
                        "value": 5,
                        "key": "PEAK_MODE_NUMBER_4",
                        "label": "Prefer #4"
                }
        ]
}
```

**NOTE:**

- PEAK_MODE_INTENSITY - the profile points are selected based on the maximum intensity value;
- PEAK_MODE_FIRST - the profile points are selected for the first peak of detection threshold;
- PEAK_MODE_LAST - the profile points are selected for the last peak of detection threshold;
- PEAK_MODE_NUMBER_2 - when selecting the profile points, the priority is given to peak No. 2;
- PEAK_MODE_NUMBER_3 - when selecting the profile points, the priority is given to peak No. 3;
- PEAK_MODE_NUMBER_4 - when selecting the profile points, the priority is given to peak No. 4.

- **user_processing_flip** - profile flip mode. Profile flip mode is only applied if value DATA_FORMAT_PROFILE is assigned to user_streams_format.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_processing_flip",
        "type": "uint32_t",
        "access": "write",
        "index": 97,
        "offset": 1148,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 3,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FLIP_MODE_OFF",
```

```
                        "label": "No"
                },
                {
                        "value": 1,
                        "key": "FLIP_MODE_X",
                        "label": "X"
                },
                {
                        "value": 2,
                        "key": "FLIP_MODE_Z",
                        "label": "Z"
                },
                {
                        "value": 3,
                        "key": "FLIP_MODE_XZ",
                        "label": "XZ"
                }
        ]
}
```

**NOTE:**

- FLIP_MODE_OFF – flip is disabled;
- FLIP_MODE_X - flip along the X axis of the scanner;
- FLIP_MODE_Z - flip along the Z axis of the scanner;
- FLIP_MODE_XZ - flip along both axes (X and Z).

## Section “Laser”

- **user_laser_enabled** – enabling/disabling laser.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_laser_enabled",
        "type": "uint32_t",
```

```
        "access": "write",
        "index": 98,
        "offset": 1152,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 1,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```


- **user_laser_value** - setting the brightness values of the laser.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_laser_value",
        "type": "uint32_t",
        "access": "write",
        "index": 99,
        "offset": 1156,
        "size": 4,
        "value": 50,
        "min": 0,
        "max": 100,
        "step": 5,
        "defaultValue": 50,
        "units": "%"
}
```

## Section “Synchronization by External Trigger”

- **user_trigger_sync_source** - selection of an input of synchronization signal or combination of inputs to start synchronization of measurements.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_trigger_sync_source",
        "type": "uint32_t",
        "access": "write",
        "index": 100,
        "offset": 1160,
        "size": 4,
        "value": 1,
        "min": 1,
        "max": 4,
        "step": 0,
        "defaultValue": 1,
        "valuesEnum": [
                {
                        "value": 1,
                        "key": "TRIG_SOURCE_IN1",
                        "label": "Input #1"
                },
                {
                        "value": 2,
                        "key": "TRIG_SOURCE_IN2",
                        "label": "Input #2"
                },
                {
                        "value": 3,
                        "key": "TRIG_SOURCE_IN1_OR_IN2",
                        "label": "Inputs #1 OR #2"
                },
                {
                        "value": 4,
                        "key": "TRIG_SOURCE_IN1_AND_IN2",
                        "label": "Inputs #1 AND #2"
                }
```

```
        ]
}
```

**NOTE:**

- TRIG_SOURCE_IN1 - synchronization by signal from input № 1;
- TRIG_SOURCE_IN2 - synchronization by signal from input № 2;
- TRIG_SOURCE_IN1_OR_IN2 - synchronization by signal from input № 1 or input № 2 (any of the signals from both inputs);
- TRIG_SOURCE_IN1_AND_IN2 - synchronization by signal from input № 1 and input № 2 (coincidence of signals on both inputs).

- **user_trigger_sync_strictEnabled** – enabling/disabling strict synchronization mode. When this mode is enabled, synchronization events that occurred during the exposure of a frame will be ignored and the next measurement will be started only by synchronization event when the sensor completes the exposure of the previous frame. In this case, if the synchronization event rate is slightly higher than the sensor's maximum frame rate, the number of profiles per second will be less than the maximum frame rate due to the stroboscopic effect. If this mode is disabled and synchronization events occurred during the exposure, the next measurement will start as soon as the sensor completes the exposure of the previous frame. In any situation, encoder counter value is latched at the beginning of the frame exposure.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_trigger_sync_strictEnabled",
        "type": "uint32_t",
        "access": "write",
        "index": 101,
        "offset": 1164,
        "size": 4,
```

```
        "value": 1,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 1,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```


- **user_trigger_sync_divider** - divider for synchronization events. Does not affect the encoder counter.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "user_trigger_sync_divider",
        "type": "uint32_t",
        "access": "write",
        "index": 102,
        "offset": 1168,
        "size": 4,
        "value": 1,
        "min": 1,
        "max": 8192,
        "step": 1,
        "defaultValue": 1
}
```


- **user_trigger_sync_delay** – synchronization delay: the value of delay of the beginning of measurement (the beginning of the frame exposure) relative to the synchronization event.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_trigger_sync_delay",
        "type": "uint32_t",
        "access": "write",
        "index": 103,
        "offset": 1172,
        "size": 4,
        "value": 700,
        "min": 700,
        "max": 100000000,
        "step": 100,
        "defaultValue": 700,
        "units": "ns"
}
```

- **user_trigger_sync_value** – the value of internal measurements counter, which counts the performed measurements.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_trigger_sync_value",
        "type": "uint32_t",
        "access": "write",
        "index": 104,
        "offset": 1176,
        "size": 4,
        "value": 12882,
```

```
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0
}
```

- **user_trigger_counter_type** - type of encoder counter - input pulse counter (internal pulse counter at the synchronization inputs).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_trigger_counter_type",
        "type": "uint32_t",
        "access": "write",
        "index": 105,
        "offset": 1180,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "TRIG_COUNTER_UNIDIR",
                        "label": "Unidirectional"
                },
                {
                        "value": 1,
                        "key": "TRIG_COUNTER_BIDIR",
                        "label": "Bidirectional"
                }
        ]
}
```

**NOTE:**

- TRIG_COUNTER_UNIDIR - unidirectional counter that does not take into account the phases of signals at inputs №1 and №2;
- TRIG_COUNTER_BIDIR - bidirectional counter that takes into account the phase of the signals at inputs №1 and №2 and can either increment or decrement.

- **user_trigger_counter_maxValue** - the maximum value of the encoder counter. The counter will be reset to 0 when exceeding this value.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_trigger_counter_maxValue",
        "type": "uint32_t",
        "access": "write",
        "index": 106,
        "offset": 1184,
        "size": 4,
        "value": 4294967295,
        "min": 1,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 4294967295
}
```

- **user_trigger_counter_resetTimerEnabled** – enabling/disabling the timer to automatically reset the encoder counter to value 0. If the timer is enabled and synchronization events doesn't occur within the period specified by the parameter "user_trigger_counter_resetTimerValue", the encoder counter will be reset.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
```

```
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_trigger_counter_resetTimerEnabled",
        "type": "uint32_t",
        "access": "write",
        "index": 107,
        "offset": 1188,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_trigger_counter_resetTimerValue** - timeout value - time interval before automatic reset of the encoder counter.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_trigger_counter_resetTimerValue",
        "type": "uint32_t",
        "access": "write",
        "index": 108,
        "offset": 1192,
        "size": 4,
        "value": 4294967295,
        "min": 1000,
        "max": 4294967295,
        "step": 1000,
        "defaultValue": 4294967295,
        "units": "ns"
}
```


- **user_trigger_counter_value** - current value of the encoder counter. This is an internal event counter at inputs №1 and №2.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "user_trigger_counter_value",
        "type": "uint32_t",
        "access": "write",
        "index": 109,
        "offset": 1196,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0
}
```


- **user_trigger_counter_dir** - ratio of signal phases at inputs №1 and №2. Determines the direction of movement when using a motion system.

**Object description**

```
{
        name: String,
        type: String,
```

```
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_trigger_counter_dir",
        "type": "uint32_t",
        "access": "read_only",
        "index": 110,
        "offset": 1200,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0
}
```

## Section “Physical Inputs”

- **user_inputs_physicalType** - switching the type of physical signal that the user applies to the inputs.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_inputs_physicalType",
        "type": "uint32_t",
```

```
        "access": "write",
        "index": 142,
        "offset": 1396,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 0,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "IN_PHYSICAL_RS485",
                        "label": "RS485"
                }
        ]
}
```


**NOTE:**

- IN_PHYSICAL_RS485 - physical levels comply with RS485/RS422 standard.

- **user_input1_enabled** – enabling/disabling input №1. If the input is disabled, all signals are ignored.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_input1_enabled",
        "type": "uint32_t",
        "access": "write",
        "index": 111,
        "offset": 1204,
        "size": 4,
        "value": 1,
        "min": 0,
```

```
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```


- **user_input1_mode** - operating mode of input №1. Determines what change or signal type is a synchronization event for this input.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_input1_mode",
        "type": "uint32_t",
        "access": "write",
        "index": 112,
        "offset": 1208,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 4,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "IN1_MODE_RISE_OR_FALL",
```

```
                        "label": "Rise or fall"
                },
                {
                        "value": 1,
                        "key": "IN1_MODE_RISE",
                        "label": "Rise"
                },
                {
                        "value": 2,
                        "key": "IN1_MODE_FALL",
                        "label": "Fall"
                },
                {
                        "value": 3,
                        "key": "IN1_MODE_LVL1",
                        "label": "High level"
                },
                {
                        "value": 4,
                        "key": "IN1_MODE_LVL0",
                        "label": "Low level"
                }
        ]
}
```


**NOTE:**

- IN1_MODE_RISE_OR_FALL - synchronization by rise or fall occurs when an input signal is switching from low level (logic 0) to high level (logic 1) (rising edge); as well as from high level to low level (falling edge);
- IN1_MODE_RISE - synchronization by rise occurs only when an input signal is switching from low level to high level (rising edge);
- IN1_MODE_FALL - synchronization by fall occurs only when an input signal is switching from high level to low level (falling edge);
- IN1_MODE_LVL1 - synchronization by high level at the input, measurements are started with the internal generator;
- IN1_MODE_LVL0 - synchronization by low level at the input, measurements are started with the internal generator.

- **user_input2_enabled** - enabling/disabling input №2. If the input is disabled, all signals are ignored.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
```

```
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_input2_enabled",
        "type": "uint32_t",
        "access": "write",
        "index": 113,
        "offset": 1212,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_input2_mode** - operating mode of input №2. Determines what change or signal type is a synchronization event for this input.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_input2_mode",
        "type": "uint32_t",
        "access": "write",
        "index": 114,
        "offset": 1216,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 4,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "IN2_MODE_RISE_OR_FALL",
                        "label": "Rise or fall"
                },
                {
                        "value": 1,
                        "key": "IN2_MODE_RISE",
                        "label": "Rise"
                },
                {
                        "value": 2,
                        "key": "IN2_MODE_FALL",
                        "label": "Fall"
                },
                {
                        "value": 3,
                        "key": "IN2_MODE_LVL1",
                        "label": "High level"
                },
                {
                        "value": 4,
                        "key": "IN2_MODE_LVL0",
                        "label": "Low level"
                }
        ]
}
```


**NOTE:**

- IN2_MODE_RISE_OR_FALL - synchronization by rise or fall occurs when an input signal is switching from low level (logic 0) to high level (logic 1) (rising edge); as well as from high level to low level (falling edge);
- IN2_MODE_RISE - synchronization by rise occurs only when an input signal is switching from low level to high level (rising edge);
- IN2_MODE_FALL - synchronization by fall occurs only when an input signal is switching from high level to low level (falling edge);
- IN2_MODE_LVL1 - synchronization by high level at the input, measurements are started with the internal generator;
- IN2_MODE_LVL0 - synchronization by low level at the input, measurements are started with the internal generator.

- **user_input3_enabled** - enabling/disabling input №3. This input is mainly used to reset the encoder counter.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_input3_enabled",
        "type": "uint32_t",
        "access": "write",
        "index": 115,
        "offset": 1220,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_input3_mode** - operating mode of input №3. Determines what change or signal type will reset the encoder counter.

**Object description**

```
{
        name: String,
```

```
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_input3_mode",
        "type": "uint32_t",
        "access": "write",
        "index": 116,
        "offset": 1224,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "IN3_MODE_RISE",
                        "label": "Rise"
                },
                {
                        "value": 1,
                        "key": "IN3_MODE_FALL",
                        "label": "Fall"
                }
        ]
}
```

**NOTE:**

- IN3_MODE_RISE - reset the encoder counter by rise of input pulse;
- IN3_MODE_FALL - reset the encoder counter by fall of input pulse.

- **user_input1_samples** - array of values of signals at input №1. The parameter is a time base of signals at input №1. Every 2 bits indicate the state of the signal at a certain point in time. Value 0b00 - low level of signal, 0b01 - state is changed (pulses), 0b10 - reserved, 0b11 - high level.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_input1_samples",
        "type": "u32_arr_t",
        "access": "read_only",
        "index": 117,
        "offset": 1228,
        "size": 24,
        "value": [
                4294967295,
                4294967295,
                4294967295,
                4294967295,
                4294967295,
                4294967295
        ],
        "defaultValue": [
                0,
                0,
                0,
                0,
                0,
                0
        ],
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "maxCount": 6
}
```

- **user_input2_samples** - array of values of signals at input №2. The parameter is a time base of signals at input №2.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_input2_samples",
        "type": "u32_arr_t",
        "access": "read_only",
        "index": 118,
        "offset": 1252,
        "size": 24,
        "value": [
                0,
                0,
                0,
                0,
                0,
                0
        ],
        "defaultValue": [
                0,
                0,
                0,
                0,
                0,
                0
        ],
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "maxCount": 6
}
```

- **user_input3_samples** - array of values of signals at input №3. The parameter is a time base of signals at input №3.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
                ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_input3_samples",
        "type": "u32_arr_t",
        "access": "read_only",
        "index": 119,
        "offset": 1276,
        "size": 24,
        "value": [
                0,
                0,
                0,
                0,
                0,
                0
        ],
        "defaultValue": [
                0,
                0,
                0,
                0,
                0,
                0
        ],
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "maxCount": 6
```

```
}
```

## Section “Physical Outputs”

- **user_outputs_physicalType** - switching the type of physical signal transmitted to the outputs of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_outputs_physicalType",
        "type": "uint32_t",
        "access": "write",
        "index": 143,
        "offset": 1400,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 0,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "OUT_PHYSICAL_RS485",
                        "label": "RS485"
                }
        ]
}
```

**NOTE:**

- OUT_PHYSICAL_RS485 - physical levels comply with RS485/RS422 standard.

- **user_output1_enabled** - enabling/disabling output №1. When disabled, the output has low level. When enabled, the signal is set by the parameters “user_output1_mode” and “user_output1_pulseWidth”.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_output1_enabled",
        "type": "uint32_t",
        "access": "write",
        "index": 120,
        "offset": 1300,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_output1_mode** - operating mode of output №1. Sets the type of output signal.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_output1_mode",
        "type": "uint32_t",
        "access": "write",
        "index": 121,
        "offset": 1304,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 5,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "OUT_MODE_EXPOSE_START",
                        "label": "Exposure start"
                },
                {
                        "value": 1,
                        "key": "OUT_MODE_EXPOSE_TIME",
                        "label": "Exposure time"
                },
                {
                        "value": 2,
                        "key": "OUT_MODE_IN1_REPEATER",
                        "label": "In1 repeater"
                },
                {
                        "value": 3,
                        "key": "OUT_MODE_IN2_REPEATER",
                        "label": "In2 repeater"
                },
                {
                        "value": 4,
                        "key": "OUT_MODE_IN3_REPEATER",
                        "label": "In3 repeater"
                },
                {
                        "value": 5,
                        "key": "OUT_MODE_IN3_REPEATER",
```

```
                        "label": "Smart"
                }
        ]
}
```

**NOTE:**

- OUT_MODE_EXPOSE_START - formation of the output pulse upon the start of frame exposure that triggers the measurement cycle;
- OUT_MODE_EXPOSE_TIME - formation of high level output signal during frame exposure for every measurement cycle;
- OUT_MODE_IN1_REPEATER - duplication of the signal from input №1 to the output whether the input is enabled or disabled;
- OUT_MODE_IN2_REPEATER - duplication of the signal from input №2 to the output whether the input is enabled or disabled;
- OUT_MODE_IN3_REPEATER - duplication of the signal from input №3 to the output whether the input is enabled or disabled.

- **user_output1_pulseWidth** – pulse width, when the value OUT_MODE_EXPOSE_START is assigned to the parameter “user_output1_mode”

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_output1_pulseWidth",
        "type": "uint32_t",
        "access": "write",
        "index": 122,
        "offset": 1308,
        "size": 4,
        "value": 1000,
        "min": 10,
        "max": 1000000,
        "step": 10,
        "defaultValue": 1000,
        "units": "ns"
}
```

- **user_output2_enabled** - enabling/disabling output №2. When disabled, the output has low level. When enabled, the signal is set by the parameters “user_output2_mode” and “user_output2_pulseWidth”.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_output2_enabled",
        "type": "uint32_t",
        "access": "write",
        "index": 123,
        "offset": 1312,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_output2_mode** - operating mode of output №2. Sets the type of output signal.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "user_output2_mode",
        "type": "uint32_t",
        "access": "write",
        "index": 124,
        "offset": 1316,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 5,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "OUT_MODE_EXPOSE_START",
                        "label": "Exposure start"
                },
                {
                        "value": 1,
                        "key": "OUT_MODE_EXPOSE_TIME",
                        "label": "Exposure time"
                },
                {
                        "value": 2,
                        "key": "OUT_MODE_IN1_REPEATER",
                        "label": "In1 repeater"
                },
                {
                        "value": 3,
                        "key": "OUT_MODE_IN2_REPEATER",
                        "label": "In2 repeater"
                },
                {
                        "value": 4,
                        "key": "OUT_MODE_IN3_REPEATER",
                        "label": "In3 repeater"
                },
                {
                        "value": 5,
                        "key": "OUT_MODE_IN3_REPEATER",
```

```
                        "label": "Smart"
                }
        ]
}
```

**NOTE:**

- OUT_MODE_EXPOSE_START - formation of the output pulse upon the start of frame exposure that triggers the measurement cycle;
- OUT_MODE_EXPOSE_TIME - formation of high level output signal during frame exposure for every measurement cycle;
- OUT_MODE_IN1_REPEATER - duplication of the signal from input №1 to the output whether the input is enabled or disabled;
- OUT_MODE_IN2_REPEATER - duplication of the signal from input №2 to the output whether the input is enabled or disabled;
- OUT_MODE_IN3_REPEATER - duplication of the signal from input №3 to the output whether the input is enabled or disabled.

- **user_output2_pulseWidth** - pulse width, when the value OUT_MODE_EXPOSE_START is assigned to the parameter “user_output2_mode”.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_output2_pulseWidth",
        "type": "uint32_t",
        "access": "write",
        "index": 125,
        "offset": 1320,
        "size": 4,
        "value": 1000,
        "min": 10,
        "max": 1000000,
        "step": 10,
        "defaultValue": 1000,
        "units": "ns"
}
```

## Section “Profile Dump”

- **user_dump_enabled** - enabling profile recording to the internal memory of the device - generating a dump. Recording will stop when the maximum dump capacity is reached, or when “user_dump_capacity” is reached, or when FALSE is written to this parameter. Before starting the dump recording, the “user_trigger_sync_value” and “user_trigger_counter_value” counters will be reset to the value “0”.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_dump_enabled",
        "type": "uint32_t",
        "access": "write",
        "index": 126,
        "offset": 1324,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_dump_capacity** - number of profiles for dump recording defined by the user. When this value is reached, recording will automatically stop and the value of the parameter “user_dump_enabled” will become FALSE.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_dump_capacity",
        "type": "uint32_t",
        "access": "write",
        "index": 127,
        "offset": 1328,
        "size": 4,
        "value": 80000,
        "min": 1,
        "max": 80000,
        "step": 0,
        "defaultValue": 80000,
        "units": "profiles"
}
```

- **user_dump_size** - current number of profiles in the dump. This value is reset to “0” before the start of dump recording. While the dump is recorded, this value is incremented.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_dump_size",
        "type": "uint32_t",
        "access": "read_only",
        "index": 128,
        "offset": 1332,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 80000,
        "step": 0,
        "defaultValue": 0,
        "units": "profiles"
}
```


- **user_dump_timeStamp** - timestamp of the dump (may be concidered as a unique identifier). It is set by the device when starting the dump recording.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_dump_timeStamp",
        "type": "uint64_t",
        "access": "read_only",
        "index": 129,
        "offset": 1336,
        "size": 8,
        "value": 0,
        "min": 0,
        "max": 18446744073709551615,
        "step": 0,
        "defaultValue": 0,
        "units": "ticks"
}
```


- **user_dump_view3d_motionType** - the type of motion system on which the device is installed. The parameter value is used to correctly display the dump as a 3D model.

**Object description**

```
{
        name: String,
```

```
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_dump_view3d_motionType",
        "type": "uint32_t",
        "access": "write",
        "index": 130,
        "offset": 1344,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "MOTION_TYPE_LINEAR",
                        "label": "Linear"
                },
                {
                        "value": 1,
                        "key": "MOTION_TYPE_RADIAL",
                        "label": "Radial"
                }
        ]
}
```

**NOTE:**

- MOTION_TYPE_LINEAR - linear motion system;
- MOTION_TYPE_RADIAL - radial motion system.

- **user_dump_view3d_ySource** - source of Y axis coordinates. The parameter value is used to correctly display the dump as a 3D model.

**Object description**

```
{
        name: String,
        type: String,
```

```
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_dump_view3d_ySource",
        "type": "uint32_t",
        "access": "write",
        "index": 131,
        "offset": 1348,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 2,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "Y_AXIS_SYSTEM_TIME",
                        "label": "System time"
                },
                {
                        "value": 1,
                        "key": "Y_AXIS_STEP_COUNTER",
                        "label": "Step counter"
                },
                {
                        "value": 2,
                        "key": "Y_AXIS_MEASURES_COUNTER",
                        "label": "Measures counter"
                }
        ]
}
```

**NOTE:**

- Y_AXIS_SYSTEM_TIME - internal timer of the device;
- Y_AXIS_STEP_COUNTER - parameter “user_trigger_counter_value”;
- Y_AXIS_MEASURES_COUNTER - measurement counter.

- **user_dump_view3d_yStep** - step multiplier along the Y axis.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_dump_view3d_yStep",
        "type": "double_t",
        "access": "write",
        "index": 132,
        "offset": 1352,
        "size": 8,
        "value": 0.0005,
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 0.0005,
        "units": "mm"
}
```

- **user_dump_view3d_paintMode** - mode of 3D model coloring. This parameter is used for displaying a 3D model in the WEB interface.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
```

```
        "name": "user_dump_view3d_paintMode",
        "type": "uint32_t",
        "access": "write",
        "index": 133,
        "offset": 1360,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "PAINT_MODE_HEIGHTMAP",
                        "label": "Heightmap"
                },
                {
                        "value": 1,
                        "key": "PAINT_MODE_INTENSITY",
                        "label": "Intensity"
                }
        ]
}
```


**NOTE:**

- PAINT_MODE_HEIGHTMAP - heightmap coloring;
- PAINT_MODE_INTENSITY - intensity mapping, the value of the parameter “user_streams_includeIntensity” must be TRUE.

- **user_dump_view3d_decimation** - decimation of profiles when displaying a 3D model. This parameter is used for displaying a 3D model in the WEB interface.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "user_dump_view3d_decimation",
        "type": "uint32_t",
        "access": "write",
        "index": 134,
        "offset": 1364,
        "size": 4,
        "value": 1,
        "min": 1,
        "max": 200,
        "step": 0,
        "defaultValue": 1,
        "valuesEnum": [
                {
                        "value": 1,
                        "key": "DUMP_VIEW3D_DECIM_1",
                        "label": "No"
                },
                {
                        "value": 2,
                        "key": "DUMP_VIEW3D_DECIM_2",
                        "label": "2"
                },
                {
                        "value": 5,
                        "key": "DUMP_VIEW3D_DECIM_5",
                        "label": "5"
                },
                {
                        "value": 10,
                        "key": "DUMP_VIEW3D_DECIM_10",
                        "label": "10"
                },
                {
                        "value": 20,
                        "key": "DUMP_VIEW3D_DECIM_20",
                        "label": "20"
                },
                {
                        "value": 50,
                        "key": "DUMP_VIEW3D_DECIM_50",
                        "label": "50"
                },
                {
                        "value": 100,
                        "key": "DUMP_VIEW3D_DECIM_100",
                        "label": "100"
                },
                {
                        "value": 200,
                        "key": "DUMP_VIEW3D_DECIM_200",
                        "label": "200"
                }
        ]
}
```


**NOTE:**

- DUMP_VIEW3D_DECIM_1 - all profile dumps are displayed;
- DUMP_VIEW3D_DECIM_2 - step for displaying profile dumps on the screen: every 2nd profile is displayed;

- DUMP_VIEW3D_DECIM_5 - step for displaying profile dumps on the screen: every 5th profile is displayed;
- DUMP_VIEW3D_DECIM_10 - step for displaying profile dumps on the screen: every 10th profile is displayed;
- DUMP_VIEW3D_DECIM_20 - step for displaying profile dumps on the screen: every 20th profile is displayed;
- DUMP_VIEW3D_DECIM_50 - step for displaying profile dumps on the screen: every 50th profile is displayed;
- DUMP_VIEW3D_DECIM_100 - step for displaying profile dumps on the screen: every 100th profile is displayed;
- DUMP_VIEW3D_DECIM_200 - step for displaying profile dumps on the screen: every 200th profile is displayed.

## Section “Protocol Ethernet/IP”

- **user_eip_tcpPort** - port number on which the device is waiting for incoming TCP connections via protocol Ethernet/IP.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_eip_tcpPort",
        "type": "uint32_t",
        "access": "write",
        "index": 135,
        "offset": 1368,
        "size": 4,
        "value": 44818,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 44818
}
```

- **user_eip_udpPort** - port number on which the device expects to receive UDP data packets via protocol Ethernet/IP.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "user_eip_udpPort",
        "type": "uint32_t",
        "access": "write",
        "index": 136,
        "offset": 1372,
        "size": 4,
        "value": 2222,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 2222
}
```

## Section “Smart”

- **user_smart_bindDistance** - distance of contours (profile fragments) “binding” for splitting into lines.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (float_t),
        defaultValue: Number (float_t),
        min: Number (float_t),
        max: Number (float_t),
        step: Number (float_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_smart_bindDistance",
        "type": "float_t",
        "access": "write",
        "index": 137,
        "offset": 1376,
        "size": 4,
```

```
        "value": 10,
        "min": 0.0010000000474974513,
        "max": 100,
        "step": 0,
        "defaultValue": 10,
        "units": "mm"
}
```


- **user_smart_divideThreshold** - threshold of line detection in profile – sets the minimum required deviation of points from the line for dividing into two new lines (“divide threshold”).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (float_t),
        defaultValue: Number (float_t),
        min: Number (float_t),
        max: Number (float_t),
        step: Number (float_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_smart_divideThreshold",
        "type": "float_t",
        "access": "write",
        "index": 138,
        "offset": 1380,
        "size": 4,
        "value": 1,
        "min": 0.0010000000474974513,
        "max": 100,
        "step": 0,
        "defaultValue": 1,
        "units": "mm"
}
```


- **user_smart_minContourSize** – the minimum required number of points in the contour (profile fragment) for its participation in the process of dividing into lines.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
```

```
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_smart_minContourSize",
        "type": "uint32_t",
        "access": "write",
        "index": 139,
        "offset": 1384,
        "size": 4,
        "value": 5,
        "min": 2,
        "max": 1296,
        "step": 0,
        "defaultValue": 5,
        "units": "points"
}
```


- **user_smart_maxLinesCount** - the maximum number of lines that the profile can be split into.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_smart_maxLinesCount",
        "type": "uint32_t",
        "access": "write",
        "index": 140,
        "offset": 1388,
        "size": 4,
        "value": 20,
        "min": 1,
        "max": 100,
        "step": 1,
        "defaultValue": 20,
        "units": "lines"
}
```

- **user_smart_sectorsEnabled** - enables the use of arcs for profile approximation.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_smart_sectorsEnabled",
        "type": "uint32_t",
        "access": "write",
        "index": 146,
        "offset": 1324,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

- **user_smart_sectorsMinPoints** - the minimum required number of points that must form an arc.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
```

```
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_smart_sectorsMinPoints",
        "type": "uint32_t",
        "access": "write",
        "index": 140,
        "offset": 1388,
        "size": 4,
        "value": 20,
        "min": 5,
        "max": 1296,
        "step": 1,
        "defaultValue": 20,
        "units": "points"
}
```

- **user_smart_sectorsMinR** - the minimum radius of the arc circle required for detection.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (float_t),
        defaultValue: Number (float_t),
        min: Number (float_t),
        max: Number (float_t),
        step: Number (float_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_smart_sectorsMinR",
        "type": "float_t",
        "access": "write",
        "index": 138,
        "offset": 1380,
        "size": 4,
        "value": 1,
        "min": 0.0010000000474974513,
        "max": 100,
        "step": 0,
        "defaultValue": 1,
        "units": "mm"
}
```

- **user_smart_sectorsMaxR** - the maximum radius of the arc circle required for detection.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (float_t),
        defaultValue: Number (float_t),
        min: Number (float_t),
        max: Number (float_t),
        step: Number (float_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_smart_sectorsMaxR",
        "type": "float_t",
        "access": "write",
        "index": 138,
        "offset": 1380,
        "size": 4,
        "value": 1,
        "min": 0.0010000000474974513,
        "max": 100,
        "step": 0,
        "defaultValue": 2,
        "units": "mm"
}
```

- **user_smart_sectorsAccuracy** - the required accuracy of approximating a group of points by arc. If the average deviation parameter is exceeded, the element is approximated by a segment.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (float_t),
        defaultValue: Number (float_t),
        min: Number (float_t),
        max: Number (float_t),
        step: Number (float_t),
        units: String
}
```

**JSON example**

```
{
```

```
        "name": "user_smart_sectorsAccuracy",
        "type": "float_t",
        "access": "write",
        "index": 138,
        "offset": 1380,
        "size": 4,
        "value": 0.5,
        "min": 0.0010000000474974513,
        "max": 100,
        "step": 0,
        "defaultValue": 0.5,
        "units": "mm"
}
```


- **user_smart_sectorsCombineThreshold** - the threshold for combining consecutive (adjacent) arcs into one. If there are two adjacent arcs in the profile, then it is checked whether it is necessary to merge them based on the difference between the coordinates of the centers and radii.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (float_t),
        defaultValue: Number (float_t),
        min: Number (float_t),
        max: Number (float_t),
        step: Number (float_t),
        units: String
}
```

**JSON example**


```
{
        "name": "user_smart_sectorsCombineThreshold",
        "type": "float_t",
        "access": "write",
        "index": 138,
        "offset": 1380,
        "size": 4,
        "value": 10,
        "min": 0.0010000000474974513,
        "max": 100,
        "step": 0,
        "defaultValue": 0.5,
        "units": "mm"
}
```


- **user_smart_elapsedTime** - time spent on mathematical processing of the profile (dividing into lines) and calculating the measurement results.

**Object description**

```
{
```

```
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        defaultValue: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "user_smart_elapsedTime",
        "type": "uint32_t",
        "access": "read_only",
        "index": 141,
        "offset": 1392,
        "size": 4,
        "value": 25,
        "min": 0,
        "max": 10000000,
        "step": 0,
        "defaultValue": 0,
        "units": "us"
}
```

# Parameter Group “Factory”

## Section “General Parameters”

- **fact_general_firmwareVer** - firmware version of the device [Major, Minor, Patch].

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
        ],
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
        ],
        min: Number (uint32_t),
```

```
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "fact_general_firmwareVer",
        "type": "u32_arr_t",
        "access": "read_only",
        "index": 0,
        "offset": 0,
        "size": 12,
        "value": [
                2,
                0,
                2
        ],
        "defaultValue": [
                2,
                0,
                2
        ],
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "maxCount": 3
}
```


- **fact_general_hardwareVer** - hardware version of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "fact_general_hardwareVer",
        "type": "uint32_t",
        "access": "read_only",
        "index": 1,
        "offset": 12,
        "size": 4,
        "value": 302388224,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 302388224
```

```
}
```

- **fact_general_productCode** - identifier of device type.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_general_productCode",
        "type": "uint32_t",
        "access": "locked",
        "index": 2,
        "offset": 16,
        "size": 4,
        "value": 627,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 627
}
```

- **fact_general_serial** - serial number of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_general_serial",
        "type": "uint32_t",
        "access": "locked",
```

```
        "index": 3,
        "offset": 20,
        "size": 4,
        "value": 7057566,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0
}
```


- **fact_general_pcbSerial** - serial number of printed curcuit board of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "fact_general_pcbSerial",
        "type": "uint32_t",
        "access": "locked",
        "index": 4,
        "offset": 24,
        "size": 4,
        "value": 7057566,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0
}
```


- **fact_general_lifeTime** - total operating time of the device in UNIX format.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
```

```
}
```

**JSON example**

```
{
        "name": "fact_general_lifeTime",
        "type": "uint32_t",
        "access": "locked",
        "index": 5,
        "offset": 28,
        "size": 4,
        "value": 10524,
        "min": 0,
        "max": 1577846300,
        "step": 0,
        "defaultValue": 0,
        "units": "s"
}
```

- **fact_general_workTime** - operating time of the device from the moment it was switched on in UNIX format.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_workTime",
        "type": "uint32_t",
        "access": "locked",
        "index": 6,
        "offset": 32,
        "size": 4,
        "value": 974,
        "min": 0,
        "max": 1577846300,
        "step": 0,
        "defaultValue": 0,
        "units": "s"
}
```

- **fact_general_startsCount** - total number of device starts.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_startsCount",
        "type": "uint32_t",
        "access": "locked",
        "index": 7,
        "offset": 36,
        "size": 4,
        "value": 2921,
        "min": 0,
        "max": 8760,
        "step": 0,
        "defaultValue": 0,
        "units": "times"
}
```

- **fact_general_customerID** – customer ID. The identifier of the company that bought / ordered the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_customerID",
        "type": "uint32_t",
        "access": "locked",
        "index": 8,
        "offset": 40,
        "size": 4,
```

```
        "value": 0,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 0,
        "units": "id"
}
```

- **fact_general_fpgaFreq** - the clock frequency of FPGA for this device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_fpgaFreq",
        "type": "uint32_t",
        "access": "locked",
        "index": 9,
        "offset": 44,
        "size": 4,
        "value": 100000000,
        "min": 10000000,
        "max": 500000000,
        "step": 0,
        "defaultValue": 100000000,
        "units": "Hz"
}
```

- **fact_general_smr** - the beginning of measuring range for Z-coordinate (in mm).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
```

```
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_smr",
        "type": "uint32_t",
        "access": "locked",
        "index": 10,
        "offset": 48,
        "size": 4,
        "value": 80,
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 80,
        "units": "mm"
}
```

- **fact_general_mr** - the size of measuring range for Z-coordinate (in mm).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_mr",
        "type": "uint32_t",
        "access": "locked",
        "index": 11,
        "offset": 52,
        "size": 4,
        "value": 130,
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 130,
        "units": "mm"
}
```

- **fact_general_xsmr** - the size of measuring range for X-coordinate at the beginning of Z.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_xsmr",
        "type": "uint32_t",
        "access": "locked",
        "index": 12,
        "offset": 56,
        "size": 4,
        "value": 40,
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 40,
        "units": "mm"
}
```

- **fact_general_xemr** - the size of measuring range for X-coordinate at the end of Z.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_xemr",
        "type": "uint32_t",
        "access": "locked",
        "index": 13,
```

```
        "offset": 60,
        "size": 4,
        "value": 86,
        "min": 0,
        "max": 20000,
        "step": 0,
        "defaultValue": 86,
        "units": "mm"
}
```


- **fact_general_pixDivider** - divider for obtaining the subpixel position of the profile points in the uncalibrated data transfer mode (the value DATA_FORMAT_RAW_PROFILE is assigned to the parameter “user_streams_format”).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "fact_general_pixDivider",
        "type": "uint32_t",
        "access": "read_only",
        "index": 14,
        "offset": 64,
        "size": 4,
        "value": 32,
        "min": 0,
        "max": 65535,
        "step": 8,
        "defaultValue": 32
}
```


- **fact_general_profDivider** - divider for obtaining the subpixel position of the profile points in the calibrated data transfer mode (the value DATA_FORMAT_PROFILE is assigned to the parameter “user_streams_format”).

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
```

```
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_general_profDivider",
        "type": "uint32_t",
        "access": "read_only",
        "index": 15,
        "offset": 68,
        "size": 4,
        "value": 16384,
        "min": 0,
        "max": 65535,
        "step": 8,
        "defaultValue": 16384
}
```

- **fact_general_oemDevName** - the device name assigned by the OEM customer.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: String,
        maxLen: Number (uint32_t),
        defaultValue: String
}
```

**JSON example**

```
{
        "name": "fact_general_oemDevName",
        "type": "string_t",
        "access": "locked",
        "index": 16,
        "offset": 72,
        "size": 128,
        "value": "Laser scanner",
        "maxLen": 128,
        "defaultValue": "Laser scanner"
}
```

- **fact_general_authStatus** - authorization status for changing the factory settings of the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "fact_general_authStatus",
        "type": "uint32_t",
        "access": "read_only",
        "index": 17,
        "offset": 200,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "AUTH_STATUS_USER",
                        "label": "User"
                },
                {
                        "value": 65535,
                        "key": "AUTH_STATUS_FACTORY",
                        "label": "Factory"
                }
        ]
}
```

**NOTE:**

- AUTH_STATUS_USER - authorized as user, factory settings cannot be changed;
- AUTH_STATUS_FACTORY - authorized as manufacturer, factory settings can be changed.

- **fact_general_calibTimestamp** - date and time of the scanner calibration. It is taken from the calibration file, stored in UNIX format.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint64_t),
        min: Number (uint64_t),
        max: Number (uint64_t),
        step: Number (uint64_t),
        defaultValue: Number (uint64_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_general_calibTimestamp",
        "type": "uint64_t",
        "access": "read_only",
        "index": 15,
        "offset": 68,
        "size": 8,
        "value": 1622796914,
        "min": 0,
        "max": 0xFFFFFFFFFFFFFFFF,
        "step": 0,
        "defaultValue": 1622796914,
        "units": "s"
}
```

## Section “Sensor”

- **fact_sensor_name** - name of the sensor used in the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: String,
        maxLen: Number (uint32_t),
        defaultValue: String
}
```

**JSON example**

```
{
        "name": "fact_sensor_name",
        "type": "string_t",
        "access": "locked",
        "index": 18,
        "offset": 204,
        "size": 64,
        "value": "TYPE1",
```

```
        "maxLen": 64,
        "defaultValue": "TYPE1"
}
```

- **fact_sensor_width** - the number of pixels in the sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_sensor_width",
        "type": "uint32_t",
        "access": "locked",
        "index": 19,
        "offset": 268,
        "size": 4,
        "value": 648,
        "min": 648,
        "max": 648,
        "step": 0,
        "defaultValue": 648,
        "units": "pixels"
}
```

- **fact_sensor_height** - the number of lines in the sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_sensor_height",
        "type": "uint32_t",
        "access": "locked",
        "index": 20,
        "offset": 272,
        "size": 4,
        "value": 488,
        "min": 488,
        "max": 488,
        "step": 0,
        "defaultValue": 488,
        "units": "lines"
}
```


- **fact_sensor_pixFreq** - pixel frequency for the installed sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "fact_sensor_pixFreq",
        "type": "uint32_t",
        "access": "locked",
        "index": 21,
        "offset": 276,
        "size": 4,
        "value": 40000000,
        "min": 1000000,
        "max": 500000000,
        "step": 0,
        "defaultValue": 40000000,
        "units": "Hz"
}
```


- **fact_sensor_frmConstPart** - constant part of the frame cycle.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
```

```
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_sensor_frmConstPart",
        "type": "uint32_t",
        "access": "locked",
        "index": 22,
        "offset": 280,
        "size": 4,
        "value": 3500,
        "min": 200,
        "max": 200000,
        "step": 0,
        "defaultValue": 3500,
        "units": "ticks"
}
```

- **fact_sensor_frmPerLinePart** - part of the frame cycle per every line.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_sensor_frmPerLinePart",
        "type": "uint32_t",
        "access": "locked",
        "index": 23,
        "offset": 284,
        "size": 4,
        "value": 160,
        "min": 10,
        "max": 100000,
        "step": 0,
        "defaultValue": 160,
        "units": "ticks"
}
```

- **fact_sensor_minExposure** - the minimum possible value of exposure time.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_sensor_minExposure",
        "type": "uint32_t",
        "access": "locked",
        "index": 24,
        "offset": 288,
        "size": 4,
        "value": 3000,
        "min": 0,
        "max": 100000000,
        "step": 10,
        "defaultValue": 3000,
        "units": "ns"
}
```

- **fact_sensor_maxExposure** - the maximum possible value of exposure time.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_sensor_maxExposure",
        "type": "uint32_t",
        "access": "locked",
```

```
        "index": 25,
        "offset": 292,
        "size": 4,
        "value": 300000000,
        "min": 0,
        "max": 300000000,
        "step": 10,
        "defaultValue": 300000000,
        "units": "ns"
}
```


- **fact_sensor_imgFlip** - image flip mode (allows to flip a profile in direction of selected axes). It is applied directly to the image transmitted by the sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**


```
{
        "name": "fact_sensor_imgFlip",
        "type": "uint32_t",
        "access": "locked",
        "index": 26,
        "offset": 296,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 3,
        "step": 0,
        "defaultValue": 0,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FLIP_MODE_OFF",
                        "label": "No"
                },
                {
                        "value": 1,
                        "key": "FLIP_MODE_X",
                        "label": "X"
```

```
                },
                {
                        "value": 2,
                        "key": "FLIP_MODE_Z",
                        "label": "Z"
                },
                {
                        "value": 3,
                        "key": "FLIP_MODE_XZ",
                        "label": "XZ"
                }
        ]
}
```

**NOTE:**

- FLIP_MODE_OFF – flip is disabled;
- FLIP_MODE_X - flip along the X axis of the scanner;
- FLIP_MODE_Z - flip along the Z axis of the scanner;
- FLIP_MODE_XZ - flip along both axes (X and Z).

- **fact_sensor_analogGain** – value of analog gain of the image by the sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_sensor_analogGain",
        "type": "uint32_t",
        "access": "locked",
        "index": 36,
        "offset": 356,
        "size": 4,
        "value": 5,
        "min": 0,
        "max": 7,
        "step": 0,
        "defaultValue": 5
}
```

- **fact_sensor_digitalGain** - value of digital gain of the image by the sensor.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_sensor_digitalGain",
        "type": "uint32_t",
        "access": "locked",
        "index": 37,
        "offset": 360,
        "size": 4,
        "value": 36,
        "min": 0,
        "max": 55,
        "step": 0,
        "defaultValue": 36
}
```

- **fact_sensor_blackOdd** - the level of black for odd lines.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_sensor_blackOdd",
        "type": "uint32_t",
        "access": "locked",
        "index": 38,
        "offset": 364,
        "size": 4,
        "value": 2600,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 2600
```

```
}
```

- **fact_sensor_blackEven** - the level of black for even lines.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_sensor_blackEven",
        "type": "uint32_t",
        "access": "locked",
        "index": 39,
        "offset": 368,
        "size": 4,
        "value": 2650,
        "min": 0,
        "max": 65535,
        "step": 0,
        "defaultValue": 2650
}
```

- **fact_sensor_edrPiecewiseDiv1** - the first inflection point of the sensor response in the mode “user_sensor_edrType” EDR_PIECEWISE.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_sensor_edrPiecewiseDiv1",
        "type": "uint32_t",
```

```
        "access": "locked",
        "index": 41,
        "offset": 628,
        "size": 4,
        "value": 32,
        "min": 1,
        "max": 255,
        "step": 0,
        "defaultValue": 32
}
```

- **fact_sensor_edrPiecewiseDiv2** - the second inflection point of the sensor response in the mode “user_sensor_edrType” EDR_PIECEWISE.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_sensor_edrPiecewiseDiv2",
        "type": "uint32_t",
        "access": "locked",
        "index": 42,
        "offset": 632,
        "size": 4,
        "value": 64,
        "min": 1,
        "max": 255,
        "step": 0,
        "defaultValue": 64
}
```

- **fact_sensor_initRegs** - register values of sensor for initialization [regAddr, regValue ...].

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
```

```
                ...
        ],
        defaultValue: [
                Number (uint32_t),
                ...
        ],
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**


```
{
        "name": "fact_sensor_initRegs",
        "type": "u32_arr_t",
        "access": "locked",
        "index": 40,
        "offset": 372,
        "size": 144,
        "value": [
                41,
                1,
                83,
                155,
                58,
                20,
                59,
                0,
                60,
                11,
                69,
                9,
                80,
                4,
                97,
                0,
                98,
                12,
                101,
                98,
                102,
                34,
                103,
                64,
                106,
                90,
                107,
                110,
                108,
                91,
                109,
                82,
                110,
                80,
                117,
                91
        ],
        "defaultValue": [
                41,
                1,
                83,
```

```
                155,
                58,
                20,
                59,
                0,
                60,
                11,
                69,
                9,
                80,
                4,
                97,
                0,
                98,
                12,
                101,
                98,
                102,
                34,
                103,
                64,
                106,
                90,
                107,
                110,
                108,
                91,
                109,
                82,
                110,
                80,
                117,
                91
        ],
        "min": 0,
        "max": 65535,
        "step": 0,
        "maxCount": 64
}
```

## Section “Ethernet network”

- **fact_network_macAddr** - physical address of the device on the network.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
        ]
```

```
        defaultValue: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
        ]
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        maxCount: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_network_macAddr",
        "type": "u32_arr_t",
        "access": "locked",
        "index": 27,
        "offset": 300,
        "size": 24,
        "value": [
                0,
                10,
                53,
                107,
                176,
                158
        ],
        "defaultValue": [
                0,
                10,
                53,
                1,
                2,
                3
        ],
        "min": 0,
        "max": 255,
        "step": 0,
        "maxCount": 6
}
```

- **fact_network_forceAutoNegTime** - the time after which autonegotiation of the Ethernet connection will be forcibly enabled if the connection is not established.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
```

```
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_network_forceAutoNegTime",
        "type": "uint32_t",
        "access": "locked",
        "index": 28,
        "offset": 324,
        "size": 4,
        "value": 5,
        "min": 0,
        "max": 255,
        "step": 0,
        "defaultValue": 5,
        "units": "s"
}
```

- **fact_network_webSockServicePort** - port number for WEB socket for transferring service data. Used by the WEB page.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_network_webSockServicePort",
        "type": "uint32_t",
        "access": "locked",
        "index": 29,
        "offset": 328,
        "size": 4,
        "value": 50002,
        "min": 16384,
        "max": 65535,
        "step": 0,
        "defaultValue": 50002
}
```

- **fact_network_webSockDataPort** - port number for WEB socket for transferring big data. Used by the WEB page.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_network_webSockDataPort",
        "type": "uint32_t",
        "access": "locked",
        "index": 30,
        "offset": 332,
        "size": 4,
        "value": 50003,
        "min": 16384,
        "max": 65535,
        "step": 0,
        "defaultValue": 50003
}
```

- **fact_network_webSockMathPort** - port number for WEB socket for transferring data of the “smart” module. Used by the WEB page.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_network_webSockMathPort",
        "type": "uint32_t",
        "access": "locked",
        "index": 31,
        "offset": 336,
        "size": 4,
        "value": 50004,
        "min": 16384,
        "max": 65535,
```

```
        "step": 0,
        "defaultValue": 50004
}
```

## Section “Laser”

- **fact_laser_waveLength** - the wavelength of the laser installed in the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**

```
{
        "name": "fact_laser_waveLength",
        "type": "uint32_t",
        "access": "locked",
        "index": 32,
        "offset": 340,
        "size": 4,
        "value": 650,
        "min": 0,
        "max": 10000,
        "step": 0,
        "defaultValue": 650,
        "units": "nm"
}
```

- **fact_laser_minValue** - the minimum value of Digital-to-Analog converter (DAC). At this value, the laser stops emitting light.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
```

```
}
```

**JSON example**

```
{
        "name": "fact_laser_minValue",
        "type": "uint32_t",
        "access": "locked",
        "index": 33,
        "offset": 344,
        "size": 4,
        "value": 0,
        "min": 0,
        "max": 4095,
        "step": 0,
        "defaultValue": 0
}
```

- **fact_laser_maxValue** - the maximum value of Digital-to-Analog converter (DAC). At this value, the laser begins to emit light at full power, further increase in the DAC value does not lead to an increase in brightness.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t)
}
```

**JSON example**

```
{
        "name": "fact_laser_maxValue",
        "type": "uint32_t",
        "access": "locked",
        "index": 34,
        "offset": 348,
        "size": 4,
        "value": 4095,
        "min": 0,
        "max": 4095,
        "step": 0,
        "defaultValue": 4095
}
```

## Section “Smart”

- **fact_smart_enabled** - enabling/disabling the capabilities of the smart device. If the parameter is not found, then the “Smart” module is not implemented in the device.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        valuesEnum: [
                {       value: Number (uint32_t),
                        key: String,
                        label: String
                },
                …
                ]
}
```

**JSON example**

```
{
        "name": "fact_smart_enabled",
        "type": "uint32_t",
        "access": "locked",
        "index": 35,
        "offset": 352,
        "size": 4,
        "value": 1,
        "min": 0,
        "max": 1,
        "step": 0,
        "defaultValue": 1,
        "valuesEnum": [
                {
                        "value": 0,
                        "key": "FALSE",
                        "label": "false"
                },
                {
                        "value": 1,
                        "key": "TRUE",
                        "label": "true"
                }
        ]
}
```

## Section “Profile Dump”

- **fact_dump_unitSize** – the size (in bytes) of unit for storing the profile in the dump. A dump is a set of units that are sequentially filled with profiles when the dump is recorded. The profile in each unit is stored in exactly the same format as transmitted via UDP. The parameter value is set during the development of scanner firmware and corresponds to the maximum possible profile size for a particular scanner version and firmware version, plus (possibly) several bytes (tens/hundreds of bytes) for alignment.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String,
}
```

**JSON example**

```
{
        "name": "fact_dump_unitSize",
        "type": "uint32_t",
        "access": "read_only",
        "index": 36,
        "offset": 356,
        "size": 4,
        "value": 8256,
        "min": 0,
        "max": 4294967295,
        "step": 0,
        "defaultValue": 8256,
        "units": "bytes"
}
```

## Section “Service Protocol”

- **fact_serviceProtocol_maxPacketSize** – the maximum size (in bytes) of a packet transmitted by service protocol as UDP packet over Ethernet interface.

**Object description**

```
{
        name: String,
        type: String,
        access: String,
        index: Number (uint32_t),
        offset: Number (uint32_t),
        size: Number (uint32_t),
```

```
        value: Number (uint32_t),
        min: Number (uint32_t),
        max: Number (uint32_t),
        step: Number (uint32_t),
        defaultValue: Number (uint32_t),
        units: String
}
```

**JSON example**


```
{
        "name": "fact_serviceProtocol_maxPacketSize",
        "type": "uint32_t",
        "access": "locked",
        "index": 28,
        "offset": 324,
        "size": 4,
        "value": 1200,
        "min": 0,
        "max": 65000,
        "step": 0,
        "defaultValue": 1200,
        "units": "bytes"
}
```

# The list of commands

## Description of device commands

The commands sent to the device are intended for searching the devices on the network, reading and setting parameters, loading service data, updating the firmware, receiving frames generated by the CMOS sensor, and other functions. Commands can be sent both via the service protocol and using WebAPI.

The list of commands supported by the device is available via WebAPI with the URI “XXX.XXX.XXX.XXX/api/v1/config/commands”. If no URI is specified for the command, it is not available via WebAPI.

The command can be unlocked for access - the status of the “access” field is “unlocked” or blocked - the status is “locked”. Blocked commands become available after authorization as a “manufacturer”.

The result of the command completion is transmitted either to the MessagePack - when the command is sent via the service protocol, or as JSON text - when the command is requested via WebAPI using the URI. In order to obtain examples of query results, the queries presented in the "WebAPI" section are used.

### General commands

#### GET_HELLO

- Function: search for the devices on the network, getting their main parameters in the form of a formalized response;
- Access: “unlocked”;
- URI “GET”: "/hello";
- Payload: no;
- Answer:

```
{
        user_general_deviceName: String,
        fact_general_productCode: Number (uint32_t),
        fact_general_serial: Number (uint32_t),
        fact_general_firmwareVer: [
                Number (uint32_t),
                Number (uint32_t),
                Number (uint32_t)
        ],
        fact_general_hardwareVer: Number (uint32_t),
        fact_general_smr: Number (uint32_t),
        fact_general_mr: Number (uint32_t),
        fact_general_xsmr: Number (uint32_t),
        fact_network_macAddr: String (*),
        fact_laser_waveLength: Number (uint32_t),
        user_network_speed: Number (uint32_t),
        user_network_autoNeg: Bool,
        user_network_ip: String (*),
        user_network_mask: String (*),
        user_network_gateway: String (*),
        user_network_hostIP: String (*),
        user_network_hostPort: Number (uint32_t),
        user_network_webPort: Number (uint32_t),
        user_network_servicePort: Number (uint32_t),
        user_streams_udpEnabled: Bool,
```

```
        user_streams_format: Number (uint32_t),
        commands: [
                {
                        name: String,
                        uri: String (*),
                        access: String (*)
                },
                ...
                ]
}
```

```
*
fact_network_macAddr: string format "XX:XX:XX:XX:XX:XX", XX - hexadecimal values from 00 to
FF;
user_network_ip: string format "XX.XX.XX.XX", XX - decimal values from 0 to 255;
user_network_mask: string format "XX.XX.XX.XX", XX - decimal values from 0 to 255;
user_network_gateway: string format "XX.XX.XX.XX", XX - decimal values from 0 to 255;
user_network_hostIP: string format "XX.XX.XX.XX", XX - decimal values from 0 to 255;
commands[i].uri: available only for commands that are accessible via webAPI.
commands[i].access: possible options "locked", "read_only" и "write".
```

- Example:

JSON


```
{
        "user_general_deviceName": "2D laser scanner",
        "fact_general_productCode": 627,
        "fact_general_serial": 6604512,
        "fact_general_firmwareVer": [
                2,
                0,
                0
        ],
        "fact_general_hardwareVer": 302388224,
        "fact_general_smr": 80,
        "fact_general_mr": 130,
        "fact_general_xsmr": 40,
        "fact_network_macAddr": "00:0A:35:64:C6:E0",
        "fact_laser_waveLength": 650,
        "user_network_speed": 1000,
        "user_network_autoNeg": true,
        "user_network_ip": "192.168.1.30",
        "user_network_mask": "255.255.255.0",
        "user_network_gateway": "192.168.1.1",
        "user_network_hostIP": "192.168.1.2",
        "user_network_hostPort": 50001,
        "user_network_webPort": 80,
        "user_network_servicePort": 50011,
        "user_streams_udpEnabled": false,
        "user_streams_format": 17,
        "commands": [
                {
                        "name": "GET_PARAMS_DESCRIPTION",
                        "uri": "/api/v1/config/params",
                        "access": "unlocked"
                },
                {
                        "name": "GET_COMMANDS_DESCRIPTION",
                        "uri": "/api/v1/config/commands",
                        "access": "unlocked"
                },
```

```
                {
                        "name": "GET_PARAMETERS",
                        "uri": "/api/v1/config/params/values",
                        "access": "unlocked"
                },
                {
                        "name": "SET_PARAMETERS",
                        "uri": "/api/v1/config/params/values",
                        "access": "unlocked"
                },
                {
                        "name": "GET_RETURN_CODES_DESCRIPTION",
                        "uri": "/api/v1/config/returnCodes",
                        "access": "unlocked"
                },
                {
                        "name": "SAVE_PARAMETERS",
                        "uri": "/api/v1/config/params/save",
                        "access": "unlocked"
                },
                {
                        "name": "GET_LOG_DATA",
                        "access": "unlocked"
                }
        ]
}
```

### GET_PARAMS_DESCRIPTION

- Function: getting general information about all device parameters. The formalized description of the parameter will contain its name, type, access mode, index in the parameter array, offset for binary data, parameter data size, current value, default value, minimum and maximum values, parameter value step, maximum number of elements - for arrays;
- Access: “unlocked”;
- URI “GET”: "/api/v1/config/params";
- Payload: no;
- Answer:

```
{
        byte_order: String (*),
        factory: [
                {} (*),
                ...
        ],
        user: [
                {} (*),
                ...
        ]
}
```

```
*
byte_order: has the value "little_endian";

{} : parameter object, description of parameter structures is given in the section
Description of device parameters
```

- Example:

JSON


```
{
        "byte_order": "little_endian",
        "factory": [
                {
                        "name": "fact_general_firmwareVer",
                        "type": "u32_arr_t",
                        "access": "read_only",
                        "index": 0,
                        "offset": 0,
                        "size": 12,
                        "value": [
                                2,
                                0,
                                0
                        ],
                        "defaultValue": [
                                2,
                                0,
                                0
                        ],
                        "min": 0,
                        "max": 4294967295,
                        "step": 0,
                        "maxCount": 3
                },
                {
                        "name": "fact_general_hardwareVer",
                        "type": "uint32_t",
                        "access": "read_only",
                        "index": 1,
                        "offset": 12,
                        "size": 4,
                        "value": 302388224,
                        "min": 0,
                        "max": 4294967295,
                        "step": 0,
                        "defaultValue": 302388224
                },
                {
                        "name": "fact_general_productCode",
                        "type": "uint32_t",
                        "access": "locked",
                        "index": 2,
                        "offset": 16,
                        "size": 4,
                        "value": 627,
                        "min": 0,
                        "max": 65535,
                        "step": 0,
                        "defaultValue": 627
                },
...
```


### GET_COMMANDS_DESCRIPTION

- Function: getting the list of commands supported by the device. The formalized description will contain the command name, Web API accessibility, the command ID, and access mode;

- Access: “unlocked”;
- URI “GET”: "/api/v1/config/commands";
- Payload: no;
- Answer:

```
{
        commands: [
                {
                        name: String,
                        uri: String (*),
                        access: String
                } ,
                ...
        ]
}
```

```
*
commands[i].uri: available only for commands that are accessible via webAPI.
```

- Example:

JSON

```
{
        "commands": [
                {
                        "name": "GET_HELLO",
                        "uri": "/hello",
                        "access": "unlocked"
                },
                {
                        "name": "GET_PARAMS_DESCRIPTION",
                        "uri": "/api/v1/config/params",
                        "access": "unlocked"
                },
                {
                        "name": "GET_COMMANDS_DESCRIPTION",
                        "uri": "/api/v1/config/commands",
                        "access": "unlocked"
                },
                {
                        "name": "GET_PARAMETERS",
                        "uri": "/api/v1/config/params/values",
                        "access": "unlocked"
                },

...
```

### GET_PARAMETERS

- Function: reading device parameters. For reading, you can request specific parameters by name or index;
- Access: “unlocked”;
- URI “GET”: "/api/v1/config/params/values";

  The WebAPI request format is described in the section “WebAPI”, subsection “Reading and Writing Device parameters”.
- Payload:

Request option 1

```
{
        name: String,
        index: Number (uint32_t)
}
```

*
Combining name and index parameters is supported at the same time. Only 1 parameter by name and/or 1 parameter by index is requested at a time.

Request option 2

```
{
        names:{
                “parameter_1_name”,.
                “parameter_2_name”,
                ...,
        },
        indexes:{
                “parameter_3_index”,.
                “parameter_4_index”,
                ...,
        }
}
```

*
Combining name and index parameters is supported at the same time.

- Answer:

```
{
        “parameter_1_name”: “parameter_1_value”,
        “parameter_2_name”: “parameter_2_value”,
        ...
}
```

*
Regardless of the type of request, the response has similar format.

- Example:

```
{
        "fact_general_hardwareVer": 302388224,
        "user_trigger_counter_dir": 0
}
```

### GET_BIN_PARAMETERS

- Function: reading parameter values in binary format. Each parameter is located according to its index and size (specified in the parameter description);
- Access: “unlocked”;
- Payload: no;
- Answer: array of bytes with sequentially packed parameter values.

### SET_PARAMETERS

- Function: reading parameter values of the device. For setting value, you can request specific parameters by name or index;
- Access: “unlocked”;
- URI “PUT”: "/api/v1/config/params/values";

  The WebAPI request format is described in the section “WebAPI”, subsection “Reading and Writing Device parameters”.
- Payload:

```
{
        “parameter_1_name”: “parameter_1_value”,
        “parameter_2_name”: “parameter_2_value”,
        ...
}
```

- Answer:

```
{
        “parameter_1_name”: String (*),
        “parameter_2_name”: String (*),
        ...
}
```

```
*
String displays the result according to the response codes. On successful completion “RF_OK”.
```

- Example:

JSON

```
{
    "user_sensor_framerate": "RF_OK",
    "user_sensor_exposure1": "RF_OK"
}
```

### GET_RETURN_CODES_DESCRIPTION

- Function: getting text description of the codes of work results and errors returned by the device (response codes);
- Access: “unlocked”;
- URI “GET”: "/api/v1/config/returnCodes";
- Payload: no;
- Answer:

```
{
        “code_1_key”: “code_1_description”,
        “code_2_key”: “code_2_description”,
        ...
}
```

- Example:

JSON


```
{
        "RF_OK": "Success",
        "RF_DISABLED_BY_FACTORY": "The operation/command is not allowed by the manufacturer",
        "RF_BUSY": "Module busy",
        "RF_SUSPENDED": "Module/device is suspended (possibly critical operation is
performed)",
        "RF_NOT_FOUND": "Module/data not found",
        "RF_NOT_ENOUGH_MEMORY": "To perform the operation is not enough memory (reduce the
volume of transmission data packet)",
        "RF_DUPLICATED": "The command or data was repeated.",
        "RF_NOT_VALIDATED": "Before performing this command, the data must be validated",
        "RF_WRITE_IMPOSSIBLE": "No parameter or data can be written",
        "RF_NOT_AUTHORIZED": "To execute an operation/command, authorization is required",
        "RF_PARAM_NOT_FOUND": "This parameter is not found in the device configuration",
        "RF_WRONG_SIZE": "Incorrect size of data or parameters in command",
        "RF_WRONG_DATA_TYPE": "Incorrect type of data or parameters in the command",
        "RF_OUT_OF_BOUNDS": "The value of the parameter or data goes beyond the allowed
limits",
        "RF_NOT_VALID": "Command, data or parameter is not correct",
        "RF_UNKN_TYPE": "Type of parameter, data or attribute not known",
        "RF_NOT_IN_STEP": "The value of the parameter or data does not match to the change
step",
        "RF_COMMAND_HANDLED": "The command with these MID, CID and UID has already been
processed",
        "RF_WRONG_CRC": "Wrong checksum for command or data",
        "RF_WRONG_DEVICE_TYPE": "Command or data does not match the type of this device
(fact_general_deviceType parameter)",
        "RF_WRONG_ARGUMENT": "The command argument(s) do not match the expected",
        "RF_NO_DATA": "No data or they're not ready yet",
        "RF_NOT_SUPPORTED": "The command cannot be handled by the module",
        "RF_INIT_FAULT": "General error during module or hardware initialization process",
        "RF_GENERAL_FAULT": "A general error occurred for an unknown reason"
}
```


## SAVE_PARAMETERS

- Function: saving the current values of the device parameters in the non-volatile memory of the device in the user area. The saved values will be used the next time the device is turned on;
- Access: “unlocked”;
- URI “GET”: "/api/v1/config/params/save";
- Payload: no;
- Answer:

```
{
        result: String (*)
}
```

```
*
result: displays the result according to the response codes. On successful completion
“RF_OK”.
```

- Example:

JSON

```
{
        "result": "RF_OK"
}
```

### SAVE_RECOVERY_PARAMETERS

- Function: saving the current values of the device parameters in the recovery area. These parameters will be applied if the parameters from the user area are damaged;
- Access: “unlocked”;
- Payload: no;
- Answer:

```
{
        result: String (*)
}
```

```
*
result: displays the result according to the response codes. On successful completion
“RF_OK”.
```

### LOAD_RECOVERY_PARAMETERS

- Function: loading device parameters from the recovery area. The loaded values will be written to the user area, the device will automatically reboot;
- Access: “unlocked”;
- URI “GET”: "/api/v1/config/params/recovery/load";
- Payload: no;
- Answer:

```
{
        result: String (*)
}
```

```
result: displays the result according to the response codes. On successful completion
“RF_OK”.
```

- Example:

JSON

```
{
        "result": "RF_OK"
}
```

### REBOOT_DEVICE

- Function: reboot the device;
- Access: “unlocked”;

- URI “GET”: "/api/v1/reboot";
- Payload: no;
- Answer:

```
{
        result: String (*)
}
```

```
result: displays the result according to the response codes. On successful completion
“RF_OK”.
```

- Example:

JSON

```
{
        "result": "RF_OK"
}
```

## Authorization commands

### GET_AUTHORIZATION_TOKEN

- Function: getting a token to generate a key for authorization as “manufacturer”;
- Access: “unlocked”;
- URI “GET”: "/api/v1/authorization";
- Payload: no;
- Answer:

```
{
        status: Number (uint32_t) (*),
        token: String
}
```

```
*
status: displays the result according to the options for the parameter
fact_general_authStatus.
```

- Example:

JSON

```
{
        "status": 0,
        "token":
"28BA6AD17486298A0FBC06C79E2343F05922D16A1BA04A79D0885AE1C91FE4129940B30CC2F34D1F93C741785EC2
9CE7"
}
```

### SET_AUTHORIZATION_KEY

- Function: sending authorization key to the device, if successful, the authorization status on the device will change to “manufacturer”;

- Access: “unlocked”;
- URI “PUT”: "/api/v1/authorization?key=KEY"; More details see in the section “WebAPI”, subsection “Authorization”..
- Payload:

```
{
        key: String
}
```

- Answer:

```
{
        result: String (*),
        status: Number (uint32_t) (*)
}
```

```
*
result: displays the result according to the response codes. On successful completion
“RF_OK”.
status: displays the result according to the options for the parameter
fact_general_authStatus.
```

- Example:

JSON

```
{
    "result": "RF_NOT_VALID",
    "status": 0
}
```

## Commands for sending and saving calibration file

### SET_CALIBRATION_DATA

- Function: writing calibration file to the device;
- Access: “unlocked”;
- Payload: array of bytes with the data of the calibration table file.
- Answer:

```
{
        result: String (*)
}
```

```
result: displays the result according to the response codes. On successful completion
“RF_OK”.
```

### SAVE_CALIBRATION_DATA

- Function: saving calibration file in the non-volatile memory of the device;
- Access: “unlocked”;
- Payload: no;

- Answer:

```
{
        result: String (*)
}
```

result: displays the result according to the response codes. On successful completion “RF_OK”.

### GET_CALIBRATION_INFO

- Function: getting information about calibration file of the device;
- Access: “unlocked”;
- URI “GET”: "/api/v1/calibration/info"; More details see in the section “WebAPI”, subsection “Calibration File”.
- Payload: no;
- Answer:

```
{
        result: String (*),
        type: Number (uint32_t),
        serial: Number (uint32_t),
        data_row_length: Number (uint32_t),
        width: Number (uint32_t),
        height: Number (uint32_t),
        mult_w: Number (uint32_t),
        mult_h: Number (uint32_t),
        time_stamp: Number (uint64_t)
}
```

result: displays the result according to the response codes. On successful completion “RF_OK”,
Other fields are described in the section “Calibration file format”.

## Commands of profile request

### CAPTURE_PROFILE

- Function: command for starting measurements (obtaining a profile) by software request via service protocol. It is available only in “Software, external” and “Software, internal” modes (parameter user_sensor_syncSource = “SYNC_SOFTWARE”). When a command is received, the device starts a measurement cycle, after which a calculation is performed and a standard packet with a profile is sent;
- Access: “unlocked”;
- URI “GET”: "/api/v1/profile/capture"; More details see in the section “WebAPI”, subsection “Profile Request”
- Payload:

```
{
        count: Number (uint32_t)
}
```

- Answer:

```
{
        result: String (*)
}
```

result: displays the result according to the response codes. On successful completion “RF_OK”.

- Example:

JSON

```
{
        "result": "RF_OK"
}
```

### GET_PROFILE

- Function: getting the last registered profile;
- Access: “unlocked”;
- Payload: no;
- Answer: array of bytes with profile data.

## Profile dump commands

### GET_DUMP_DATA

- Function: getting the content of the profile dump;
- Access: “unlocked”;
- Payload:

```
{
        index: Number (uint32_t),
        count: Number (uint32_t),
}
```

```
*
index: the number of requested profile from the memory;
count: total number of requested profiles, starting from the profile specified in index;
```

- Answer: array of bytes with sequentially packed profiles. Profiles are packed in units, the size of which is equal to the value of the parameter “fact_dump_unitSize”. Each profile in the unit has the same structure as profiles transmitted via UDP protocol. An example of the response to the request with parameters “index” = 100, “count” = 3:

| profile №100 | profile №101 | profile №102 |
|---|---|---|
| unit №0 | unit №1 | unit №2 |

| profile (header + data) | alignment | profile (header + data) | alignment | profile (header + data) | alignment |
|---|---|---|---|---|---|

## Commands for image frame acquisition

### GET_FRAME

- Function: getting an image frame from the CMOS sensor;
- Access: “unlocked”;
- Payload: no;
- Answer:

```
If the frame is successfully received:
{
        frame: Blob (*),
        user_roi_active: Bool,
        user_roi_enabled: Bool,
        user_roi_pos: Number (uint32_t),
        user_roi_size: Number (uint32_t),
        frame_width: Number (uint32_t),
        frame_height: Number (uint32_t),
        fact_sensor_width: Number (uint32_t),
        fact_sensor_height: Number (uint32_t)
}
If no frame was received (timeout expired, external synchronization):
{
        "result": "RF_BUSY"
}
```

```
*
frame: an array of bytes with a frame of matrix, each byte is responsible for the brightness
of the pixel starting from the upper left pixel. The image sizes are taken from the
parameters fact_sensor_width and fact_sensor_height.
```

## Commands for getting the log file of the device

### GET_LOG_DATA

- Function: getting a fragment of the device log file with a full description of the records;
- Access: “unlocked”;
- Payload: no;
- Answer:

```
{
        messages: [
                {
                        time: Number (uint32_t),
                        text: String
                },
                ...
        ]
}
```

### GET_LOG_TEXT

- Function: getting a log file of the device operation in a shortened, easy-to-read form.
- Access: “unlocked”;
- URI “GET”: "/api/v1/log";
- Payload: no;
- Answer:

```
{
        “msg_0”: String,
        “msg_1”: String,
        ...
}
```

- Example:

JSON


```
{
"msg_0": "[   INFO   ]  |=====================================================|",
"msg_1": "[   INFO   ]  |================STARTING 2D LASER SCANNER============|",
"msg_2": "[   INFO   ]  |=====================================================|",
"msg_3": "[----------]          System monitor module",
"msg_4": "[RUN       ]  Setup errors handlers",
"msg_5": "[   INFO   ]  Success",
"msg_6": "[RUN       ]  Init GPIO_PS for leds and button(s)",
"msg_7": "[   INFO   ]  Success",
"msg_8": "[RUN       ]  Init CPU temperature reader",
"msg_9": "[   INFO   ]  Success",
"msg_10": "[RUN       ]  Init temperature sensors reader",
"msg_11": "[   INFO   ]  Success",
"msg_12": "[RUN       ]  System monitor thread",
"msg_13": "[   INFO   ]  Success",
...
```


## Commands for working with device firmware

### ERASE_FLASH

- Function: erasing of internal non-volatile memory of the device (running the command may lead to the inoperability of the device);
- Access: “locked”;
- Payload: no;
- Answer:

```
{
        result: String (*)
}
```

result: displays the result according to the response codes. On successful completion “RF_OK”.

### GET_FIRMWARE

- Function: getting full firmware of the device;
- Access: “unlocked”;
- Payload:

| Option 1 |
| --- |
| No data available (*) |
| If there is no request data, the full version of the firmware is downloaded with reference to saved configuration. |

| Option 2 |
| --- |
| {<br>type: String (*)<br>} |
| *<br>type: identifies the version of downloaded firmware (a set of sectors and files). When set to value “factory”, only firmware files are downloaded that are not associated with the saved configuration of a particular scanner. |

- Answer: array of bytes with data of firmware file.

### SET_FIRMWARE

- Function: setting firmware of the device;
- Access: “unlocked”;
- Payload: array of bytes with data of firmware file;
- Answer:

| {<br>result: String (*)<br>} |
| --- |
| result: displays the result according to the response codes. On successful completion “RF_OK”. |

### SAVE_FIRMWARE

- Function: saving the installed firmware in the non-volatile memory of the device. After saving, the device will automatically reboot;
- Access: “unlocked”;
- Payload: no;
- Answer:

| {<br>result: String (*)<br>} |
| --- |

```
result: displays the result according to the response codes. On successful completion
“RF_OK”.
```

## Commands for working with the periphery

Periphery are the devices that are “external” to the scanner, for example: stepper motor drivers, pneumatic valves, thermostats, etc. Peripheral devices are connected via standard interfaces such as USART, I2C, etc. Commands for working with the periphery ensure transparent exchange without adding or removing any data.

### SEND_TO_PERIPHERY

- Function: sending data to peripheral device;
- Access: “unlocked”;
- Payload:

| Option 1 | |
|---|---|
| { interface: string, payload: blob, wait_answer: bool, answer_timeout: number (uint32_t) } | |
| interface | Specifies the interface where the data will be sent. If the interface is not specified or does not exist, the error “RF_WRONG_ARGUMENT” will occur. Supported values: “Usart0” - sending to the peripheral device connected via the USART0 interface - for the 627th it is RS232. |
| payload | The set of bytes to be sent. If not specified, the error “RF_NO_DATA” will occur. Length limit: 1024 bytes. If the length is exceeded, the error “RF_OUT_OF_BOUNDS” will occur. |
| wait_answer | Waiting for response. Optional parameter. |
| answer_timeout | Response timeout in microseconds. The minimum value is 100 µs. If the requested number of bytes is not received when the specified time expires, the received number will be returned. If not specified or equal to 0, the data will not be expected, the data already in the buffer will be returned. |

- Answer:

```
{
        result: string,
        answer: {
                        result:  string,
                        payload:  blob
                }
}
```

| | |
|---|---|
| result | the result according to the response codes. On successful completion “RF_OK”. |
| answer | object with the result of waiting for a response. Available only if wait_answer = true. |
| answer: result | the result of waiting for a response. |
| answer: payload | The set of bytes that were received. If the data was not received, the field is missing.<br>Length limit: 1024 bytes. |

### RECEIVE_FROM_PERIPHERY

- Function: receiving data from peripheral device;
- Access: “unlocked”;
- Payload:

| Option 1 |
|---|
| {<br>interface: string,<br>count: number (uint16_t),<br>timeout: number (uint32_t)<br>} |

| | |
|---|---|
| interface | Specifies the interface where the data will be sent. If the interface is not specified or does not exist, the error “RF_WRONG_ARGUMENT” will occur.<br>Supported values:<br>“Usart0” - sending to the peripheral device connected via the USART0 interface - for the 627th it is RS232. |
| count | The number of bytes expected to be received. The number of bytes expected to be received. If not specified, the error “RF_NO_DATA” will occur.<br>Length limit: 1024 bytes. If the length is exceeded, the error “RF_OUT_OF_BOUNDS” will occur. |
| timeout | Data timeout in microseconds. The minimum value is 100 µs. If the requested number of bytes is not received when the specified time expires, the received number will be returned. If not specified or equal to 0, the data will not be expected, the data already in the buffer will be returned. |

- Answer:

```
{
        result: string,
        payload: blob
}
```

| | |
|---|---|
| result | the result according to the response codes. On successful completion “RF_OK”. |
| payload | The set of bytes that were received.<br>Length limit: 1024 bytes. If the length is exceeded, the error “RF_OUT_OF_BOUNDS” will occur. |