


# LiDAR API Documentation



This description covers all possibilities to work with the following sensors.

- [multiScan100](#) 

... you have your sensor on hand: [Kick start here](#) 

---

## ► Table of content

- [Introduction](#)
- [General information](#)
  - [IP addresses and ports](#)
  - [Drivers and SDKs](#)
- [Measurement Data Streaming](#)
  - [Comparison and choice of format](#)
  - [COMPACT Format](#)
  - [MSGPACK Format](#)
    - [MSGPACK code examples](#)
      - [MSGPACK C++ Example](#)
      - [MSGPACK Python Example](#)
  - [WebSocket](#)
    - [WebSocket code examples](#)
      - [Python](#)
      - [Node RED](#)
- [Sensor configuration](#)
  - [Choice of configuration method](#)
  - [HTTP / REST](#)
    - [OpenAPI Specification](#)
    - [Challenge and response process](#)
      - [Challenge and Response Python example](#)
    - [Insomnia Collection](#)
  - [CoLa](#)
    - [General information](#)
      - [Variable types](#)
      - [Command basics](#)
      - [Required user level](#)
    - [CoLa B](#)
    - [CoLa A](#)
    - [CoLa telegram overview](#)
    - [CoLa error codes](#)
    - [CoLa code examples](#)
      - [CoLa A](#)
        - [Read a parameter](#)
        - [Python example](#)

- C++ example
  - Write a parameter
    - Python example
    - C++ example
  - CoLa B
    - Read a parameter
      - Python example
      - C++ example
    - Write a parameter
      - Python example
      - C++ example
- [Getting started to work with a sensor](#)
  - [0 Prerequisites](#)
  - [1 See scan data on sensor web server](#)
  - [2 Decide how you want to work with the device](#)
- [FAQ](#)
  - [How to export all settings or parameter?](#)
  - [I can not connect to my sensor? Any ideas?](#)
    - [Change subnet of client system](#)
      - [Windows](#)
      - [Linux](#)
    - [Change sensor IP address](#)
- [Glossary](#)
  - [TCP](#)
  - [UDP](#)
  - [HTTP](#)
  - [REST](#)
  - [OpenAPI](#)
- [Keywords](#)

---

# Introduction

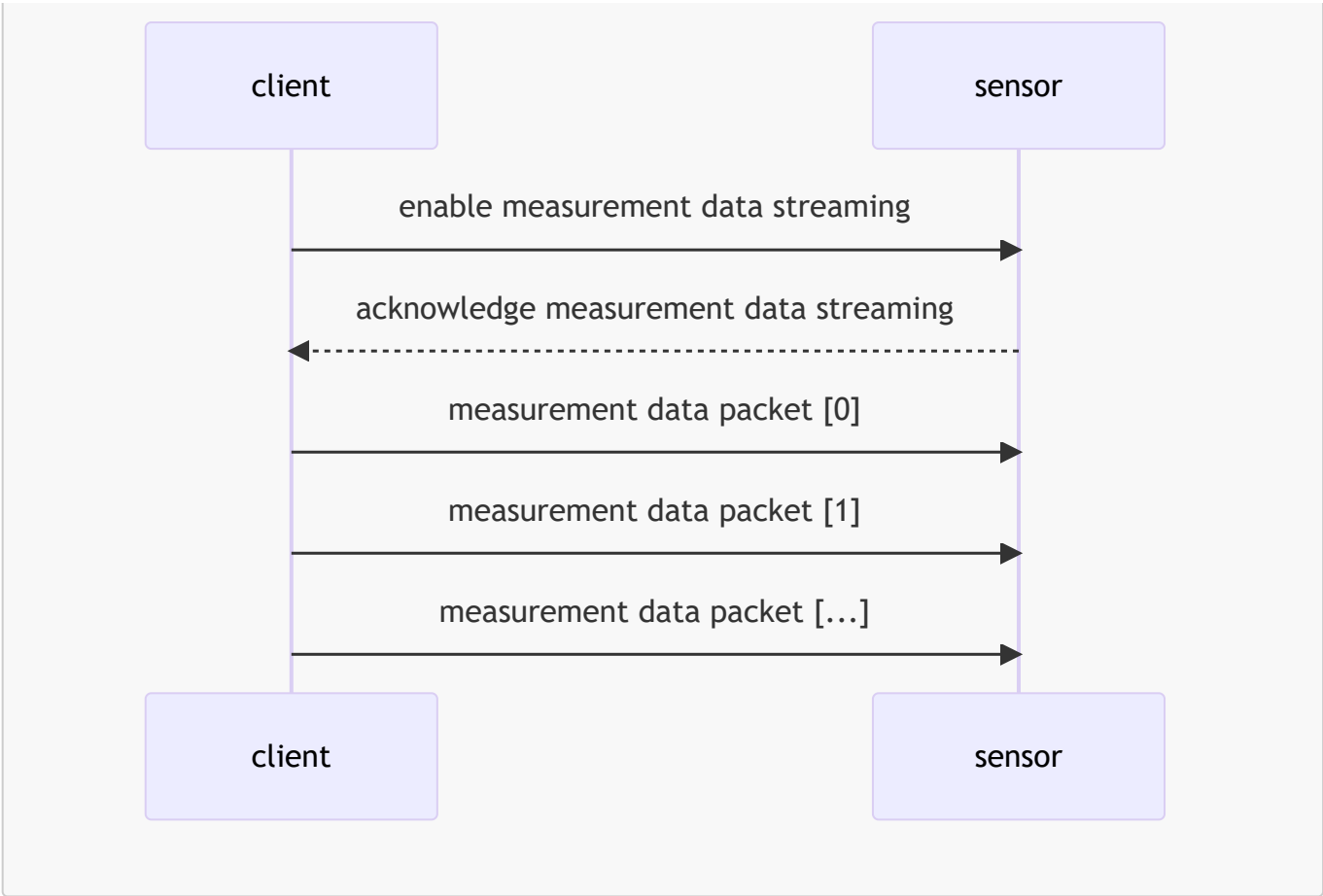
---

The LiDAR API documentation differentiates between two communication concepts. For both cases there are multiple ways to work with the sensor.

## 1. Receive event-driven measurement data streaming

- [COMPACT Format via UDP](#)
- [MSGPACK Format via UDP](#)
- [ScanData via WebSockets](#)

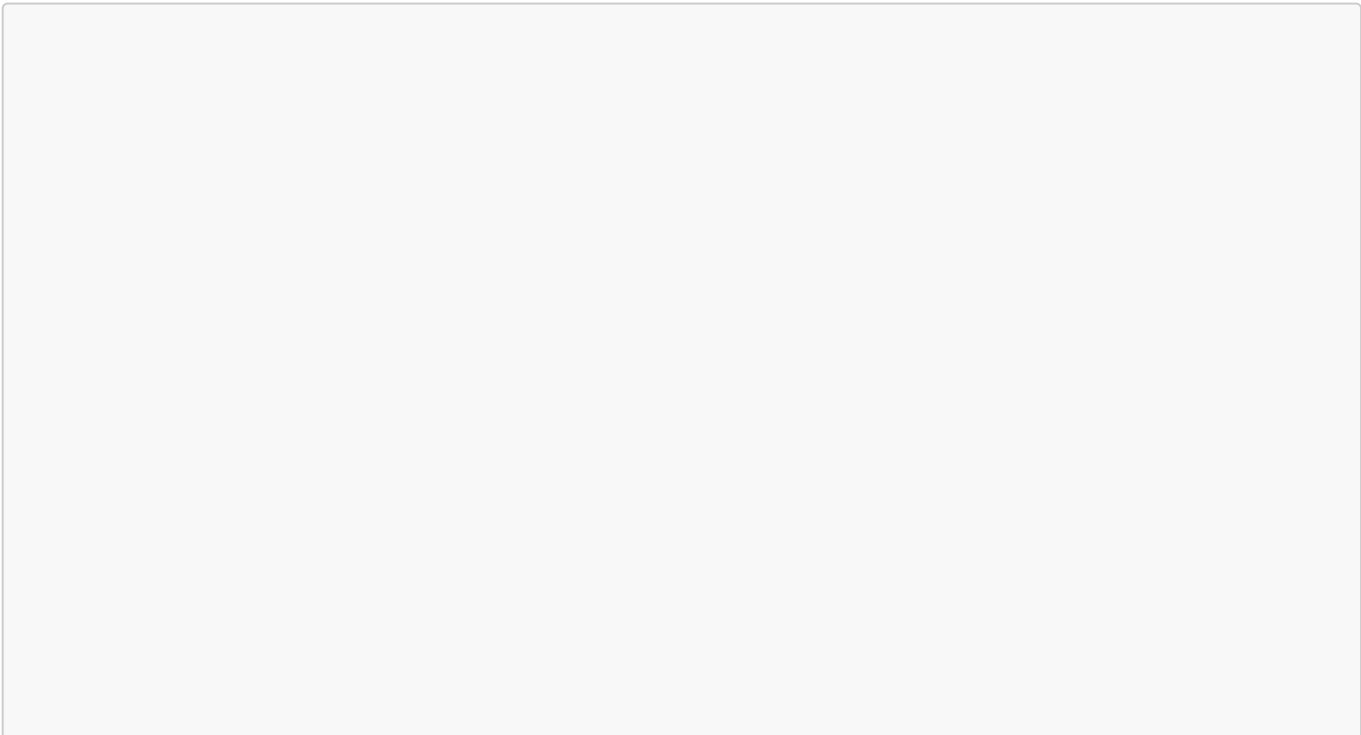
Communication concept of measurement data streaming:

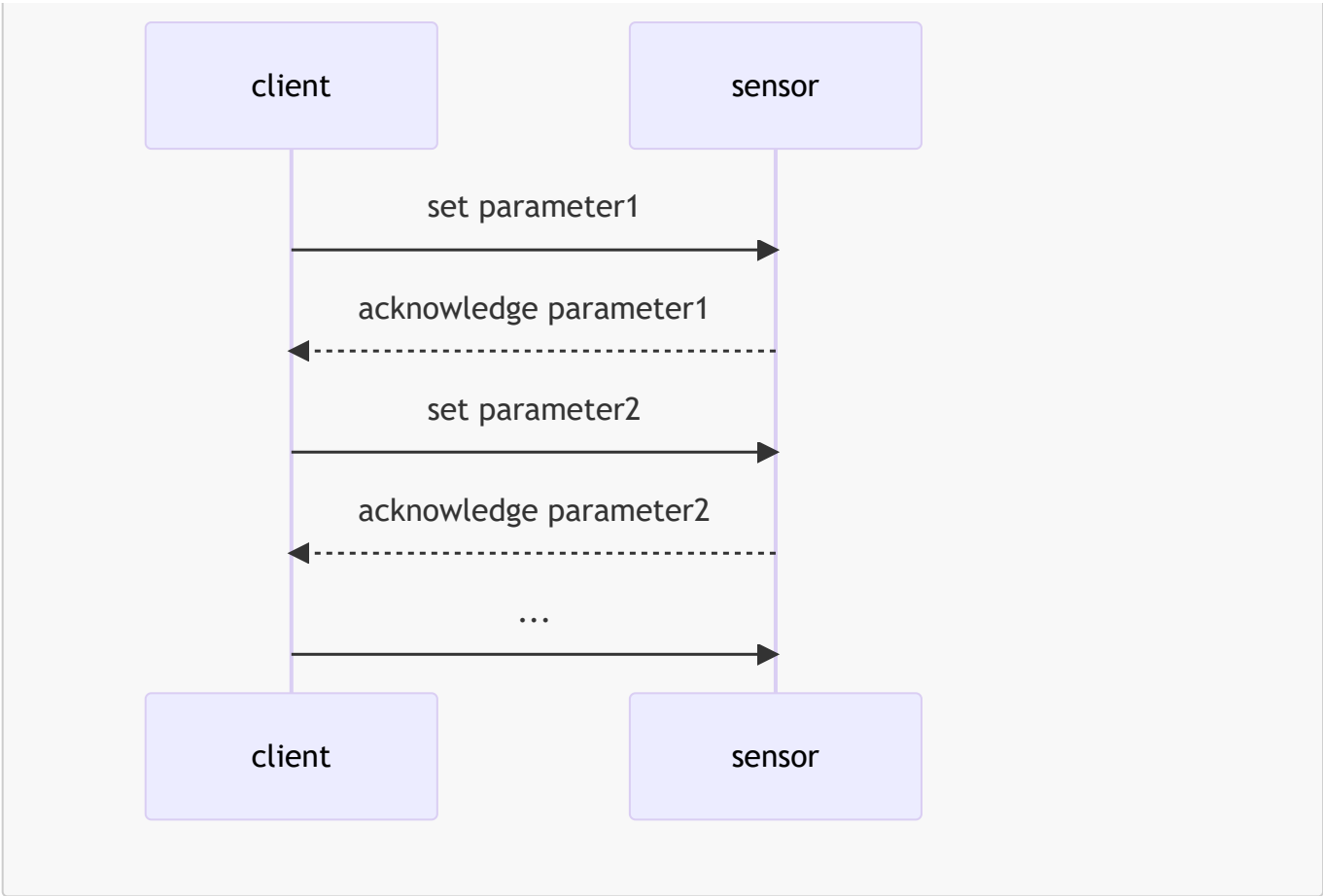


2. Read or write sensor parameters

- [HTTP/REST](#)
- [CoLa](#)

Communication concept of sensor configuration:





# General information

## IP addresses and ports

The default IP address for the sensors (if not specified different is 192.168.0.1)

Usage	Default Port
HTTP	80
WebSocket	80
CoLa A	2111
CoLa B	2112

## Drivers and SDKs

For complete drivers instead of single telegrams, the following options are available:

- [ROS drivers](#)
- [ROS2 drivers](#)
- [Python drivers](#)
- [C++ drivers](#)



# Measurement Data Streaming

## Comparison and choice of format

	Advantages	Bandwidth factor
<b>COMPACT</b>	<ul style="list-style-type: none"><li>- smallest traffic</li><li>- best fit for PLCs</li></ul>	1
<b>MSGPACK</b>	<ul style="list-style-type: none"><li>- available libraries in almost all programming languages</li></ul>	~2
<b>WebSocket</b>	<ul style="list-style-type: none"><li>- uses port 80 which is open in most circumstances</li><li>- good for low output frequencies</li><li>- uses JSON structure</li></ul>	~3

## COMPACT Format

- *works via UDP*
- *more information will follow*

## MSGPACK Format

MSGPACK is a binary data serialization format that is designed to be more efficient and COMPACT than [JSON](#).

One of the main advantages of MSGPACK is its COMPACT binary representation, which can significantly reduce the size of data when compared to JSON. This makes it well-suited for use cases where bandwidth or storage space is limited, such as in embedded systems or mobile devices.

MSGPACK also supports a wide range of data types, including integers, floats, strings, arrays, and maps. This allows it to handle complex data structures with ease. MSGPACK is that it is language-independent, which means that you can use it to communicate between different programming languages. It has implementations for many programming languages like C, C++, C#, D, Go, Java, Lua, Perl, PHP, Python, Ruby, Rust, Scala, Shell, Swift and more.

- *works via UDP*
- *more information will follow*

## MSGPACK code examples

### MSGPACK C++ Example

Here is an example of C++ code that receives a MSGPACK data string via UDP and deserializes it.

#### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.
- You need to include the `msgpack` library in your project to use this code.

not available yet

This code uses the `msgpack` library to deserialize a MSGPACK data string that is received via UDP. The code first creates a `socket` and binds it to a `port`, then enters a loop to continuously receive data from the socket. The received data is passed to the `msgpack::unpack` function, which returns an `object` handle that can be used to access the deserialized data. The code then outputs the deserialized data to the console.

## MSGPACK Python Example

Here is an example of Python code that receives a MSGPACK data string via UDP and deserializes it:

### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.
- For this sample code you need to install the python msgpack module (`$ pip install msgpack`)

not available yet

In this example, the code creates a UDP socket and binds it to the address `192.168.0.1` on port `2115`. It then enters a loop to continuously receive data from the `socket`, deserialize it using the `msgpack` library, and print the deserialized data to the console. The `recvfrom` method is used to receive data from the socket and also obtain the address of the sender. The `unpackb` method from the `msgpack` library is used to deserialize the received data. The `raw=False` argument is used to ensure that the deserialized data is returned as a Python data structure (e.g. dictionary or list) instead of a bytes object.

## WebSocket

WebSockets is a protocol that enables real-time communication between a client and a server (e.g. a streaming sensor). WebSockets use a single, long-lived connection between the client and the server, allowing for fast and efficient communication. This is useful for streaming applications that require near-instant responses. WebSockets are built on top of the same underlying technology as HTTP (Hypertext Transfer Protocol) and can be used in conjunction with it, making it easy to integrate into existing web infrastructure. WebSockets are used for Real-time data visualization or IoT (Internet of Things) applications. Overall, WebSockets technology is useful when you want to build low-latency, real-time applications that require fast and efficient communication between a client and a server.

**NOTE** The distance data is sent as in a Base64 encoded way. Base64 is a group of binary-to-text encoding schemes that represent binary data in an ASCII string format. The primary benefit of Base64 is that it allows you to represent binary data in a text format, which is useful when the data needs to be transmitted or stored in a text-based format. Base64 encoding is used to convert the binary data into a 64-bit encoded string. The encoded data is smaller than the original data, making it more efficient for storage and transmission. After decoding it the distance data is presented in `float32`. It is a data type that represents a 32-bit floating-point number.

## WebSocket code examples

### Python

#### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.
- You can use this sample code in Browsers like Chrome or Edge directly after adapting the IP address if needed (Press **F12** and copy paste the code to the console.)

```
const base64ToArrayBuffer = (base64) => {
  const binaryString = atob(base64);
  const len = binaryString.length;
  const bytes = new Uint8Array(len);
  for (let i = 0; i < len; i++) {
    bytes[i] = binaryString.charCodeAt(i);
  }
  return bytes.buffer;
};

const ws = new WebSocket('ws://192.168.0.1/crownJSON');

ws.onerror = function (error) {
  console.log('Shit happens', error)
};

ws.onmessage = function (msgRaw) {
  const msg = JSON.parse(msgRaw.data);

  if (msg.header.clientId === 1) {
    const handleId = msg.data.handle.id;

    ws.send(`{"header":
{"type":"FunctionCall","clientId":2,"function":"View/Present/setID"},"data":
{"id":"default","handle":{"type":"handle","id":"' + handleId + '"},"options":{}}}')
    ws.send(`{"header":
{"type":"FunctionCall","clientId":3,"function":"View/Present/register"},"data":
{"eventname":"OnPresentLive","handle":{"type":"handle","id":"' + handleId +
'"},"options":{"queue":{"priority":"MID","maxSize":1,"discardIfFull":"OLDEST"}}}')
  }
  else if (msg.header.clientId === 2) {
    // answer setID
  }
  else if (msg.header.clientId === 3) {
    // new Scan Event

    for (const entry of msg.data.viewObject) {
      for (const iconic of entry.data.data.Iconics) {

        const binaryDataOfDistChannel1 =
base64ToArrayBuffer(iconic.data.DistValues[0].data)
```

```

    const distChannelFloat32 = new Float32Array(binaryDataOfDistChannel1);

    // carefully // this line floods the console
    console.log(iconic.class, distChannelFloat32);
  }
}
}

ws.onopen = function () {
  ws.send('{ "header":
{"type": "FunctionCall", "clientId": 1, "function": "View/Present/create"}}');
};

```

## Node RED

### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.

### ► Node-RED code example

```

[
  {
    "id": "d8aa5b41.a5e8a8",
    "type": "tab",
    "label": "picoScan",
    "disabled": false,
    "info": ""
  },
  {
    "id": "7d2321a1.99311",
    "type": "websocket in",
    "z": "d8aa5b41.a5e8a8",
    "name": "picoScan",
    "server": "",
    "client": "e1796c15.80e9b",
    "x": 260,
    "y": 240,
    "wires": [
      [
        "fd271125.4bc94"
      ]
    ]
  },
  {
    "id": "9f6b5bb2.281db8",
    "type": "websocket out",
    "z": "d8aa5b41.a5e8a8",
    "name": "picoScan WebSocket",

```



```

    "server": "",
    "client": "e1796c15.80e9b",
    "x": 880,
    "y": 160,
    "wires": []
  },
  {
    "id": "4f06e15e.845e",
    "type": "function",
    "z": "d8aa5b41.a5e8a8",
    "name": "create",
    "func": "msg.payload = '{\"header\":\n{\n\"type\": \"FunctionCall\", \"clientId\": 1, \"function\": \"View/Present/create\"}}'\n\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 630,
    "y": 160,
    "wires": [
      [
        "9f6b5bb2.281db8"
      ]
    ]
  },
  {
    "id": "8b6189fc.10d788",
    "type": "inject",
    "z": "d8aa5b41.a5e8a8",
    "name": "trigger connection",
    "props": [
      {
        "p": "payload"
      },
      {
        "p": "topic",
        "vt": "str"
      }
    ],
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 290,
    "y": 160,
    "wires": [
      [
        "4f06e15e.845e"
      ]
    ]
  }
]

```

```

    ]
  },
  {
    "id": "f363d3e7.0546a",
    "type": "function",
    "z": "d8aa5b41.a5e8a8",
    "name": "setID/register",
    "func": "decodeBase64 = function(s) {\n    var e=
    {},i,b=0,c,x,l=0,a,r='',w=String.fromCharCode,L=s.length;\n    var
    A=\"ABCDEFGHIIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/\";\n
    for(i=0;i<64;i++){e[A.charAt(i)]=i;}\n    for(x=0;x<L;x++){
    c=e[s.charAt(x)];b=(b<<6)+c;l+=6;\n    while(l>=8){((a=(b>>>(l-=8))&0xff)||
    (x<(L-2)))&&(r+=w(a));}\n    }\n    return r;\n};\n\nconst base64ToArrayBuffer =
    (base64) => {\n    const binaryString = decodeBase64(base64);\n    const len =
    binaryString.length;\n    const bytes = new Uint8Array(len);\n    for (let i = 0; i <
    len; i++) {\n        bytes[i] = binaryString.charCodeAt(i);\n    }\n    return
    bytes.buffer;\n};\n\nif (msg.payload.header.clientId === 1) {\n    var handleId =
    msg.payload.data.handle.id;\n    var msg1 = { payload: '{\"header\":
    {\"type\": \"FunctionCall\", \"clientId\": 2, \"function\": \"View/Present/setID\"}, \"d
    ata\": {\"id\": \"default\", \"handle\": {\"type\": \"handle\", \"id\": ' + handleId +
    '}}, \"options\": {}}' };
    var msg2 = { payload: '{\"header\":
    {\"type\": \"FunctionCall\", \"clientId\": 3, \"function\": \"View/Present/register\"},
    \"data\": {\"eventname\": \"OnPresentLive\", \"handle\": {\"type\": \"handle\", \"id\": '
    + handleId + '}}, \"options\": {\"queue\":
    {\"priority\": \"MID\", \"maxSize\": 1, \"discardIfFull\": \"OLDEST\"}}}' };
    return [[msg1, msg2 ],null]\n }
    \nelse if (msg.payload.header.clientId === 2)
    {\n    msg.payload = '';
    return msg\n}
    \nelse if
    (msg.payload.header.clientId === 3) {\n// new Scan Event\n    for (const entry of
    msg.payload.data.viewObject) {\n        for (const iconic of
    entry.data.data.Iconics) {\n            \n            const binaryDataOfDistChannel1 =
    base64ToArrayBuffer(iconic.data.DistValues[0].data)\n            const
    distChannelFloat32 = new Float32Array(binaryDataOfDistChannel1);\n\n            //
    carefully // this line floods the console\n            msg.payload =
    distChannelFloat32;\n            return msg\n        }\n    }\n}
    },
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 650,
    "y": 240,
    "wires": [
      [
        "d78848e0.359e28",
        "4ad2c046.dfcf8"
      ]
    ]
  }
]
},
{
  "id": "fd271125.4bc94",
  "type": "json",
  "z": "d8aa5b41.a5e8a8",
  "name": "convert to JSON",
  "property": "payload",

```

```

    "action": "",
    "pretty": false,
    "x": 460,
    "y": 240,
    "wires": [
      [
        "f363d3e7.0546a"
      ]
    ]
  },
  {
    "id": "d78848e0.359e28",
    "type": "websocket out",
    "z": "d8aa5b41.a5e8a8",
    "name": "picoScan WebSocket",
    "server": "",
    "client": "e1796c15.80e9b",
    "x": 880,
    "y": 240,
    "wires": []
  },
  {
    "id": "4ad2c046.dfcf8",
    "type": "debug",
    "z": "d8aa5b41.a5e8a8",
    "name": "distanceData as float32 array",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 900,
    "y": 320,
    "wires": []
  },
  {
    "id": "e1796c15.80e9b",
    "type": "websocket-client",
    "path": "ws://10.33.57.52/crownJSON",
    "tls": "",
    "wholemsg": "false"
  }
]

```

---

## Sensor configuration

---

### Choice of configuration method

### When to use what

<b>HTTP / REST</b>	- if you only want to read data - if your system can handle a challenge and response process
<b>CoLa</b>	- if your system cannot handle a challenge and response process - it is the best fit for PLCs.

## HTTP / REST

Background Information: TCP is a transport-layer protocol that ensures reliable data transmission, HTTP is an application-layer protocol that is used to transmit data over the internet, REST is an architectural style for building web services that is based on the principles of HTTP, and OpenAPI specification (formerly known as Swagger) is a specification for building RESTful web services that helps to ensure consistency and ease of use for developers.

## OpenAPI Specification

The use of an OpenAPI Specification brings several advantages.



👍 Improved documentation: OpenAPI provides a standardized format for documenting APIs, making it easier for developers to understand and use the API. It's easy to navigate the endpoints, query parameters, request and response payloads, and other details of an API.

👍 Code generation: OpenAPI specifications can be used to generate code in a variety of programming languages, which can speed up the development process and reduce the risk of errors.

👍 Testing: OpenAPI specifications can be used to test sensors fast and easily.

👍 Interoperability: OpenAPI is an open-source and vendor-neutral specification, which means that APIs written in different languages and frameworks can be described using the same format, making it easier to integrate them with other systems.

👍 Tooling: OpenAPI specification is widely adopted and supported by various tools, frameworks and libraries, which can make it easier to work with an API that has an OpenAPI specification.

👍 Community: OpenAPI has a large community of developers, which means that there is a lot of support and resources available for working with OpenAPI.

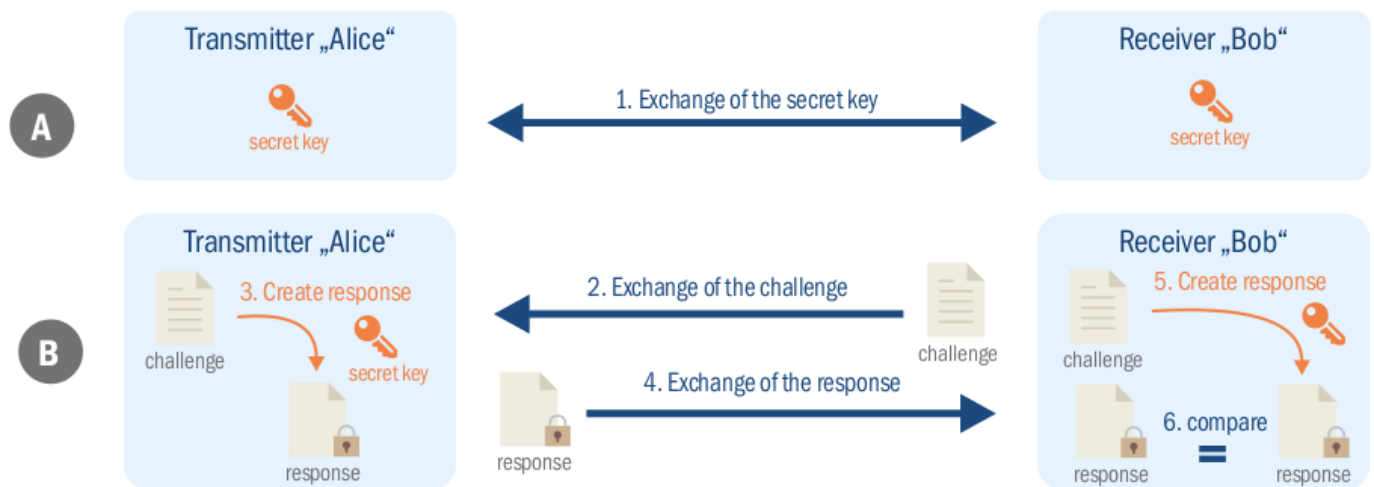
Please refer to the following OpenAPI Specifications. These are used to describe the HTTP interface of the listed sensors. The majority of HTTP requests can be used among all listed sensors. Due to hardware specifics some HTTP requests are only valid for some sensors.

- [General OpenAPI Specification](#)
- [multiScan100 specific OpenAPI Specification](#)

## Challenge and response process

To prevent unauthorized access, the HTTP interfaces underlies an Challenge-response process. It is a participant's secure authentication process based on knowledge. For this purpose, a participant poses a challenge to which others have to respond in order to prove that he or she knows a certain shared secret without having to transmit this information themselves. This is protection against a password from being listened to by attackers on the line.

For this purpose, there are different detailed methods that are based on this basic principle: If one side (Alice) wants to be authenticated with respect to the other side (Bob), Bob sends a random number N (nonce) to Alice (Bob poses the challenge). Alice adds her password to this number N, applies a cryptological hash function or encryption to this combination, and sends the result to Bob (and consequently returns the response). Bob, who knows the random number, the shared secret (= Alice's password) as well as the hash function used or encryption, carries out the same calculation, and compares his result to the response from Alice. If both files are identical, Alice is successfully authenticated.



*Overview challenge and response process*

### Challenge and Response Python example

#### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.
- For this sample code you need to install the python requests module (`$ pip install requests`)

```
import json
import hashlib
import struct
import requests

# login
username = 'Service'
password = 'servicelevel'

def __getAuthPostHeader(url, varname, value):
    # get challenge from sensor
    url = url + 'getChallenge'
```

```

requestPayload = '{ "data": { "user": "' + username + '" } }'
r = requests.post(url, data=requestPayload)
chal = r.json()

# parse challenge response
realm = chal['challenge']['realm']
nonce = chal['challenge']['nonce']
opaque = chal['challenge']['opaque']
hstr1 = username + ":" + realm + ":" + password
ha1 = hashlib.sha256(hstr1.encode()).hexdigest()

# build strings and hashes for header
methodType = 'POST'
hstr2 = methodType + ":" + varname
ha2 = hashlib.sha256(hstr2.encode()).hexdigest()
hstr3 = ha1 + ":" + nonce + ":" + ha2
response = hashlib.sha256(hstr3.encode()).hexdigest()

# build payload
payload = {}
payload['header'] = {}
payload['data'] = {}

# fill payload
payload['header']['user'] = username
payload['header']['realm'] = realm
payload['header']['nonce'] = nonce
payload['header']['response'] = response
payload['header']['opaque'] = opaque
payload['data'][varname] = value

return json.dumps(payload)

#Example
# {
#     "header": {
#         "user": "Service",
#         "realm": "SICK Sensor",
#         "nonce":
"284ba8fbb07d440a9532f751459490e6d8a35a4e9d93695e4d446d0f713451cc",
#         "response":
"c38218aa7601ad13dfb9a9ef556c7102c8a7a84ef7dc54d059e7df6a73ba509a",
#         "opaque":
"9b65dcbb17f5ad588b1220b55e3f31b2c469d5e986d3674ed2d0df2ce54a867"
#     },
#     "data": {
#         "ScanDataEnable": true
#     }
# }

def getVariable(url, varname):
    r = requests.get(url + varname)
    return(r.json())

```

```
def postVariableAuth(url, varname, value):
    payload = __getAuthPostHeader(url, varname, value)
    b = requests.post(url + varname, data=payload)
    return(b.text)

# MAIN -----
url = 'http://192.168.0.1/api/' # must end with '/'

# GET
res = getVariable(url, 'ScanDataEnable')
print(res)

# POST with Authentication
res = postVariableAuth(url, "ScanDataEnable", True)
print(res)

# GET
res = getVariable(url, "ScanDataEnable")
print(res)
# -----
```

## Insomnia Collection

Sometimes it is helpful to test an interface with a REST client like Insomnia. If you want to write a sensor parameter via RES, you must first log in and pass a corresponding header via a challenge response procedure for each write operation.

To make this easier you can use this Insomnia [plugin](#) and import the following Insomnia collection to test the device via the HTTP/REST interface.

```
{ "_type": "export", "__export_format": 4, "__export_date": "2023-01-26T19:35:26.955Z", "__export_source": "insomnia.desktop.app:v2022.7.5", "resources": [ { "_id": "req_wrk_0dfc97e7b56a4e39a705c2dfb199fccb13858317", "parentId": "fld_31fbda9d77a04160b100bc34e5df9f68", "modified": 1674757562626, "created": 1674751360717, "url": "{ _address }/api/DeviceStatus", "name": "DeviceStatus", "description": "", "method": "GET", "body": {}, "parameters": [], "headers": [], "authentication": {}, "metaSortKey": -1674751360717, "isPrivate": false, "settingStoreCookies": true, "settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request" }, { "_id": "fld_31fbda9d77a04160b100bc34e5df9f68", "parentId": "fld_wrk_0dfc97e7b56a4e39a705c2dfb199fccbcbdc0eaf", "modified": 1674757711840, "created": 1674757546624, "name": "Status information", "description": "", "environment": {}, "environmentPropertyOrder": null, "metaSortKey": -1674753579769.4062, "_type": "request_group" }, { "_id": "fld_wrk_0dfc97e7b56a4e39a705c2dfb199fccbcbdc0eaf", "parentId": "wrk_0dfc97e7b56a4e39a705c2dfb199fccb", "modified": 1674751360726, "created": 1674751360726, "name": "picoScan150", "description": "This collection shows the possibilities to work with the picoScan150.", "environment": {}, "environmentPropertyOrder": null, "metaSortKey": -1674751360726, "_type": "request_g" }
```

```
roup"},
{"_id":"wrk_0dfc97e7b56a4e39a705c2dfb199fccb","parentId":null,"modified":167476169
3714,"created":1674742943027,"name":"picoScan150","description":"","scope":"design
","_type":"workspace"},
{"_id":"req_71372fa3b3b64261aa8cb986b14f3397","parentId":"fld_31fbda9d77a04160b100
bc34e5df9f68","modified":1674757584005,"created":1674752724093,"url":"{{ _.address
}}/api/DistMeasState","name":"DistMeasState","description":"","method":"GET","body":
{},"parameters":[],"headers":[],"authentication":
{},"metaSortKey":-1674751360667,"isPrivate":false,"settingStoreCookies":true,"sett
ingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":tr
ue,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"req_1c95a3d4485640809392dfcb01e25a5f","parentId":"fld_de33c621e2414506a365
74cdc437197a","modified":1674754801510,"created":1674754796541,"url":"{{ _.address
}}/api/OrderNumber","name":"OrderNumber","description":"","method":"GET","body":
{},"parameters":[],"headers":[],"authentication":
{},"metaSortKey":-1674754788157,"isPrivate":false,"settingStoreCookies":true,"sett
ingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":tr
ue,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"fld_de33c621e2414506a36574cdc437197a","parentId":"fld_wrk_0dfc97e7b56a4e39
a705c2dfb199fccbcbdbc0eaf","modified":1674757715350,"created":1674754786810,"name":
"Device Information","description":"","environment":
{},"environmentPropertyOrder":null,"metaSortKey":-1674753451560.7969,"_type":"requ
est_group"},
{"_id":"req_7c103ccf24964cc8bab5f3d54408926b","parentId":"fld_de33c621e2414506a365
74cdc437197a","modified":1674754846640,"created":1674754831143,"url":"{{ _.address
}}/api/SerialNumber","name":"SerialNumber","description":"","method":"GET","body":
{},"parameters":[],"headers":[],"authentication":
{},"metaSortKey":-1674754788144.5,"isPrivate":false,"settingStoreCookies":true,"se
ttingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":
true,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"
},
{"_id":"req_59778e295dc44fc38ed37ab1fcdc133d","parentId":"fld_de33c621e2414506a365
74cdc437197a","modified":1674754852872,"created":1674754851025,"url":"{{ _.address
}}/api/FirmwareVersion","name":"FirmwareVersion","description":"","method":"GET","
body":{},"parameters":[],"headers":[],"authentication":
{},"metaSortKey":-1674754788138.25,"isPrivate":false,"settingStoreCookies":true,"s
ettingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl"
:true,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request
"},
{"_id":"req_0e2e3c8158ea4364952752b3088f7c05","parentId":"fld_de33c621e2414506a365
74cdc437197a","modified":1674754817081,"created":1674752549730,"url":"{{ _.address
}}/api/DeviceIdent","name":"DeviceIdent","description":"","method":"GET","body":
{},"parameters":[],"headers":[],"authentication":
{},"metaSortKey":-1674754788132,"isPrivate":false,"settingStoreCookies":true,"sett
ingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":tr
ue,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"req_0c9098507d7f42e19d2663ca2bb7ac81","parentId":"fld_de33c621e2414506a365
74cdc437197a","modified":1674754876485,"created":1674753145497,"url":"{{ _.address
}}/api/LocationName","name":"LocationName","description":"","method":"GET","body":
{},"parameters":[],"headers":[],"authentication":
{},"metaSortKey":-1674754788107,"isPrivate":false,"settingStoreCookies":true,"sett
ingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":tr
ue,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"req_c270107bc02343ad87d8e3dac8f0618f","parentId":"fld_de33c621e2414506a365
```



```

74cdc437197a", "modified": 1674754874603, "created": 1674753174472, "url": "{ _address
}}/api/LocationName", "name": "LocationName", "description": "", "method": "POST", "body"
: {"mimeType": "application/json", "text": "{\n\t\data\": {\n\t\t\"LocationName\":
\"Silas+Zechine\"\n\t\t}\n}", "parameters":
[{"id": "pair_c3784e7cc8464c31ac96f529cd2518c8", "name": "", "value": "", "description":
""}], "headers": [{"name": "Content-
Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1674754788082, "isPrivate": false, "settingStoreCookies": true, "sett
ingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": tr
ue, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request"},
{"_id": "req_634b978bdf3347eba30654324963fc3d", "parentId": "fld_329acd539bde4f208558
29ebcc265e08", "modified": 1674761274888, "created": 1674761251986, "url": "{ _address
}}/api/SetScanConfigList", "name": "SetScanConfigList", "description": "nicht
funktionsfähig", "method": "GET", "body":
{"mimeType": "application/json", "text": ""}, "parameters": [], "headers":
[{"name": "Content-Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1674757497808.8438, "isPrivate": false, "settingStoreCookies": true,
"settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUr
l": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "reque
st"},
{"_id": "fld_329acd539bde4f20855829ebcc265e08", "parentId": "fld_5fa426ff476d471da442
10856ed1d33b", "modified": 1674757675991, "created": 1674757463968, "name": "Basic
Settings", "description": "", "environment":
{}, "environmentPropertyOrder": null, "metaSortKey": -1674757463968, "_type": "request_g
roup"},
{"_id": "fld_5fa426ff476d471da44210856ed1d33b", "parentId": "fld_wrk_0dfc97e7b56a4e39
a705c2dfb199fccbcdbc0eaf", "modified": 1674757720270, "created": 1674757672419, "name":
"Configuration", "description": "", "environment":
{}, "environmentPropertyOrder": null, "metaSortKey": -1674753387456.4922, "_type": "requ
est_group"},
{"_id": "req_edea5a57b19a436b8f23c9166f288ae2", "parentId": "fld_329acd539bde4f208558
29ebcc265e08", "modified": 1674761272756, "created": 1674757634661, "url": "{ _address
}}/api/SetScanConfigList", "name": "SetScanConfigList", "description": "nicht
funktionsfähig", "method": "POST", "body": {"mimeType": "application/json", "text": "
{\n\t\t\"ScanConfigList\": 0\n\t\t}", "parameters": [], "headers": [{"name": "Content-
Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1674757497111.4844, "isPrivate": false, "settingStoreCookies": true,
"settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUr
l": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "reque
st"},
{"_id": "req_6145b595c5a44427b17bc2b5da4b8a13", "parentId": "fld_245189db835741708ad4
020c41193fb8", "modified": 1674756387974, "created": 1674755553770, "url": "{ _address
}}/api/LEDEnable", "name": "LEDEnable", "description": "", "method": "POST", "body":
{"mimeType": "application/json", "text": "{\n\t\t\data\": {\n\t\t\t\"LEDEnable\":
false\n\t\t\t}\n\t\t}", "parameters": [], "headers": [{"name": "Content-
Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1668328962412.625, "isPrivate": false, "settingStoreCookies": true,
"settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUr
l": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "reques
t"},
{"_id": "fld_245189db835741708ad4020c41193fb8", "parentId": "fld_329acd539bde4f208558
29ebcc265e08", "modified": 1674757752296, "created": 1674754417354, "name": "LED", "desc
ription": "", "environment":
{}, "environmentPropertyOrder": null, "metaSortKey": -1674757496414.125, "_type": "reque

```

```

st_group"},
{"_id":"req_8def378177764f5ea33bee3319e25079","parentId":"fld_245189db835741708ad4020c41193fb8","modified":1674755589422,"created":1674755544708,"url":{"_address":"/api/LEDEnable","name":"LEDEnable","description":"","method":"GET","body":{},"parameters":[],"headers":[],"authentication":{},"metaSortKey":-1668328962362.625,"isPrivate":false,"settingStoreCookies":true,"settingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":true,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"req_wrk_0dfc97e7b56a4e39a705c2dfb199fccbe5cbbba9","parentId":"fld_245189db835741708ad4020c41193fb8","modified":1674755591394,"created":1674751360724,"url":{"_address":"/api/LEDState","name":"LEDState","description":"","method":"GET","body":{},"parameters":[],"headers":[],"authentication":{},"metaSortKey":-1668328962312.625,"isPrivate":false,"settingStoreCookies":true,"settingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":true,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"req_ed337257c8d44782bbbc6c225ede37bc","parentId":"fld_af6da995114e4dc79a70e0009a7e9651","modified":1674754980180,"created":1674754975658,"url":{"_address":"/api/FREchoFilter","name":"FREchoFilter","description":"","method":"GET","body":{},"parameters":[],"headers":[],"authentication":{},"metaSortKey":-1674754788150.75,"isPrivate":false,"settingStoreCookies":true,"settingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":true,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"fld_af6da995114e4dc79a70e0009a7e9651","parentId":"fld_329acd539bde4f20855829ebcc265e08","modified":1674757620514,"created":1674754969795,"name":"Filter","description":"","environment":{},"environmentPropertyOrder":null,"metaSortKey":-1674757494554.5,"_type":"request_group"},
{"_id":"req_f9e5ec8c80bf4c5b8901136ff02ac565","parentId":"fld_af6da995114e4dc79a70e0009a7e9651","modified":1674755360476,"created":1674754987542,"url":{"_address":"/api/FREchoFilter","name":"FREchoFilter","description":"\\\"FREchoFilter\\\": 0 = First Echo\\n\\\"FREchoFilter\\\": 1 = All Echos\\n\\\"FREchoFilter\\\": 2 = Last Echo","method":"POST","body":{"mimeType":"application/json","text":{"\\n\\t\\t\\\"FREchoFilter\\\": 2\\n\\t\\t\\n\\n"},"parameters":[],"headers":[{"name":"Content-Type","value":"application/json"}],"authentication":{},"metaSortKey":-1674754788149.9688,"isPrivate":false,"settingStoreCookies":true,"settingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":true,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"req_a03dd6a11a1a40ccbad080fd00b5e379","parentId":"fld_af6da995114e4dc79a70e0009a7e9651","modified":1674755452033,"created":1674755346869,"url":{"_address":"/api/MCSenseLevel","name":"MCSenseLevel","description":"Fog Filter","method":"GET","body":{},"parameters":[],"headers":[],"authentication":{},"metaSortKey":-1674754788149.1875,"isPrivate":false,"settingStoreCookies":true,"settingSendCookies":true,"settingDisableRenderRequestBody":false,"settingEncodeUrl":true,"settingRebuildPath":true,"settingFollowRedirects":"global","_type":"request"},
{"_id":"req_9c245089a41748e1876a6a56404b0bda","parentId":"fld_af6da995114e4dc79a70e0009a7e9651","modified":1674756394103,"created":1674755356591,"url":{"_address":"/api/MCSenseLevel","name":"MCSenseLevel","description":"Fog Filter\\n\\n0 = off","method":"POST","body":{"mimeType":"application/json","text":{"\\n\\t\\t\\\"data\\\":

```

19 / 38

```

: "GET", "body": {}, "parameters": [], "headers": [], "authentication":
{ }, "metaSortKey": -1674752731355.25, "isPrivate": false, "settingStoreCookies": true, "settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request" },
{ "_id": "req_0a9667d44fbf4befb8868abae6b3b919", "parentId": "fld_7f245f36ca564bd48e1f485b15700f24", "modified": 1674754368697, "created": 1674754355906, "url": "{ _address }/api/EtherAddressingMode", "name": "EtherAddressingMode", "description": "", "method": "POST", "body": { "mimeType": "application/json", "text": "{\n\t\t\"data\": {\n\t\t\t\t\"EtherAddressingMode\": 1\n\t\t}\n\t\t}", "parameters": [], "headers": [ { "name": "Content-Type", "value": "application/json" } ], "authentication": { }, "metaSortKey": -1674752727724.125, "isPrivate": false, "settingStoreCookies": true, "settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request" },
{ "_id": "req_7a30dbecbc19460ea0133106e69c7836", "parentId": "fld_102247cc62be4dd2b6819e5cee3addad", "modified": 1674755694964, "created": 1674755693504, "url": "{ _address }/api/ScanDataEnable", "name": "ScanDataEnable", "description": "", "method": "GET", "body": { }, "parameters": [], "headers": [], "authentication": { }, "metaSortKey": -1674752042407, "isPrivate": false, "settingStoreCookies": true, "settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request" },
{ "_id": "fld_102247cc62be4dd2b6819e5cee3addad", "parentId": "fld_38db0d925f49472fba63aab4dedb7f45", "modified": 1674757920977, "created": 1674752722620, "name": "Data output", "description": "", "environment": { }, "environmentPropertyOrder": null, "metaSortKey": -1674752810517.75, "_type": "request_group" },
{ "_id": "fld_38db0d925f49472fba63aab4dedb7f45", "parentId": "fld_wrk_0dfc97e7b56a4e39a705c2dfb199fccbcdbc0eaf", "modified": 1674757906725, "created": 1674757901002, "name": "Application", "description": "", "environment": { }, "environmentPropertyOrder": null, "metaSortKey": -1674753098987.121, "_type": "request_group" },
{ "_id": "req_653298a094a94ab0ae7983591f71c345", "parentId": "fld_102247cc62be4dd2b6819e5cee3addad", "modified": 1674756397882, "created": 1674755700543, "url": "{ _address }/api/ScanDataEnable", "name": "ScanDataEnable", "description": "", "method": "POST", "body": { "mimeType": "application/json", "text": "{\n\t\t\"data\": {\n\t\t\t\t\"ScanDataEnable\": true\n\t\t}\n\t\t}", "parameters": [], "headers": [ { "name": "Content-Type", "value": "application/json" } ], "authentication": { }, "metaSortKey": -1674751701564, "isPrivate": false, "settingStoreCookies": true, "settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request" },
{ "_id": "req_28387acebeeb4d2caae5fa81c67b7955", "parentId": "fld_102247cc62be4dd2b6819e5cee3addad", "modified": 1674755843703, "created": 1674755838529, "url": "{ _address }/api/ScanDataFormat", "name": "ScanDataFormat", "description": "", "method": "GET", "body": { }, "parameters": [], "headers": [], "authentication": { }, "metaSortKey": -1674751701514, "isPrivate": false, "settingStoreCookies": true, "settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request" },
{ "_id": "req_46bda0b83c2942c59b1f51febfa760bb", "parentId": "fld_102247cc62be4dd2b6819e5cee3addad", "modified": 1674755953571, "created": 1674755853396, "url": "{ _address }/api/ScanDataFormat", "name": "ScanDataFormat", "description": "\n\"ScanDataFormat\": 1 = MSGPACK\n\n\"ScanDataFormat\": 2 = COMPACT\n\n", "method": "POST", "body": { "mimeType": "application/json", "text": "{\n\t\t\"data\": {\n\t\t\t\t\"ScanDataFormat\": 0\n\t\t}\n\t\t}", "parameters": [], "headers": [ { "name": "Content-

```



21 / 38

22 / 38

```
{}, "metaSortKey": -1674751360669.5, "isPrivate": false, "settingStoreCookies": true, "settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl": true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request"}, {"_id": "req_2941425c6924464d8432cbb66417d23d", "parentId": "fld_cfe7af1739fb47f3ac5cdff39d9303a3", "modified": 1674758044274, "created": 1674758025820, "url": "{ _address }/api/NotificationList", "name": "NotificationList", "description": "", "method": "GET", "body": {}, "parameters": [], "headers": [], "authentication": {}}, {"_id": "fld_cfe7af1739fb47f3ac5cdff39d9303a3", "parentId": "fld_wrk_0dfc97e7b56a4e39a705c2dfb199fccbcbdbc0eaf", "modified": 1674757829648, "created": 1674757823329, "name": "Diagnostics", "description": "", "environment": {}}, {"environmentPropertyOrder": null, "metaSortKey": -1674751587993.5, "_type": "request_group"}, {"_id": "req_8feb89b0efc347838d4b99f4d4d0cb0e", "parentId": "fld_c5ba5259821b4846b5ee4db412430f44", "modified": 1674754541601, "created": 1674754500459, "url": "{ _address }/api/PowerOnCnt", "name": "PowerOnCnt", "description": "", "method": "GET", "body": {}, "parameters": [], "headers": [], "authentication": {}}, {"_id": "fld_c5ba5259821b4846b5ee4db412430f44", "parentId": "fld_cfe7af1739fb47f3ac5cdff39d9303a3", "modified": 1674757840622, "created": 1674751573799, "name": "Operating Information", "description": "", "environment": {}}, {"environmentPropertyOrder": null, "metaSortKey": -1674751580846.25, "_type": "request_group"}, {"_id": "req_17fe38b9d8154cc29eff12f45a448cf2", "parentId": "fld_c5ba5259821b4846b5ee4db412430f44", "modified": 1674754565638, "created": 1674754559706, "url": "{ _address }/api/OpHours", "name": "OpHours", "description": "", "method": "GET", "body": {}, "parameters": [], "headers": [], "authentication": {}}, {"_id": "req_052f6ff3f41241349a9acf1f0e98c437", "parentId": "fld_c5ba5259821b4846b5ee4db412430f44", "modified": 1674754547608, "created": 1674754546161, "url": "{ _address }/api/DailyOpHours", "name": "DailyOpHours", "description": "", "method": "GET", "body": {}, "parameters": [], "headers": [], "authentication": {}}, {"_id": "req_9e92c29df730467a9b6609acc75caa3f", "parentId": "fld_c5ba5259821b4846b5ee4db412430f44", "modified": 1674754723404, "created": 1674754610392, "url": "{ _address }/api/LSPdatetime", "name": "LSPdatetime", "description": "LSPdatetime returns the time set in the device.", "method": "GET", "body": {}, "parameters": [], "headers": [], "authentication": {}}, {"_id": "req_4526cfa73546458fbf868e266f44e763", "parentId": "fld_24e66e9429894918bbfd430161970d2a", "modified": 1674757077909, "created": 1674755980772, "url": "{ _address
```

```

}}/api/WriteEeprom", "name": "WriteEeprom", "description": "", "method": "POST", "body":
{"mimeType": "application/json", "text": "{\n\n}", "parameters": [], "headers":
[{"name": "Content-Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1674755404030.625, "isPrivate": false, "settingStoreCookies": true,
settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl":
true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request"},
{"_id": "fld_24e66e9429894918bbfd430161970d2a", "parentId": "fld_wrk_0dfc97e7b56a4e39a705c2dfb199fccbcb0eaf", "modified": 1674755991455, "created": 1674755975505, "name":
"Others", "description": "", "environment":
{"environmentPropertyOrder": null, "metaSortKey": -1674751573699, "_type": "request_group"},
{"_id": "req_afc9f92173e54a09b6e275dd8210c2da", "parentId": "fld_24e66e9429894918bbfd430161970d2a", "modified": 1674758538702, "created": 1674758537404, "url": "{ _address
}}/api/RebootDevice", "name": "RebootDevice", "description": "", "method": "POST", "body":
{"mimeType": "application/json", "text": "{\n\n}", "parameters": [], "headers":
[{"name": "Content-Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1674755096093.8125, "isPrivate": false, "settingStoreCookies": true,
"settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl":
true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request"},
{"_id": "req_bf2ad3087bc04f8a961a9eccefa93087", "parentId": "fld_24e66e9429894918bbfd430161970d2a", "modified": 1674758863533, "created": 1674758555892, "url": "{ _address
}}/api/LoadFactoryDefaults", "name": "LoadFactoryDefaults", "description": "", "method":
"POST", "body": {"mimeType": "application/json", "text": "{\n\n}", "parameters":
[], "headers": [{"name": "Content-Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1674754942125.4062, "isPrivate": false, "settingStoreCookies": true,
"settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl":
true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request"},
{"_id": "req_7584c961632c43b9864a81ec9e90ebde", "parentId": "fld_24e66e9429894918bbfd430161970d2a", "modified": 1674758686299, "created": 1674758565745, "url": "{ _address
}}/api/LoadApplicationDefaults", "name": "LoadApplicationDefaults", "description": "",
"method": "POST", "body": {"mimeType": "application/json", "text": "{\n\n\n\n}", "parameters": [], "headers": [{"name": "Content-Type", "value": "application/json"}], "authentication":
{}, "metaSortKey": -1674754865141.2031, "isPrivate": false, "settingStoreCookies": true,
"settingSendCookies": true, "settingDisableRenderRequestBody": false, "settingEncodeUrl":
true, "settingRebuildPath": true, "settingFollowRedirects": "global", "_type": "request"},
{"_id": "env_5c8d779fbc8dfcb8a356b31a26e96f800fb54d02", "parentId": "wrk_0dfc97e7b56a4e39a705c2dfb199fccb", "modified": 1674751526388, "created": 1674742943033, "name": "Base Environment", "data": {"base_url": "192.168.136.1"}, "dataPropertyOrder": {"&":
["base_url"]}, "color": null, "isPrivate": false, "metaSortKey": 1674742943033, "_type": "environment"},
{"_id": "jar_5c8d779fbc8dfcb8a356b31a26e96f800fb54d02", "parentId": "wrk_0dfc97e7b56a4e39a705c2dfb199fccb", "modified": 1674742943035, "created": 1674742943035, "name": "Default Jar", "cookies": [], "_type": "cookie_jar"},
{"_id": "spc_b2368f4bd9b141b4986df9a23564ee6c", "parentId": "wrk_0dfc97e7b56a4e39a705c2dfb199fccb", "modified": 1674761693713, "created": 1674742943027, "fileName": "picoScan150", "contents": "", "contentType": "yaml", "_type": "api_spec"},
{"_id": "uts_a2342cda41434ccf9ca48f8af0ec71e2", "parentId": "wrk_0dfc97e7b56a4e39a705c2dfb199fccb", "modified": 1674752623982, "created": 1674752623982, "name": "Tests all

```



```
picoScan Calls", "_type": "unit_test_suite"},
{"_id": "env_2174061f37d9460ebcbcec3d505500f5", "parentId": "env_5c8d779fbc8dfcb8a356b31a26e96f800fb54d02", "modified": 1674754101092, "created": 1674753615919, "name": "picoScan Env", "data":
{"address": "192.168.136.1:80", "user": "Service", "password": "servicelevel", "enableAuthentication": true}, "dataPropertyOrder": {"&":
["address", "user", "password", "enableAuthentication"]}, "color": null, "isPrivate": false, "metaSortKey": 1674753615920, "_type": "environment"]}]}
```

- Additionally, you can find an Insomnia Collection [here](#) to work with the sensors.

## CoLa

CoLa (Command Language) is SICK proprietary way to serialize and deserialize data. Its a low level and proofed approach for many years. For the listed sensors it utilizes TCP as a transport layer and is used to read and write parameters.

In general we differentiate between CoLa A (ASCII) and CoLa B (binary)

### General information

#### Variable types

Variable type	Length (byte)	Value range	Sign
Bool	1	0 or 1	No
UInt8	1	0 ... 255	No
Int_8	1	-128 ... +127	Yes
UInt16	2	0 ... 65,535	No
Int16	2	-32,768 ... +32,767	Yes
UInt32	4	0 ... 4,294,967,295	No
Int32	4	-2,147,483,648 ... +2,147,483,647	Yes
Enum8	1	Certain values defined in a list	No
Enum16	2	Certain values defined in a list	No
String	Context-dependent	Strings are not terminated in zeroes	

#### Command basics

Description	Value ASCII	Binary Value	Hex Value
Start of text	STX	2	02 02 02 02 + given length
End of text	ETX	3	Calculated checksum
Read	sRN	73524E	73524E

Description	Value ASCII	Binary Value	Hex Value
Write	sWN	73574E	73574E
Method	sMN	734D4E	734D4E
Event	sEN	73454E	73454E
Answer	sRA	735241	735241
Answer	sWA	735741	735741
Answer	sAN	73414E	73414E
Answer	sEA	734541	734541
Answer	sSN	73534E	73534E
Space	SPC	20	20

If values are divided into two parts (e.g. measurement data), they are documented according to LSB (e.g. 00 07), output however is according to MSB (e.g. 07 00).

### Required user level

Whether a parameter can be written or a method can be executed by a user depends on the least user level. Defined user levels are:

# User level	User level	Hash value
1	Operator	-
2	Maintenance	B21ACE26
3	Authorized Client	F4724744
4	Service	81BE23AA

This table indicates which user level you need for which actions.

Task	Required user level
Reading parameters	None
Writing parameters	Authorized Client
Manage password	Service

In general, every sWN command for changing parameters requires to log in to the sensor first. When being logged in, any desired parameter valid for this user level can be changed. All changes become active only after having logged off again from the sensor via the sMN Run command.

### CoLa B

The binary telegram is the basic protocol of the scanner (CoLa B). All values are in hexadecimal code and grouped into pairs of two digits (1 byte). The string consists of four parts: header, data length, data and

**checksum** (CS). The **header** indicates with 4 × **STX** (02 02 02 02) the start of the telegram. The **data length** defines the size of the data part (command part) by indicating the number of digit pairs in the third part. The size of the data length itself is 4 bytes, which means that the data part might have a maximum of 2<sup>32</sup> digit pairs. The **data** part comprises the actual command with letters and characters converted to Hex (according to the **ASCII** chart) and the parameters of either decimal numbers converted to Hex or fixed Hex values with a specific, intrinsic meaning (no conversion). There is always a blank (20<sub>hex</sub>) between the command and the parameters, but not between the different parameter values. The **checksum** finally serves to verify that the telegram has been transferred correctly. The length of the checksum is 1 byte, CRC8. It is calculated with XOR.

This is an example telegram for setting the user level "AuthorizedClient":

	Hex Data	Note
Header	02 02 02 02	
Data length	00 00 00 17	12 <sub>hex</sub> = 23 <sub>dec</sub> digit pairs
Data	73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20	SetAccessMode = actual command for setting the user level (and 20 = blank)
Data	03	03 = fixed Hex value meaning user level "Authorized Client"
Data	F4 72 47 44	F4 72 47 44 = fixed Hex value, serving as password for the selected user level "Authorized Client"
CS	B3	B3 = checksum from XOR calculation

CoLa A

The ASCII telegram is an alternative to the binary telegram. Due to the variable string length of ASCII telegrams, the Binary telegram (CoLa B) is recommended when using scanners with a PLC. The ASCII telegram has the advantage that commands can be written in plaintext. The string consists only of two parts: the framing and the data part. The framing indicates with **STX** and **ETX** the start and stop of each telegram. The data part comprises the actual command with letters and characters (plaintext), parameter values either in decimal (special indicator required) or in hexadecimal (e.g.: a frequency of 25 Hz = +2500<sub>dez</sub> = 9C4<sub>hex</sub>) and fixed hexadecimal values with a specific, intrinsic meaning. As leading zeros are being deleted, there is always a blank required between all command parts and parameter parts.

NOTE The sensor will confirm parameter values always in hexadecimal code, regardless of the code sent.

As further alternative within CoLa A, depending on the preferences of the user, all values can be written directly in Hex. This means however a 1:1 conversion of all letters and characters including numbers and fixed hexadecimal values via the ASCII chart.

This is again an example telegram for setting the user level "Authorized Client". As only fixed hexadecimal parameter values are needed, the option to use parameter values in decimal code with special indicator cannot be applied here:

Hex Data	ASCII Data	Note
02	STX	Framing
73 4D 4E	sMN	start of SOPAS command
20	SPC	Blank
53 65 74 41 63 63 65 73 73 4D 6F 64 65	SetAccessMode	command for setting the user level
20	SPC	Blank
30 33	31	fixed Hex value meaning user level "Authorized Client"
20	SPC	Blank
46 34 37 32 34 37 34 34	F4724744	fixed Hex value, serving as password for the selected user level "Authorized Client"
03	ETX	Framing

## CoLa telegram overview

Please refer to the following CoLa specifications. These are used to describe the CoLa interface of the listed sensors. The majority of CoLa telegrams can be used among all listed sensors. Due to hardware specifics some HTTP requests are only valid for some sensors.

- [General CoLa specification \(not available yet\)](#)
- [multiScan100 specific CoLa specification \(not available yet\)](#)

## CoLa error codes

Error code	Description	Dec	Hex
Ok	No error	0	0
METHODIN_ACCESSDENIED	Wrong user level, access to method not allowed	1	1
METHODIN_UNKNOWNINDEX	Trying to access a method with an unknown index	2	2
VARIABLE_UNKNOWNINDEX	Trying to access a variable with an unknown index	3	3
LOCALCONDITIONFAILED	Local condition violated, e.g. giving a value that exceeds the minimum or maximum allowed value for this variable	4	4
INVALID_DATA	Invalid data given for variable, this error code is deprecated (is not used anymore).	5	5
UNKNOWN_ERROR	An error with unknown reason occurred, this error code is deprecated.	6	6

Error code	Description	Dec	Hex
BUFFER_OVERFLOW	The communication buffer was too small for the amount of data that should be serialized.	7	7
BUFFER_UNDERFLOW	More data was expected, the allocated buffer could not be filled.	8	8
UNKNOWN_TYPE	The variable that shall be serialized has an unknown type. This can only happen when there are variables in the firmware of the sensor that do not exist in the released description of the sensor. This should never happen.	9	9
VARIABLE_WRITE_ACCESSDENIED	It is not allowed to write values to this variable. Probably the variable is defined as read-only.	10	A
UNKNOWN_CMD_FOR_NAMESERVER	When using names instead of indices, a command was issued that the name server does not understand.	11	B
UNKNOWN_COLA_COMMAND	The CoLa protocol specification does not define the given command, command is unknown.	12	C
METHODIN_SERVER_BUSY	It is not possible to issue more than one command at a time to an SRT sensor.	13	D
FLEX_OUT_OF_BOUNDS	An array was accessed over its maximum length.	14	E
EVENTREG_UNKNOWNINDEX	The event you wanted to register for does not exist, the index is unknown.	15	F
COLA_A_VALUE_OVERFLOW	The value does not fit into the value field, it is too large.	16	10
COLA_A_INVALID_CHARACTER	Character is unknown, probably not alphanumeric.	17	11
OSAI_NO_MESSAGE	Only when using SRTOS in the firmware and distributed variables this error can occur. It is an indication that no operating system message could be created. This happens when trying to GET a variable.	18	12
OSAI_NO_ANSWER_MESSAGE	This is the same as OSAI_NO_MESSAGE with the difference that it is thrown when trying to PUT a variable.	19	13
INTERNAL	Internal error in the firmware, probably a pointer to a parameter was null.	20	14
HubAddressCorrupted	The Sopas Hubaddress is either too short or too long.	21	15

Error code	Description	Dec	Hex
HubAddressDecoding	The Sopas Hubaddress is invalid, it can not be decoded (Syntax).	22	16
HubAddressAddressExceeded	Too many hubs in the address	23	17
HubAddressBlankExpected	When parsing a HubAddress an expected blank was not found. The HubAddress is not valid.	24	18
AsyncMethodsAreSuppressed	An asynchronous method call was made although the sensor was built with "AsyncMethodsSuppressed". This is an internal error that should never happen in a released sensor.	25	19
ComplexArraysNotSupported	sensor was built with „ComplexArraysSuppressed“ because the compiler does not allow recursions. But now a complex array was found. This is an internal error that should never happen in a released sensor.	26	20

## CoLa code examples

### CoLa A

#### Read a parameter

##### Python example

#### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.

```
import socket

ip = "192.168.0.1"
port = 2111
STX = "\x02"
ETX = "\x03"

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connection to hostname on the port.
client_socket.connect((ip, port))

# read location name
cola_command = "sRN LocationName"
message = STX + cola_command + ETX
```

```

print("Request: " + message)
message = message.encode('ascii')
client_socket.sendall(message)
data = client_socket.recv(1024)
print("Response: " + data.decode('ascii'))

client_socket.close()

```

#### C++ example

##### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.

**NOTE** C++ is a compiled language meaning your program's source code must be translated (compiled) before it can be run on your computer. On windows machines you may experiment with [Visual Studio Code](#) and [MSYS2](#). You can build the sample code with the following steps on a Windows machine.

- build: `g++ -o colaa-read .\colaa-read.cpp -lwsock32`
- execute: `.\colaa-read.exe`

```

#include <iostream>
#include <WinSock2.h>
#include <ws2tcpip.h>

#pragma comment(lib, "ws2_32.lib")

int main()
{
    // Initialize Winsock
    WSADATA wsaData;
    int result = WSASStartup(MAKEWORD(2, 2), &wsaData);
    if (result != 0) {
        std::cout << "WSAStartup failed: " << result << std::endl;
        return 1;
    }

    // Create a socket
    SOCKET sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sock == INVALID_SOCKET) {
        std::cout << "socket failed: " << WSAGetLastError() << std::endl;
        WSACleanup();
        return 1;
    }

    // Prepare the sockaddr_in structure
    sockaddr_in servAddr;
    servAddr.sin_family = AF_INET;

```

```
servAddr.sin_port = htons(2111);
servAddr.sin_addr.S_un.S_addr = inet_addr("192.168.136.1");

// Connect to server
result = connect(sock, (sockaddr*)&servAddr, sizeof(servAddr));
if (result == SOCKET_ERROR) {
    std::cout << "connect failed: " << WSAGetLastError() << std::endl;
    closesocket(sock);
    WSACleanup();
    return 1;
}

// Build request string
std::string s = "sRN LocationName";
const char length = s.length();
char* sendBuf = new char[length + 2];
sendBuf[0] = (char) 0x02; //attach STX
strcpy((sendBuf+1), s.c_str());
sendBuf[length+1] = (char) 0x03; //attach ETX

std::cout << "Request: ";
for (int i = 0; i < (length+2); i++)
{
    std::cout << sendBuf[i]; //print request
}
std::cout << "\n";

// Send request string
result = send(sock, sendBuf, strlen(sendBuf), 0);
if (result == SOCKET_ERROR) {
    std::cout << "send failed: " << WSAGetLastError() << std::endl;
    closesocket(sock);
    WSACleanup();
    return 1;
}

// Receive the response
char recvBuf[1024];
result = recv(sock, recvBuf, 1024, 0);
if (result > 0) {
    std::cout << "Response: " << std::string(recvBuf, result) << std::endl;
}
else if (result == 0) {
    std::cout << "Connection closed" << std::endl;
}
else {
    std::cout << "recv failed: " << WSAGetLastError() << std::endl;
}

// Cleanup
closesocket(sock);
WSACleanup();
return 0;
}
```



## Write a parameter

### Python example

#### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.

```
import socket

ip = "192.168.0.1"
port = 2111
STX = "\x02"
ETX = "\x03"

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connection to hostname on the port.
client_socket.connect((ip, port))

# LogIn sensor with password "81BE23AA"
cola_command = "sMN SetAccessMode 4 81BE23AA"
message = STX + cola_command + ETX
print("Request: " + message)
message = message.encode('ascii')
client_socket.sendall(message)
data = client_socket.recv(1024)
print("Response: " + data.decode('ascii'))

# write Location name to "Testsensor"
cola_command = "sWN LocationName +10 Testsensor"
message = STX + cola_command + ETX
print("Request: " + message)
message = message.encode('ascii')
client_socket.sendall(message)
data = client_socket.recv(1024)
print("Response: " + data.decode('ascii'))

# LogOut sensor
cola_command = "sMN Run"
message = STX + cola_command + ETX
print("Request: " + message)
message = message.encode('ascii')
client_socket.sendall(message)
data = client_socket.recv(1024)
print("Response: " + data.decode('ascii'))

# read location name
```

```
cola_command = "sRN LocationName"
message = STX + cola_command + ETX
print("Request: " + message)
message = message.encode('ascii')
client_socket.sendall(message)
data = client_socket.recv(1024)
print("Response: " + data.decode('ascii'))

client_socket.close()
```

#### C++ example

##### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.

cpp code

#### CoLa B

##### Read a parameter

#### Python example

##### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.

python code

#### C++ example

##### NOTE

- This is a simple example and you may want to add more error handling and more robustness to your final code.

not available yet

##### Write a parameter

**Python example****NOTE**

- This is a simple example and you may want to add more error handling and more robustness to your final code.

not available yet

**C++ example****NOTE**

- This is a simple example and you may want to add more error handling and more robustness to your final code.

not available yet

---

## Getting started to work with a sensor

---

### 0 Prerequisites

- ✓ SICK sensor (e.g. multiScan100)
- ✓ Power cable
- ✓ Ethernet cable to connect the sensor to your system

### 1 See scan data on sensor web server

- Power the sensor
- Connect the sensor to your system. (FAQ: [I can not connect to my sensor? Any ideas?](#))
- Open the sensor web browser (default ip address: <http://192.168.0.1/>)

### 2 Decide how you want to work with the device

- **Case 1:** You want to test the device and change some settings with the given web browser.
  - You are ready to go and do not need any further actions.
- **Case 2:** You want to integrate the device into a ROS environment (ROS / ROS2)
  - Please refer to our [ROS drivers](#).
- **Case 3:** You want to use an existing driver (C++ or Python)
  - Please refer to our [C++ / Python driver](#).

- **Case 4:** You want to build your own driver in specific programming language.
  - Decide which measurement data streaming approach you want you use. You can choose from these:
    - [COMPACT Format via UDP](#)
    - [MSGPACK Format via UDP](#)
    - [ScanData via WebSockets](#)
  - Decide which sensor configuration approach you want you use. You can choose from these:
    - [HTTP/REST](#)
    - [CoLa](#)

---

## FAQ

---

### How to export all settings or parameter?

Currently there is no automated solution to export all settings.

### I can not connect to my sensor? Any ideas?

Make sure your sensor and client system is in the same subnet. You want to either change the subnet of your ethernet interface or you want to change the IP address of the sensor.

#### Change subnet of client system

You want make sure

**NOTE** It's important to note that changes the IP address on your computer may affect your ability to connect to other sensors on your network or access the internet.

#### Windows

To change the IP address on a Windows computer, you can follow these steps:

- Open the Start menu and search for "Control Panel."
- Click on "Network and Sharing Center" in the Control Panel.
- Click on "Change adapter settings" on the left side of the window.
- Right-click on the network adapter that you want to change the IP address for and select "Properties."
- Select "Internet Protocol Version 4 (TCP/IPv4)" and then click the "Properties" button.
- Select "Use the following IP address" and enter the new IP address, subnet mask, and default gateway.
  - IP address (example): **192.168.0.10**
  - subnet mask (example): **255.255.255.0**
- Click "OK" to save the changes.
- Close all open windows.

#### Linux

- Open the network settings on your Linux distribution, depending on the Linux distribution you are using the location may vary.

- You can find it in the settings, preferences or system settings.
- Once you've found it, select the network adapter you want to change the IP address for.
- enter the new IP address, subnet mask, and default gateway.
  - IP address (example): 192.168.0.10
  - subnet mask (example): 255.255.255.0
- save the changes and close the settings.

## Change sensor IP address

- Open the sensor web browser: http://192.168.0.1/
- Navigate to **Configuration** → **Connection options**
- Change the IP address and the sub net mask

---

# Glossary

---

## TCP

TCP (Transmission Control Protocol) is a transport-layer protocol used to establish and maintain connections between devices on a network. It is responsible for ensuring that data is transmitted reliably and in the correct order, by using a system of acknowledgements and retransmissions.

## UDP

UDP (User Datagram Protocol) is a transport-layer protocol used for communication in a computer network. It is a connectionless protocol, meaning that it does not establish a dedicated connection before sending data, unlike TCP (Transmission Control Protocol). This makes UDP faster and more efficient, but also less reliable because there is no built-in mechanism for error checking and retransmission of lost packets.

## HTTP

HTTP (Hypertext Transfer Protocol) is an application-layer protocol that is used to transmit data over the internet. It is the foundation of the web, and is used by browsers to request and receive information from web servers. HTTP defines a set of request methods, such as GET, POST, PUT, and DELETE, which are used to indicate the desired action to be performed on a specified resource.

## REST

REST (Representational State Transfer) is an architectural style for building web services. It is based on the principles of HTTP and is designed to work with the existing infrastructure of the web. RESTful web services use HTTP methods to indicate the desired action to be performed on a specified resource, and return data in a format that can be easily consumed by a client, such as JSON or XML.

## OpenAPI

OpenAPI is a specification for building RESTful web services. It is used to describe the structure and behavior of an API (Application Programming Interface), including the available endpoints, the request and response formats, and the authentication methods. OpenAPI is a machine-readable format, which means that tools can

be used to generate client libraries, documentation, and other artifacts based on the specification. This makes it easy for developers to understand and interact with an API, and also helps ensure consistency across different implementations.

## Keywords

---

LiDAR SICK LiDAR SICK Laser SICK Laserscanner multiscan multiScan100 multiScan136