SICK
Sensor Intelligence.

Integration & Connectivity:
ScanSegmentApi & ScanSegmentDecoding

Agenda

1. How does SICK customers to develop own applications?
2. ScanSegmentApi for MSGPACK/Compact
   a. What is supported?
   b. Why such a complex preparation?
   c. Deep dive documentation
   d. Understand the programming
3. Links

LiDAR Integration & Connectivity Focus Topic

	Compact	MSGPACK
 **picoScan**	ScanSegmentApi (Python)	ScanSegmentApi & Scan Segment Decoding (Python)
 **multiScan**	ScanSegmentApi (Python)	ScanSegmentApi & Scan Segment Decoding (Python)
 **LRS4000**	ScanSegmentApi (Python)	-
 **RMS**	-	-
 **Others**	-	-

Supported

Scheduled

Not Planned

ScanSegmentApi for MSGPACK/Compact

What is supported by the protocol?

	Distance Data	IMU Data	UDP	TCP
Compact	All	picoScan & multiScan	picoScan & multiScan	LRS4000
MSGPACK	picoScan & multiScan	-	picoScan & multiScan	-

Distance Date includes Distance, Remission, time stamps and reflector bit if available

ScanSegmentApi for MSGPACK/Compact

What is supported by the ScanSegmentApi?

	Distance Data	IMU Data	UDP	TCP
Compact	All	picoScan & multiScan	picoScan & multiScan	LRS4000
MSGPACK	picoScan & multiScan	-	picoScan & multiScan	-

Supported

Scheduled

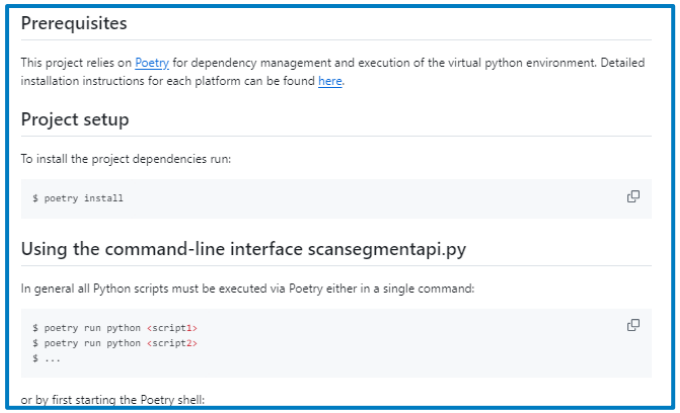
Distance Date includes Distance, Remission, time stamps and reflector bit if available

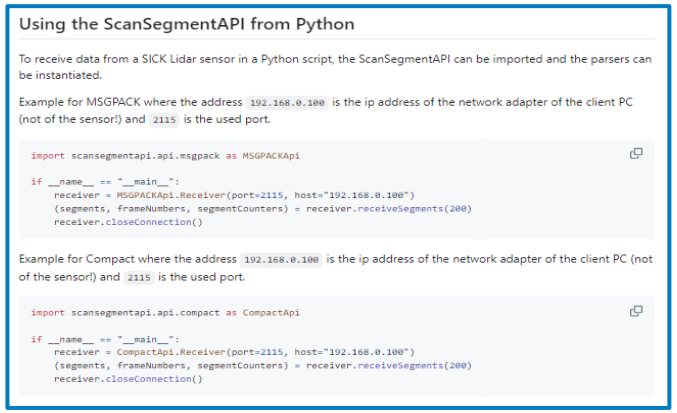
ScanSegmentApi for MSGPACK/Compact

What is part of the ScanSegmentApi?

1. Sourcecode + Project Documentation

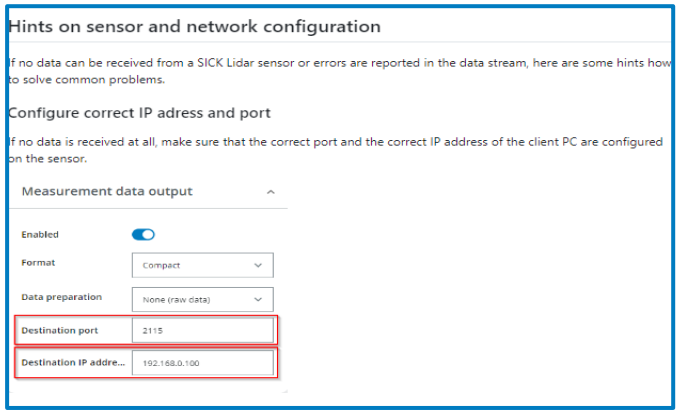
2. Project Setup & Poetry





3. Python API description

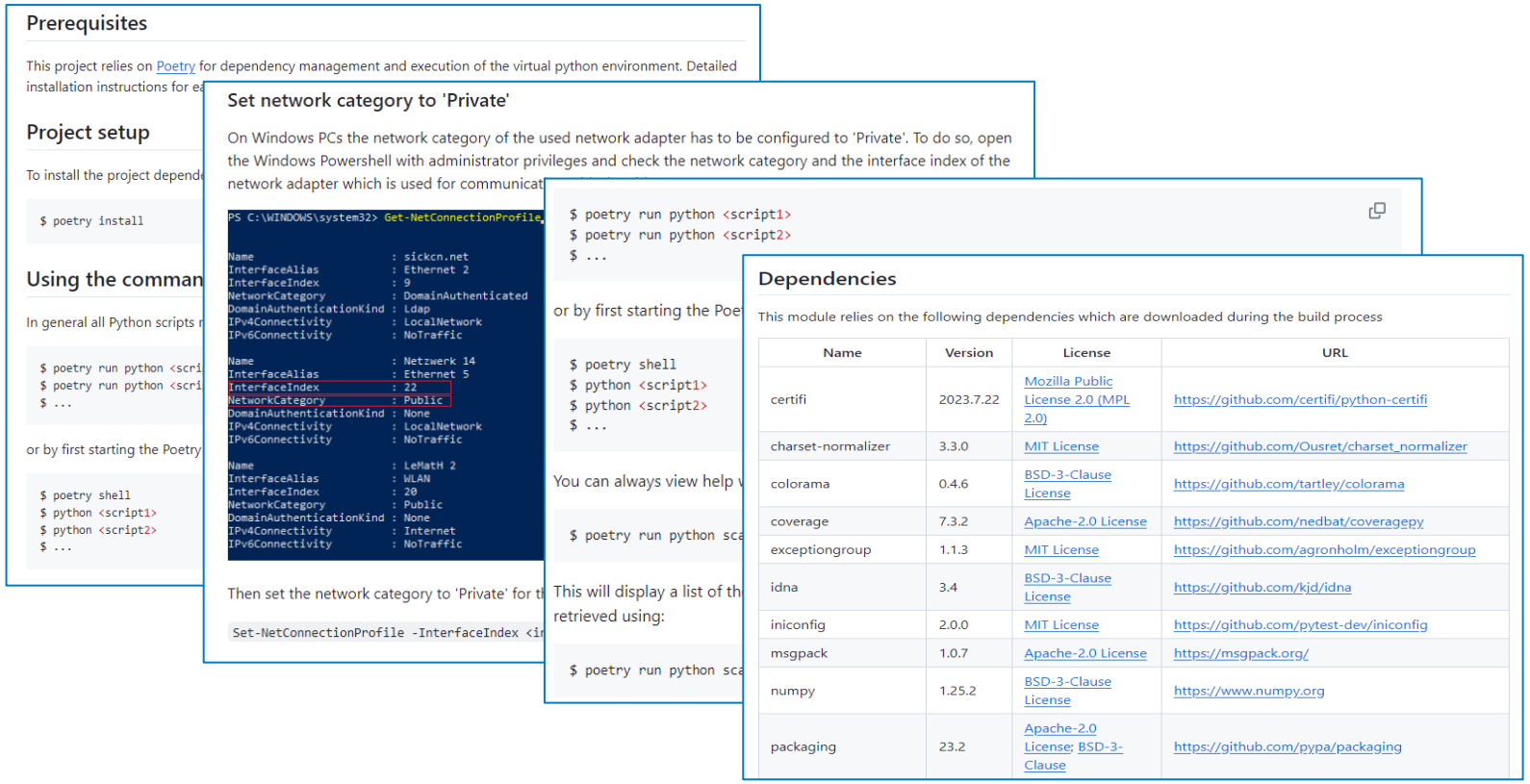
4. Required Network configuration
5. Source Code Dependencies



ScanSegmentApi for MSGPACK/Compact

Why do we need Poetry? 1/2

- ✓ Poetry creates a virtual Python environment
- ✓ Fix software module versions can be used
- → Ensures same system behaviour on all PCs, Operation systems and Python versions



Prerequisites

This project relies on Poetry for dependency management and execution of the virtual python environment. Detailed installation instructions for ea

Project setup

To install the project depend

```
$ poetry install
```

Using the comman

In general all Python scripts r

```
$ poetry run python <scri
$ poetry run python <scri
$ ...
```

or by first starting the Poetry

```
$ poetry shell
$ python <script1>
$ python <script2>
$ ...
```

Set network category to 'Private'

On Windows PCs the network category of the used network adapter has to be configured to 'Private'. To do so, open the Windows Powershell with administrator privileges and check the network category and the interface index of the network adapter which is used for communicat

```
PS C:\WINDOWS\system32> Get-NetConnectionProfile

Name                   : sickcn.net
InterfaceAlias         : Ethernet 2
InterfaceIndex         : 9
NetworkCategory        : DomainAuthenticated
DomainAuthenticationKind : Ldap
IPv4Connectivity       : LocalNetwork
IPv6Connectivity       : NoTraffic

Name                   : Netzwerk 14
InterfaceAlias         : Ethernet 5
InterfaceIndex         : 22
NetworkCategory        : Public
DomainAuthenticationKind : None
IPv4Connectivity       : LocalNetwork
IPv6Connectivity       : NoTraffic

Name                   : LeMatH 2
InterfaceAlias         : WLAN
InterfaceIndex         : 20
NetworkCategory        : Public
DomainAuthenticationKind : None
IPv4Connectivity       : Internet
IPv6Connectivity       : NoTraffic
```

Then set the network category to 'Private' for t

```
Set-NetConnectionProfile -InterfaceIndex <i
```

```
$ poetry run python <script1>
$ poetry run python <script2>
$ ...
```

or by first starting the Poe

```
$ poetry shell
$ python <script1>
$ python <script2>
$ ...
```

You can always view help v

```
$ poetry run python sca
```

This will display a list of th
retrieved using:

```
$ poetry run python sca
```

Dependencies

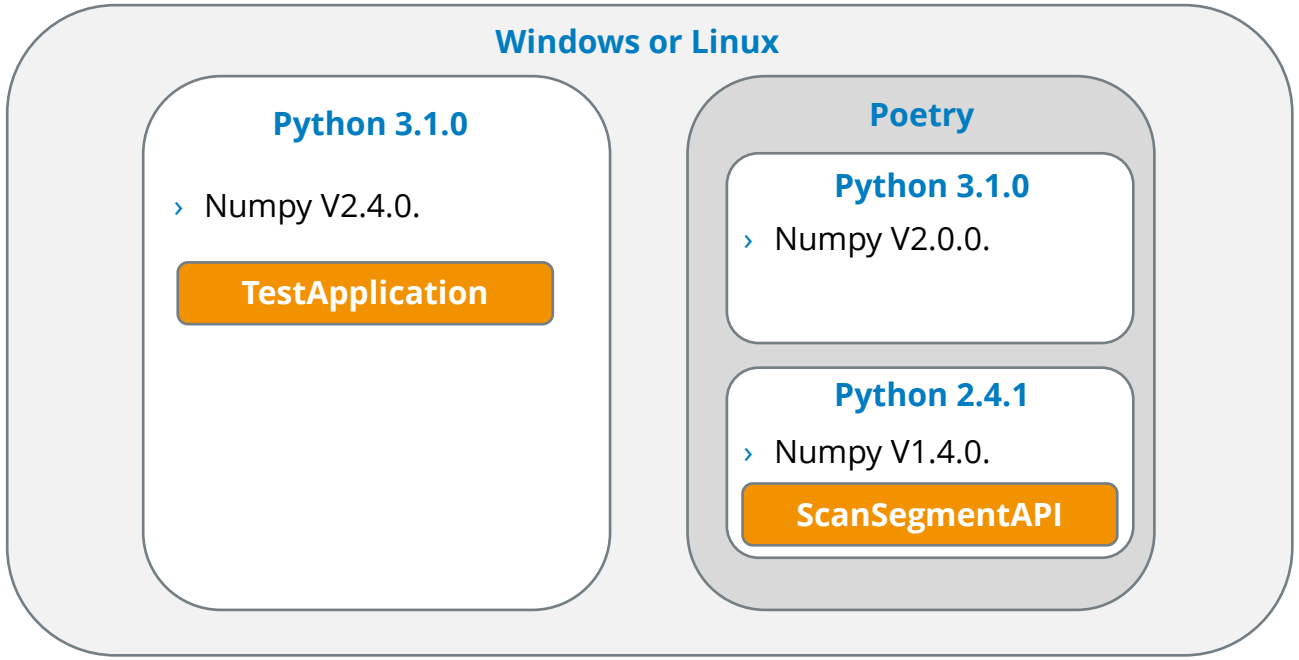
This module relies on the following dependencies which are downloaded during the build process

Name	Version	License	URL
certifi	2023.7.22	Mozilla Public License 2.0 (MPL 2.0)	https://github.com/certifi/python-certifi
charset-normalizer	3.3.0	MIT License	https://github.com/Ousret/charset_normalizer
colorama	0.4.6	BSD-3-Clause License	https://github.com/tartley/colorama
coverage	7.3.2	Apache-2.0 License	https://github.com/nedbat/coveragepy
exceptiongroup	1.1.3	MIT License	https://github.com/agronholm/exceptiongroup
idna	3.4	BSD-3-Clause License	https://github.com/kjd/idna
iniconfig	2.0.0	MIT License	https://github.com/pytest-dev/iniconfig
msgpack	1.0.7	Apache-2.0 License	https://msgpack.org/
numpy	1.25.2	BSD-3-Clause License	https://www.numpy.org
packaging	23.2	Apache-2.0 License; BSD-3-Clause	https://github.com/pypa/packaging

ScanSegmentApi for MSGPACK/Compact

Why do we need Poetry? 2/2

This example shows how poetry works on an example with the Python module numpy.



ScanSegmentApi for MSGPACK/Compact

Deep dive documentation

The description assumes that the user is aware of the Compact/MSGPACK contend as described in the data format description

Name	Last commit message
..	
README.md	Release 2.0.1
print_segment_content.py	Release 2.0.1
process_compact.py	Release 2.0.0
process_msgpack.py	Release 2.0.0
store_segments_json_compact.py	Release 2.0.0
store_segments_json_msgpack.py	Release 2.0.0

Runable Example Code „how to“

ScanSegmentAPI

This project contains scripts written in Python which receive and parse scan segments from SICK LiDAR sensors, either in the Compact or in the MSGPACK format.

The following product families are currently supported:

- multiScan100 (e.g. 1131164)
- picoScan100 (e.g. 1134610)

The scripts which do the actual parsing are located in `api/msgpack.py` and `api/compact.py` respectively. They are written and documented with the intention to support the understanding of the two data formats. The scripts are not optimized for performance and not intended to be used in productive code.

To quickly test that data from a SICK LiDAR sensor is received successfully on your client, the command line tool `scansegmentapi.py` can be used. Hints for the configuration of the sensor and the network of the client PC can be found here.

To use the parsers in a python script, this package can simply be imported as shown in the examples below.

Finally, sample binary files with MSGPACK data and Compact data are provided in ´tests/sample_files´. They can be used to check whether the parsing results of your own parser match with the results of the scripts provided here. Some documentation on the sample binary files is found in tests/sample_files/README.md.

Description of API and setup guide

weinmalSICKAG Release 2.0.1	
api	Release 2.0.0
doc	Release 2.0.0
examples	Release 2.0.1
tests	Release 2.0.0
.gitignore	Release 2.0.0
.gitlab-ci.yml	Release 2.0.1
CHANGELOG.md	Release 2.0.1
LICENSE	Release 2.0.0
README.md	Release 2.0.1
poetry.lock	Release 2.0.1
pyproject.toml	Release 2.0.1
scansegmentapi.py	Release 2.0.0

ScanSegmentApi for MSGPACK/Compact

Deep dive documentation

SICK
Sensor Intelligence.

› Mapping Table refers to protocol fields defined in „data format description“ and describes how to access them using the ScanSegmentAPI

Compact & MSGPACK API mapping table

The following table shows all information contained in a MSGPACK or Compact UDP package as described in the chapters "MSGPACK format" and "Compact format" of the data format description PDF which can be downloaded from sick.com.

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment

Compact	Data access to Compact packages using the ScanSegmentAPI	MSGPACK	Data access to MSGPACK packages using the ScanSegmentAPI
Header: StartOfFrame	-	Framing: StartOfFrame	-
Header: CommandId	segment["CommandId"]	-	-
-	-	Framing: MSGPACK buffer	-
Header: TelegramCounter	segment["TelegramCounter"]	ScanSegment: TelegramCounter	segment["TelegramCounter
Header: TimeStampTransmit	segment["TimestampTransmit"]	ScanSegment: TimeStampTransmit	segment["TimestampTransn
Header: TelegramVersion	segment["Version"]	-	-
Header: SizeModule0	-	-	-
MetaData Module *M*: SegmentCounter	segment["Modules"][*M*] ["SegmentCounter"]	ScanSegment: SegmentCounter	segment["SegmentCounter
MetaData Module *M*: FrameNumber	segment["Modules"][*M*] ["FrameNumber"]	ScanSegment: FrameNumber	segment["FrameNumber"]

ScanSegmentApi for MSGPACK/Compact

Deep dive documentation

SICK
Sensor Intelligence.

› Legend contains the used abbreviation to describe a index of the frame

Compact & MSGPACK API mapping table

The following table shows all information contained in a MSGPACK or Compact UDP package as described in the chapters "MSGPACK format" and "Compact format" of the data format description PDF which can be downloaded from sick.com.

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment

Compact	Data access to Compact packages using the ScanSegmentAPI	MSGPACK	Data access to MSGPACK packages using the ScanSegmentAPI
Header: StartOfFrame	-	Framing: StartOfFrame	-
Header: CommandId	segment["CommandId"]	-	-
-	-	Framing: MSGPACK buffer	-
Header: TelegramCounter	segment["TelegramCounter"]	ScanSegment: TelegramCounter	segment["TelegramCounter
Header: TimeStampTransmit	segment["TimestampTransmit"]	ScanSegment: TimeStampTransmit	segment["TimestampTransn
Header: TelegramVersion	segment["Version"]	-	-
Header: SizeModule0	-	-	-
MetaData Module *M*: SegmentCounter	segment["Modules"][*M*] ["SegmentCounter"]	ScanSegment: SegmentCounter	segment["SegmentCounter
MetaData Module *M*: FrameNumber	segment["Modules"][*M*] ["FrameNumber"]	ScanSegment: FrameNumber	segment["FrameNumber"]

ScanSegmentApi for MSGPACK/Compact

Deep dive documentation

SICK
Sensor Intelligence.

› Symbol „ : “ is used to show sub structure
e.g. „StartOfFrame“ is part of „Header“

Compact & MSGPACK API mapping table

The following table shows all information contained in a MSGPACK or Compact UDP package as described in the chapters "MSGPACK format" and "Compact format" of the data format description PDF which can be downloaded from sick.com.

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment

Compact	Data access to Compact packages using the ScanSegmentAPI	MSGPACK	Data access to MSGPACK packages using the ScanSegmentAPI
Header: StartOfFrame	-	Framing: StartOfFrame	-
Header: CommandId	segment["CommandId"]	-	-
-	-	Framing: MSGPACK buffer	-
Header: TelegramCounter	segment["TelegramCounter"]	ScanSegment: TelegramCounter	segment["TelegramCounter
Header: TimeStampTransmit	segment["TimestampTransmit"]	ScanSegment: TimeStampTransmit	segment["TimestampTransn
Header: TelegramVersion	segment["Version"]	-	-
Header: SizeModule0	-	-	-
MetaData Module *M*: SegmentCounter	segment["Modules"][*M*]["SegmentCounter"]	ScanSegment: SegmentCounter	segment["SegmentCounter
MetaData Module *M*: FrameNumber	segment["Modules"][*M*]["FrameNumber"]	ScanSegment: FrameNumber	segment["FrameNumber"]

ScanSegmentApi for MSGPACK/Compact

Deep dive documentation

SICK
Sensor Intelligence.

› Mapping Table column „Compact“ and „MSGPACK“ refer to protocol fields defined in „data format description“

Compact & MSGPACK API mapping table

The following table shows all information contained in a MSGPACK or Compact UDP package as described in the chapters "MSGPACK format" and "Compact format" of the data format description PDF which can be downloaded from sick.com.

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment

Compact	Data access to Compact packages using the ScanSegmentAPI	MSGPACK	Data access to MSGPACK packages using the ScanSegmentAPI
Header: StartOfFrame	-	Framing: StartOfFrame	-
Header: CommandId	segment["CommandId"]	-	-
-	-	Framing: MSGPACK buffer	-
Header: TelegramCounter	segment["TelegramCounter"]	ScanSegment: TelegramCounter	segment["TelegramCounter
Header: TimeStampTransmit	segment["TimestampTransmit"]	ScanSegment: TimeStampTransmit	segment["TimestampTransn
Header: TelegramVersion	segment["Version"]	-	-
Header: SizeModule0	-	-	-
MetaData Module *M*: SegmentCounter	segment["Modules"][*M*]["SegmentCounter"]	ScanSegment: SegmentCounter	segment["SegmentCounter
MetaData Module *M*: FrameNumber	segment["Modules"][*M*]["FrameNumber"]	ScanSegment: FrameNumber	segment["FrameNumber"]

ScanSegmentApi for MSGPACK/Compact

Deep dive documentation

SICK
Sensor Intelligence.

› Other colums show example code how to acces data

Compact & MSGPACK API mapping table

The following table shows all information contained in a MSGPACK or Compact UDP package as described in the chapters "MSGPACK format" and "Compact format" of the data format description PDF which can be downloaded from sick.com.

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment

Compact	Data access to Compact packages using the ScanSegmentAPI	MSGPACK	Data access to MSGPACK packages using the ScanSegmentAPI
Header: StartOfFrame	-	Framing: StartOfFrame	-
Header: CommandId	segment["CommandId"]	-	-
-	-	Framing: MSGPACK buffer	-
Header: TelegramCounter	segment["TelegramCounter"]	ScanSegment: TelegramCounter	segment["TelegramCounter'
Header: TimeStampTransmit	segment["TimestampTransmit"]	ScanSegment: TimeStampTransmit	segment["TimestampTransn
Header: TelegramVersion	segment["Version"]	-	-
Header: SizeModule0	-	-	-
MetaData Module *M*: SegmentCounter	segment["Modules"][*M*]["SegmentCounter"]	ScanSegment: SegmentCounter	segment["SegmentCounter'
MetaData Module *M*: FrameNumber	segment["Modules"][*M*]["FrameNumber"]	ScanSegment: FrameNumber	segment["FrameNumber"]

ScanSegmentApi for MSGPACK/Compact

Deep dive documentation

SICK
Sensor Intelligence.

› Names in „“ are Python keys

Compact & MSGPACK API mapping table

The following table shows all information contained in a MSGPACK or Compact UDP package as described in the chapters "MSGPACK format" and "Compact format" of the data format description PDF which can be downloaded from sick.com.

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment

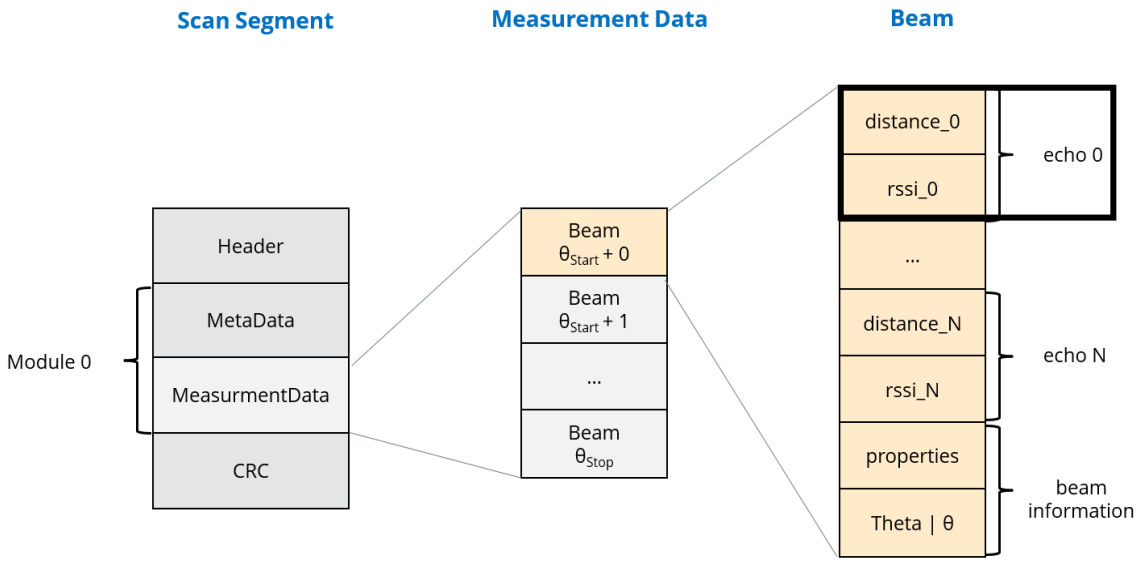
Compact	Data access to Compact packages using the ScanSegmentAPI	MSGPACK	Data access to MSGPACK packages using the ScanSegmentAPI
Header: StartOfFrame	-	Framing: StartOfFrame	-
Header: CommandId	segment["CommandId"]	-	-
-	-	Framing: MSGPACK buffer	-
Header: TelegramCounter	segment["TelegramCounter"]	ScanSegment: TelegramCounter	segment["TelegramCounter
Header: TimeStampTransmit	segment["TimestampTransmit"]	ScanSegment: TimeStampTransmit	segment["TimestampTransn
Header: TelegramVersion	segment["Version"]	-	-
Header: SizeModule0	-	-	-
MetaData Module *M*: SegmentCounter	segment["Modules"][*M*]["SegmentCounter"]	ScanSegment: SegmentCounter	segment["SegmentCounter
MetaData Module *M*: FrameNumber	segment["Modules"][*M*]["FrameNumber"]	ScanSegment: FrameNumber	segment["FrameNumber"]

ScanSegmentApi for MSGPACK/Compact

Legend view picoScan

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment



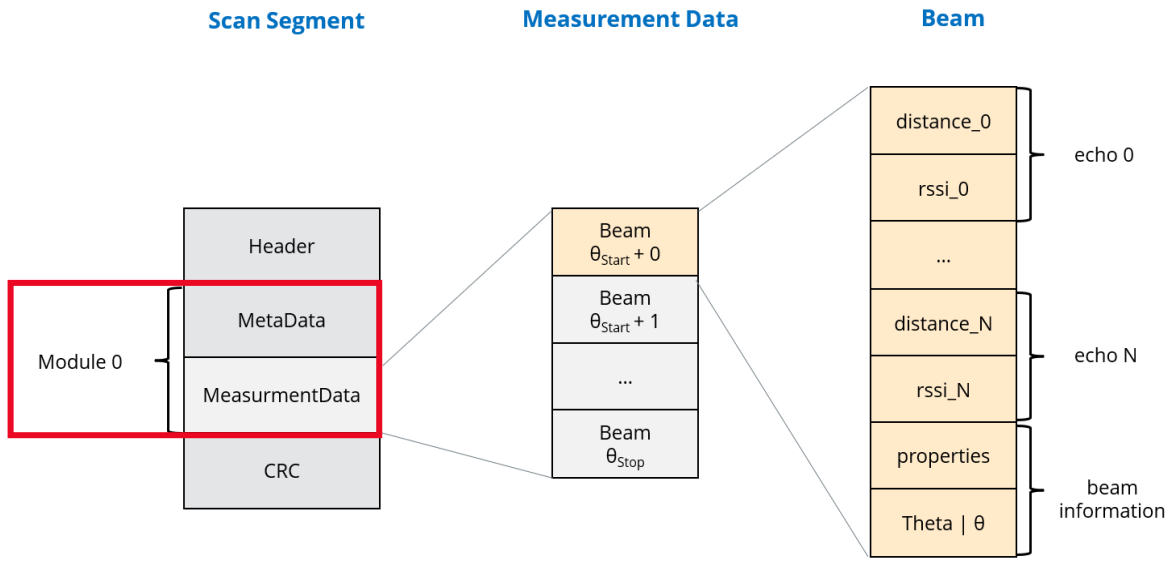
Module „M“ and scan number „SC“ and „SM“ are always set to **0** since picoScan has only one module and one scan beam set

ScanSegmentApi for MSGPACK/Compact

Legend view picoScan

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment



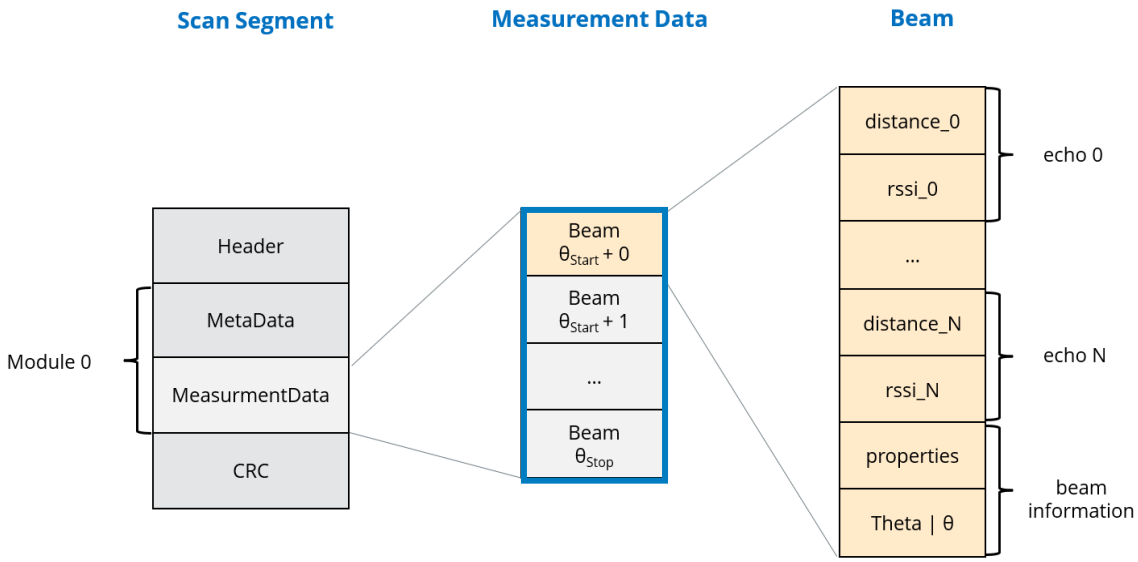
Module „M“ and scan number „SC“ and „SM“ are always set to **0** since picoScan has only one module and one scan beam set

ScanSegmentApi for MSGPACK/Compact

Legend view picoScan

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment



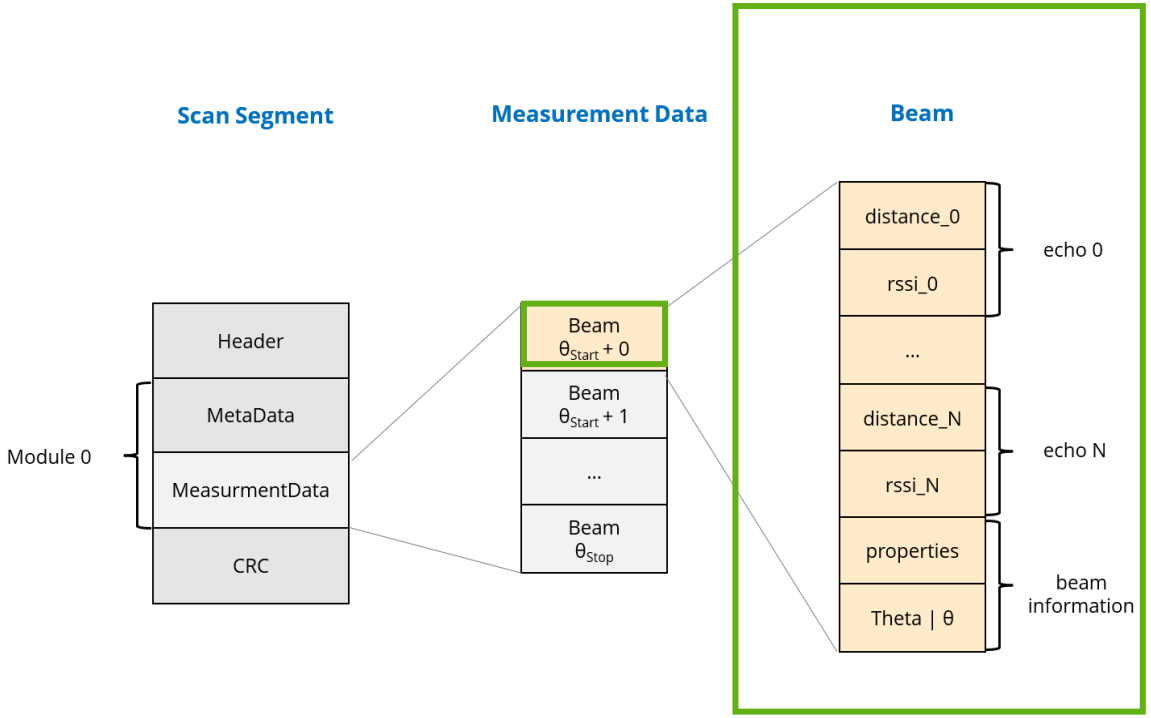
Module „M“ and scan number „SC“ and „SM“ are always set to **0** since picoScan has only one module and one scan beam set

ScanSegmentApi for MSGPACK/Compact

Legend view picoScan

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment



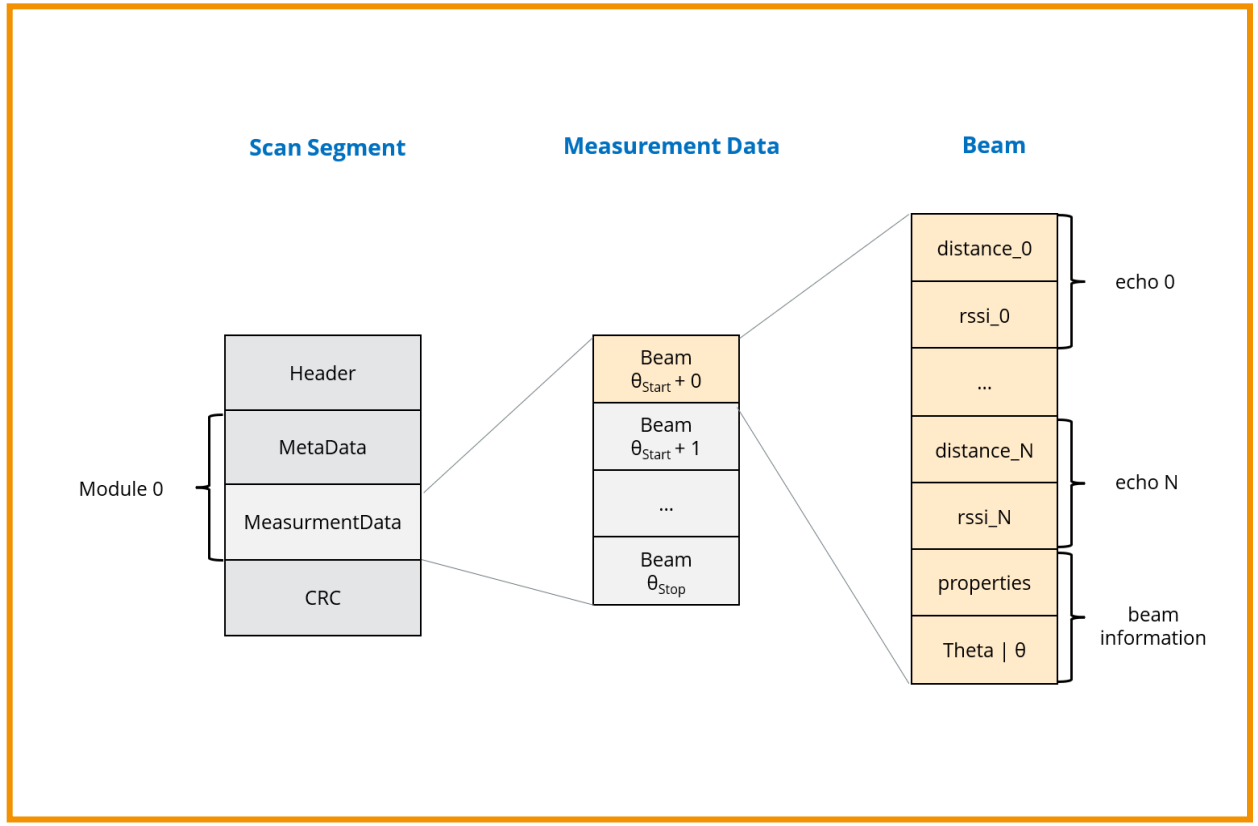
Module „M“ and scan number „SC“ and „SM“ are always set to **0** since picoScan has only one module and one scan beam set

ScanSegmentApi for MSGPACK/Compact

Legend view picoScan

Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment



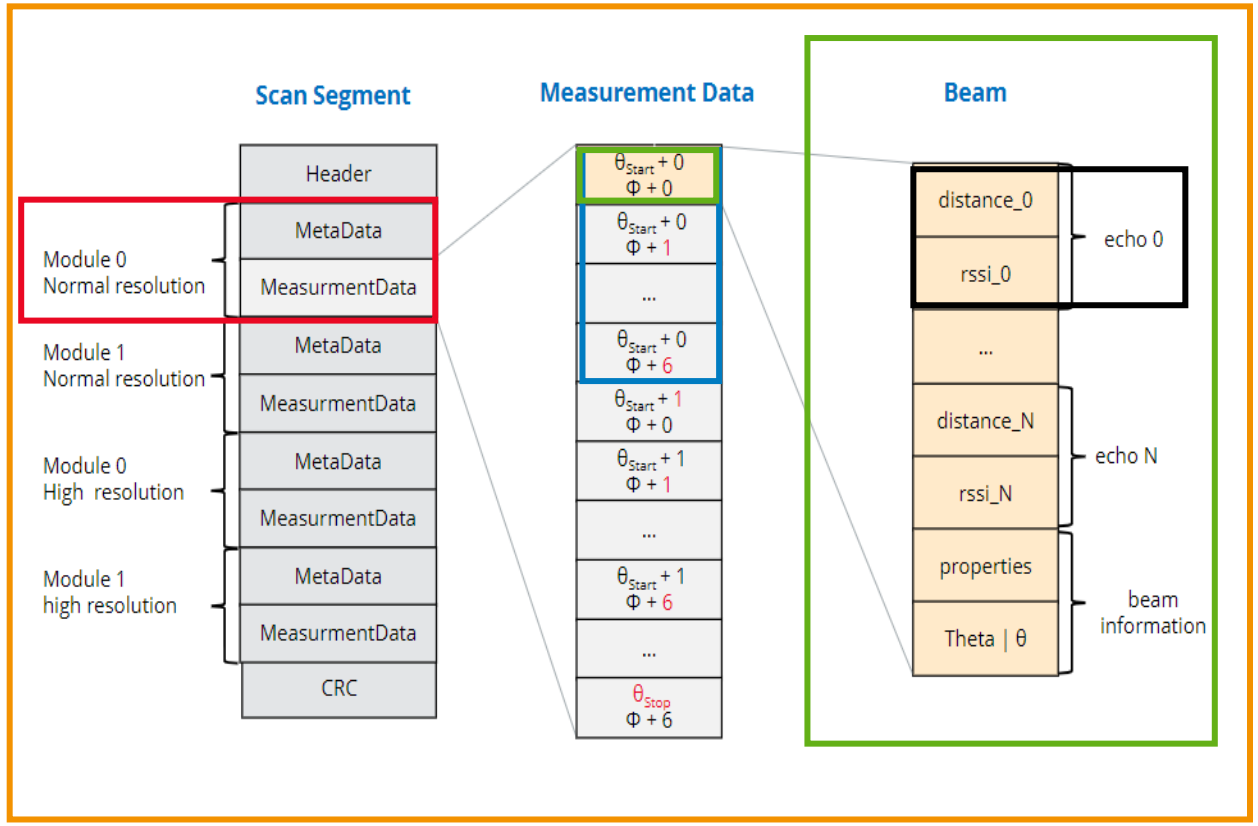
Module „M“ and scan number „SC“ and „SM“ are always set to **0** since picoScan has only one module and one scan beam set

ScanSegmentApi for MSGPACK/Compact

Legend view multiScan

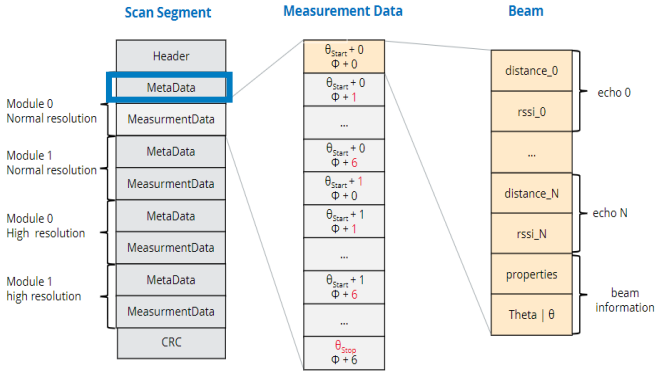
Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment



ScanSegmentApi for MSGPACK/Compact

Deep dive documentation



Legend:

- *E* stands for a specific echo number
- *M* stands for a specific module number
- *SC* stands for a specific scan number within a module, which is used in the Compact format
- *SM* stands for a specific scan number within a segment, which is used in the MSGPACK format
- *B* stands for a specific beam
- *segment* is a single received segment

Compact	Data access to Compact packages using the ScanSegmentAPI	MSGPACK	Data access to MSGPACK packages using the ScanSegmentAPI
MetaData Module *M*: FrameNumber	segment["Modules"][*M*]["FrameNumber"]	ScanSegment: FrameNumber	segment["FrameNumber"]

Reference to „data format description“

5.3.1 Meta data

This section describes the metadata of a module. The names of the individual data fields in the following table are for orientation purposes only. Since the data are present as a byte sequence only, the names are not used explicitly.

Table 11: Metadata of a module for the Compact format

Name	Size	Type	Description
SegmentCounter	8 bytes	uint64	Segment counter as described in section 6.1.
FrameNumber	8 bytes	uint64	Counts the number of full revolutions since the device was started.
SenderId	4 bytes	uint32	Device serial code. It can be used to detect on the receiver which sensor the data was sent from.
numberOfLinesInModule	4 bytes	uint32	Number of layers contained in one module, see figure 11.

4.3.2 Serialization of the “Scan segment” class

The ScanSegment class is represented as a nesting of MSGPACK maps as follows:

```
msgpack_map_header {
"classname": ScanSegment,
"data":
msgpack_map_header {
"TelegramCounter": <telegramCounter>,
"TimeStampTransmit": <timeStampTransmit>,
"SegmentCounter": <segmentCounter>,
"FrameNumber": <frameNumber>,
"Availability": <availability>,
"SenderId": <senderId>,
"LayerId": layerIdVector,,
"SegmentData", segmentData
}
}
```

Definition: Definition of the ScanSegment class

The meaning of the individual fields is shown in the following table.

Table 6: Description of the attributes of the ScanSegment class

Name	Type	Description
TelegramCounter	MSGPACK int [1]	Counts all telegrams with measurement data sent in MSGPACK format since switching on the device. The counter starts at 1.
TimeStampTransmit	MSGPACK int [1]	Sensor system time in µs since 1.1.1970 00:00 in UTC.
SegmentCounter	MSGPACK int [1]	Segment counter as described in section.
FrameNumber	MSGPACK int [1]	Counts the number of full revolutions since the device was started.

Get used to it

TASK: Reflector Bit Verification

Configure a picoScan to send Compact data to your PC and write a Python program with the following requirements:

- Program should run with picoScan.
- Compact protocol should be used.
- 200 Segments should be received.
- Print distance, RSSI and angle of each beam that hits a reflector.
- In case your device is unable to detect reflectors, print the values at angular 0° instead.
- Consider only the first echo.
- Store the printed information in an additional .txt file (optional)

Put a reflector on your table in case you want to use the reflector bit

Links

SICK
Sensor Intelligence.

- ScanSegmentApi https://github.com/SICKAG/ScanSegmentAPI
- ScanSegmentDecoding https://github.com/SICKAG/ScanSegmentDecoding
- ScanXD https://github.com/SICKAG/sick_scan_xd
- CSK https://github.com/SICKAG/SICK_AppSpace_CodingStarterKit
- ScanRestClient https://github.com/SICKAG/sick_scan_rest_client