™

Firmware User Manual

Firmware v2.4.0 for all Ouster sensors

Ouster

Sep 21, 2022

Contents

1 Important Safety Information

1.1 Safety & Legal Notices

All Ouster sensors have been evaluated to be **Class 1 laser products** per **60825-1: 2014 (Ed. 3)** and operate in the 865nm band.

Tous les capteurs Ouster répondent aux critères des **produits laser de classe 1**, selon la norme **IEC 60825-1: 2014 (3ème édition)** et émettent dans le domaine de l'infrarouge, à une longueur d'onde de 865nm environ.

FDA 21CFR1040 Notice: All Ouster sensors comply with FDA performance standards for laser products except for deviations pursuant to Laser Notice No. 56, dated January 19, 2018.

Notice FDA 21CFR1040: Tous les capteurs Ouster sont conformes aux exigences de performances établies par la FDA pour les produits laser, à l'exception des écarts en application de l'avis nº56, daté du 19 janvier 2018.

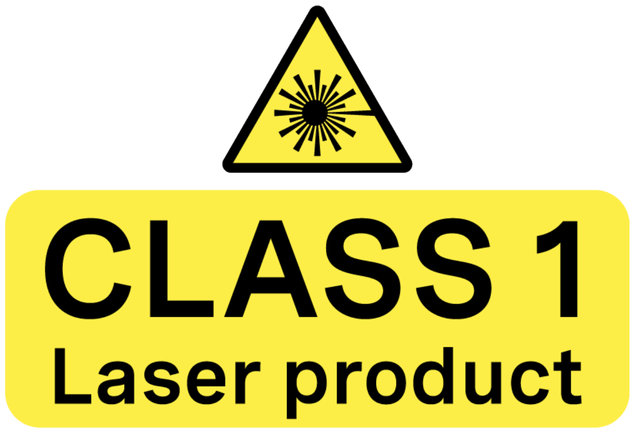


Figure 1.1: Class 1 Laser Product

Figure 1.2: Caution "Sharp Edges"

The following symbols appear on the product label and in the user manual have the following meaning.

CAUTIONS

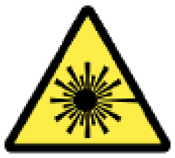


Figure 1.3: This symbol indicates that the sensor emits laser radiation.

Figure 1.4: This symbol indicates the presence of a hot surface that may cause skin burn.

- All Ouster sensors are hermetically sealed unit, and are non user-serviceable.
- Use of controls, or adjustments, or performance of procedures other than those specified herein, may result in hazardous radiation exposure.
- Use of any Ouster sensor is subject to the Terms of Sale that you agreed and signed with Ouster or your distributor/integrator. Included in these terms are the prohibitions of:
  - Removing or otherwise opening the sensor housing
  - Inspecting the internals of the sensor
  - Reverse-engineering any part of the sensor
  - Permitting any third party to do any of the foregoing
- Operating the sensor without the attached mount that is shipped with the sensor, or attaching the sensor to a surface of inappropriate thermal capacity runs the risk of having the sensor overheat under certain circumstances.
- The Ouster sensor features a modular cap design to enable more flexible mounting and integration solutions for the sensor.
- The modular cap design increases design flexibility but it does not remove the need for thermal management on top of the sensor. The attached radial cap serves an important thermal management purpose and the sensor will not operate properly without a cap.
- Operation for extended periods of time without the cap will result in system errors and the sensor overheating. The cap can be replaced with alternative solutions but it cannot be left off altogether.
- If you wish to operate the sensor with a custom mounting solution, please contact our Field Application Team and we can answer your questions and provide guidance for achieving proper operations.
- This product emits Class 1 invisible laser radiation. The entire window is considered to be the laser aperture. While Class 1 lasers are considered to be "eye safe", avoid prolonged direct view-

ing of the laser and do not use optical instruments to view the laser.

- When operated in an ambient temperature >40 °C, the metallic surfaces of the sensor may be hot enough to potentially cause skin burn. Avoid skin contact with the sensor's base, lid and the heatsink when the sensor is operated under these conditions. The sensor should not be used in an ambient temperature above 60°C. The maximum safety certified ambient operating temperature is 60°C.

PRECAUTIONS:

- Tous les capteurs Ouster sont des unité hermétiquement scellée, qui ne peut être entretenue ou modifiée par l'utilisateur.
- L'utilisation de commandes, de réglages, ou l'exécution de procédures autres que celles spécifiées dans le présent document peuvent entraîner des rayonnements laser dangereux.
- L'utilisation d'un capteur Ouster est soumise aux conditions de vente signées avec Ouster ou le distributeur/intégrateur, incluant l'interdiction de:
    - Retirer ou ouvrir de quelque façon le boîtier du capteur
    - Analyser les composants internes du capteur
    - Pratiquer la rétro-ingénierie de toute ou partie du capteur
    - Autoriser une tierce personne à mener les actions listées ci-dessus
- L'utilisation du capteur sans le support (fourni avec les capteur) ou sans contact avec une surface ayant des capacités thermiques adéquates peut entraîner une surchauffe du capteur dans certaines conditions.
- Ce capteur présente une conception avec un dissipateur thermique supérieur modulaire, ceci pour apporter plus de flexibilité de montage et d'intégration au capteur.
- Cette conception modulaire augmente la flexibilité de conception mais ne supprime pas le besoin de dissipation thermique au-dessus du capteur. Le dissipateur thermique radial fourni est essentiel à une bonne gestion thermique. Le capteur ne fonctionnera pas correctement sans cette pièce.
- Une utilisation prolongée du capteur sans le dissipateur thermique supérieur peut résulter à des erreurs système ainsi qu'à une surchauffe du capteur pouvant aller jusqu'à son extinction. Le dissipateur thermique fourni peut être remplacé par une autre solution de dissipation thermique adéquate, mais ne doit pas être simplement retiré.
- Si vous souhaitez utiliser votre capteur avec une dissipation thermique personnalisée, merci de contacter notre Équipe Support qui pourra répondre à vos questions et vous apporter le support et le conseil nécessaire.
- Ce produit émet un rayonnement laser invisible de classe 1. L'ouverture de sortie du laser est constituée par la fenêtre du capteur dans sa totalité. Même si les lasers de classe 1 ne sont pas considérés comme dangereux pour les yeux, ne regardez pas directement le rayonnement laser de façon prolongée et n'utilisez pas d'instruments optiques pour observer le rayonnement laser.
- Lors d'une utilisation à température ambiante supérieure à 40°C, la surface métallique du cap-

teur peut présenter des risques de brûlures pour la peau. Dans ces conditions, il est important d'éviter tout contact avec la partie supérieure, la base ou le dissipateur thermique du capteur. Le capteur ne doit pas être utilisé à une température ambiante supérieure à 60°C. 60°C est la température maximale certifiée d'opération sûre du capteur.

Equipment Label: Includes model and serial number and a notice that states the unit is a Class 1 Laser Product, is affixed to the underside of the Sensor Enclosure Base. It is only visible after the attached mount with which the Sensor is shipped, is removed. For location details please refer to the hardware user manual.

L'étiquette de l'équipement, comprenant le modèle, le numéro de série, et la classification du produit laser (ici, classe 1), est apposée au-dessous de la base du boîtier du capteur. Il n'est visible qu'après avoir retiré le diffuseur de chaleur avec lequel le capteur est expédié. L'emplacement est décrit précisément dans le manuel d'utilisation du matériel.

Electromagnetic Compatibility: The Ouster sensors are an FCC 47 CfR 15 Subpart B device. This device complies with part 15 of the FCC Rules. Operation is subject to the following conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

"Ouster" and Ouster sensors are both registered trademarks of Ouster, Inc. They may not be used without express permission from Ouster, Inc.

If you have any questions about the above points, contact us at legal@ouster.io.

1.2 Proper Assembly, Maintenance and Safe Use

All Ouster sensors can be easily set up by following the instructions outlined in the hardware user manual. Any mounting orientation is acceptable. Each sensor is shipped with an attached mount that can be used for test or normal use within the specified operating conditions. The sensor may also be affixed to any other user specific mount of appropriate thermal capacity. Please contact Ouster for assistance with approving the use of user specific mounting arrangements.

Any attempt to utilize the sensor outside the environmental parameters delineated in the relevant data sheet for your Ouster sensor may result in voiding the warranty.

When power is applied, the sensor powers up and commences boot-up with the laser disabled. The boot-up sequence is approximately 60s in duration, after which the internal sensor optics subassembly commences spinning, the laser is activated, and the unit operates in the default 1024 x 10 Hz mode. When the sensor is running, and the laser is operating, a faint red flickering light may be seen behind the optical window.

Note: All Ouster sensors utilize an 865nm infrared laser that is only dimly discernible to the naked eye. The sensor is fully Class 1 eye safe, though Ouster strongly recommends against peering into the optical window at close range while the sensor is operating. Ouster sensors are equipped with a multi-layer series of internal safety interlocks to ensure compliance to Class 1 Laser Eye Safe limits.

All Ouster sensors are hermetically sealed units, and are not user-serviceable. Any attempt to unseal the enclosure has the potential to expose the operator to hazardous laser radiation.

The sensor user interface may be used to configure the sensor to a number of combinations of scan rates and resolutions other than the default values of 1024 x 10 Hz resolution. In all available combinations, the unit has been evaluated by an NRTL to remain within the classification of a Class 1 Laser Device as per IEC 60825-1:2014 (Ed. 3).

1.2.1 Assemblage correct et utilisation sûre

Tous les capteurs Ouster s'installe facilement en fixant la base sur un support percé de trous concordants, et en suivant les instructions d'interconnexion décrites dans le manuel d'utilisation du matériel. Toute orientation de montage est acceptable. Chaque capteur est expédié équipé d'un dissipateur de chaleur, utilisable en phase de test et en conditions normales. Néanmoins tout autre support présentant une capacité thermique appropriée pour l'application de l'utilisateur peut être utilisé. Veuillez contacter Ouster dans le cas où un montage spécifique à votre application serait nécessaire.

Toute tentative d'utilisation du capteur en dehors des paramètres environnementaux définis dans la fiche technique de votre capteur Ouster peut entraîner l'annulation de la garantie.

Lorsque le capteur est sous tension, celui-ci démarre et commence son initialisation avec le laser désactivé. Le temps de démarrage est d'environ 60s, après quoi le sous-système optique entre en rotation et le laser est activé, le capteur opère alors dans son mode par défaut de 1024 x 10 Hz. Lorsque le capteur est en marche et que le laser est activé, on peut apercevoir une faible lumière rouge vacillante derrière la vitre teintée. Tous les capteurs Ouster utilisent une longueur d'ondes infra-rouge de 865nm à peine perceptible pour l'œil humain, et le rayonnement laser IR émis est sans danger pour les yeux. Cependant, bien que les rayonnements laser de classe 1 soient sans danger dans des conditions raisonnablement prévisibles, Ouster recommande fortement de ne pas regard er fixement la vitre teintée pendant que le capteur est en marche. Tous les capteurs Ouster sont des unités hermétiquement scellées, qui ne peuvent pas être entretenues, modifiées ou réparées par l'utilisateur. Toute tentative d'ouverture du boîtier a pour risque d'exposer l'opérateur à un rayon-nement laser dangereux.

Les capteurs Ouster sont équipés d'une série de dispositifs de sécurité à plusieurs niveaux, de façon à assurer en toutes circonstances le respect des limites d'irradiance correspondant aux rayonnements lasers de classe 1, sans danger pour les yeux.

L'interface utilisateur du logiciel du capteur peut être utilisée pour configurer le capteur selon un certain nombre de combinaisons de vitesses de balayage et de résolutions autres que les valeurs utilisées par défaut, respectivement de 1024 x 10 Hz.

2 Firmware Introduction

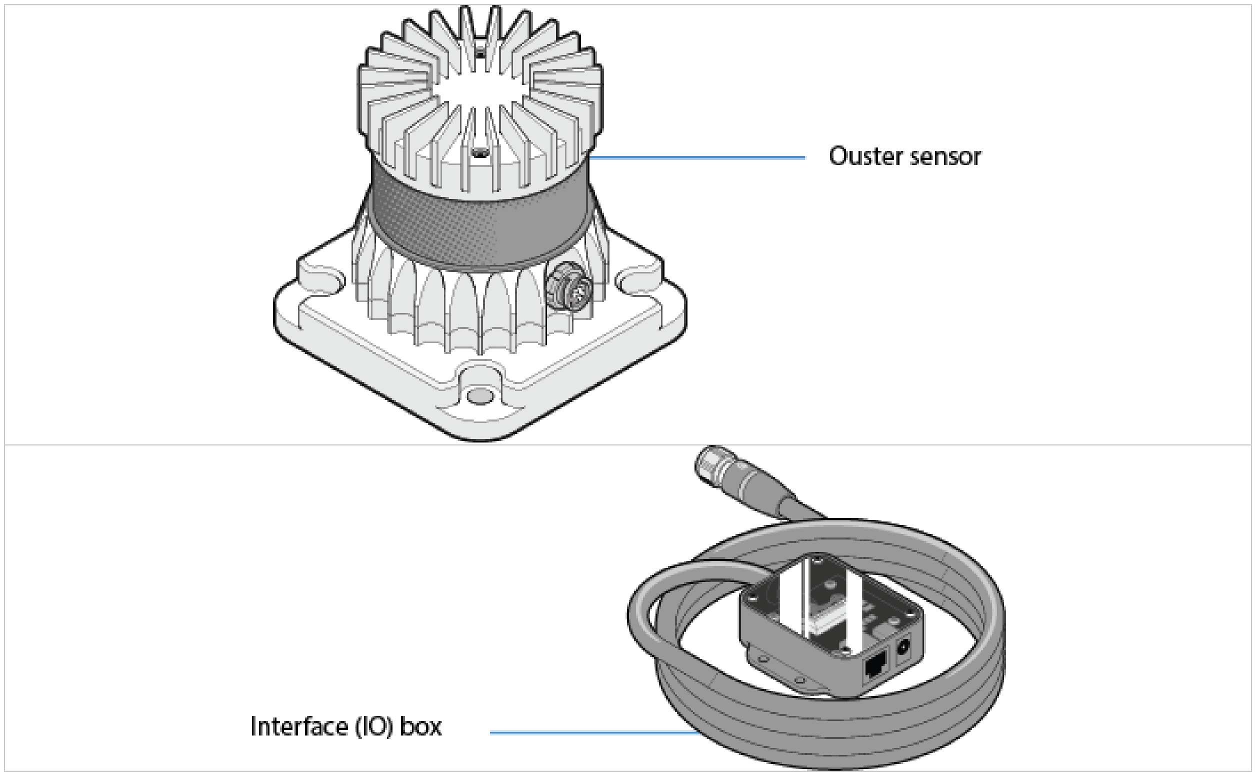
This **Firmware User Manual** is meant to allow the users to take advantage of all the features that are available with Ouster Sensors. Detailed Instructions regarding lidar operations, lidar data, API Guides and Troubleshooting guide are present in this user manual.

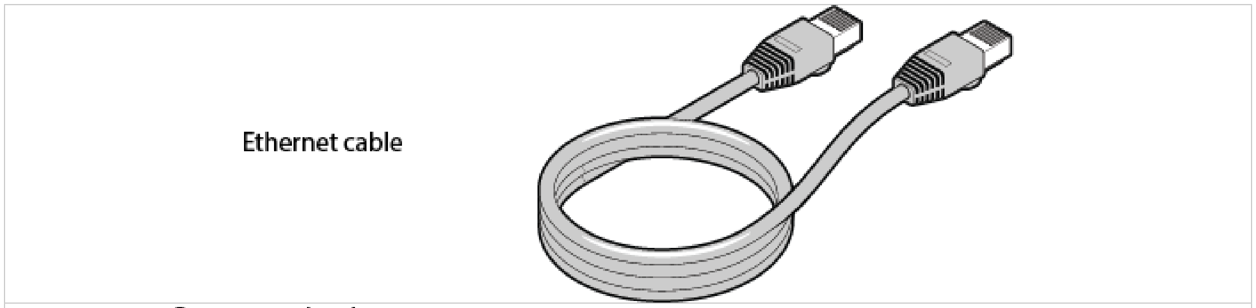
For information on the mechanical and electrical operations or the interface box, please refer to the Hardware User Manual.

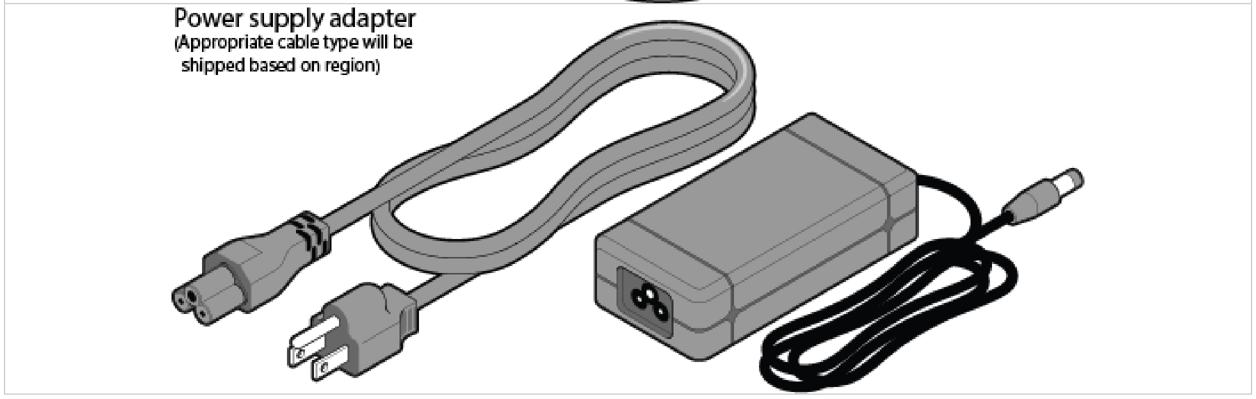
To know more about Ouster sensors and their specifications please refer to the datasheets available on our Website.

3 Connecting to Sensor

3.1 What's in the box



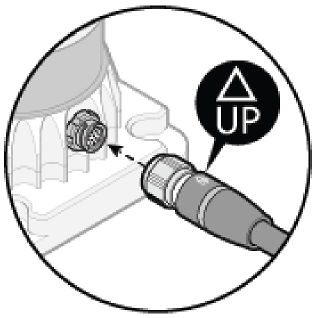




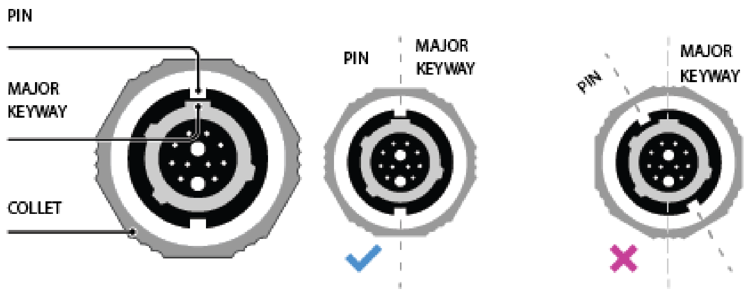
Note: Ibox is not always shipped with the sensor, based on customer requirement it could be a pig tail connector cable or a custom cable.

3.2 Sensor Setup

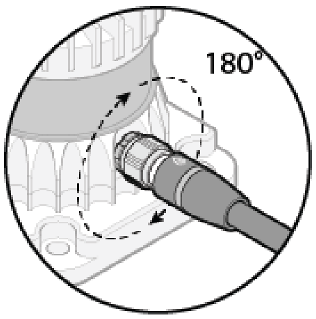
- Connect one end of the bayonet-style connector to the Ouster sensor as shown. Verify that the plug "UP" indicator is pointed up.



- Rotate the collet on the plug until one of its two pins is aligned with the major keyway. This will allow its two pins to enter the receptacle channel.



- Connect the plug to the sensor, then rotate the collet 180 degrees clockwise until it clicks. This indicates that it is fully seated.



- Connect one end of the power supply to the wall socket and the other end to the IO box.
- Connect one end of the ethernet cable provided to the IO box and the other end to a PC/LINUX/-MAC user interface.

3.3 Network Configuration

The sensor is designed to communicate with a host machine through a variety of different methods such a DHCP, IPv6/IPv4 link-local, and static IP.

On most systems you should be able to connect the sensor into your network or directly to a host machine and simply use the sensor hostname to communicate with it.

Your Ouster sensor requires a computer with a gigabit Ethernet connection and a 24V supply.

Optionally you may time synchronize the sensor through an external time source or through the computer via PTP.

The sensor hostname is, `os-991234567890.local`, where `991234567890` is the sensor serial number. The sensor serial number can be found on a sticker affixed to the top of the sensor.

For more detailed guidance on communicating with the sensor on various operating systems and network settings please reference the *Networking Guide* in the Appendix.

Commands for *setting* and *deleting* a static IP address can be found in the *HTTP API Reference Guide* section.

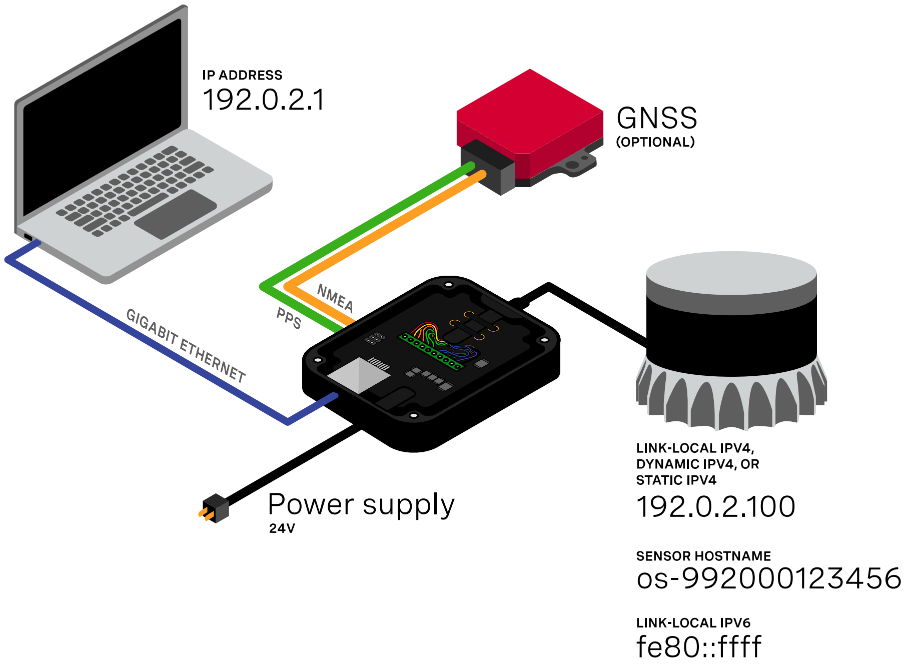


Figure 3.1: Network Configuration and Setup

Note: May be required to deactivate the firewall to connect with the sensor and access sensor data.

Open Google Chrome/Microsoft Edge/Firefox. Use the hostname in the format of `http://OS-99xxxxxxxxxx.local` and click on "Enter/Character turn" to open Ouster Dashboard.

Note:

- The serial number of the sensor need not start with `99` and is only taken as an example in this document, the sensor serial number can be found on a sticker affixed to the top of the sensor.
- Please keep in mind **NOT** to use `https://` as it will result in an error, use without `s` as shown `http://OS-99xxxxxxxxxx.local`.

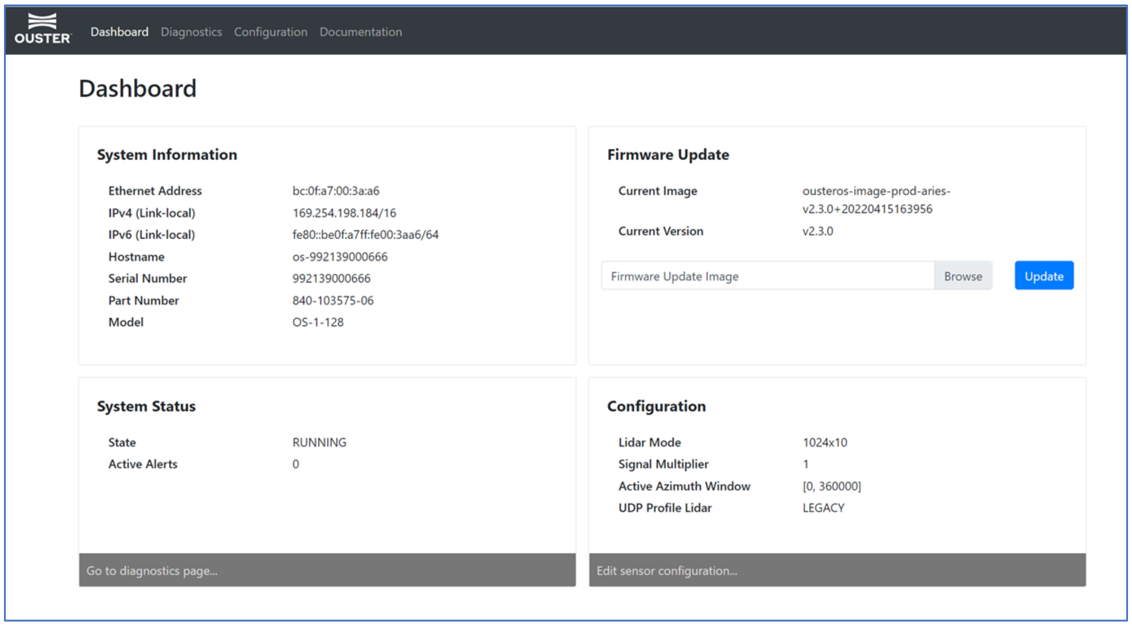
3.4 Web Interface

The sensor homepage can be accessed by typing in the sensor's address (IPv4, IPv6, or hostname) in a web browser (http://os-991234567890.local/ where 991234567890 is the serial number). From here you can see information about the sensor, access documentation, and configure sensor settings.

Note: This new version of the web UI will only be accessible after updating to FW 2.2.0 or later.

Dashboard: Contains an overview of the sensor.

- **System Information**: This panel provides information regarding the network configuration and hardware details that are unique to each sensor
- **Firmware Update**: You can update firmware on this panel. See *Updating Firmware* for more details.
- **System Status**: This panel displays the status of the sensor and information regarding any Active Alerts. More information on the status of the sensor can be found by clicking the link, which will take the user to the Diagnostics tab
- **Configuration**: An overview of the sensor configuration is available on this panel. The sensor configuration can also be edited by clicking on the link below, which will take the user to the Configuration tab



Dashboard

System Information

Ethernet Address	bc:0f:a7:00:3a:a6
IPv4 (Link-local)	169.254.198.184/16
IPv6 (Link-local)	fe80::be0f:a7ff:fe00:3aa6/64
Hostname	os-992139000666
Serial Number	992139000666
Part Number	840-103575-06
Model	OS-1-128

Firmware Update

Current Image	ousteros-image-prod-aries-v2.3.0+20220415163956
Current Version	v2.3.0

System Status

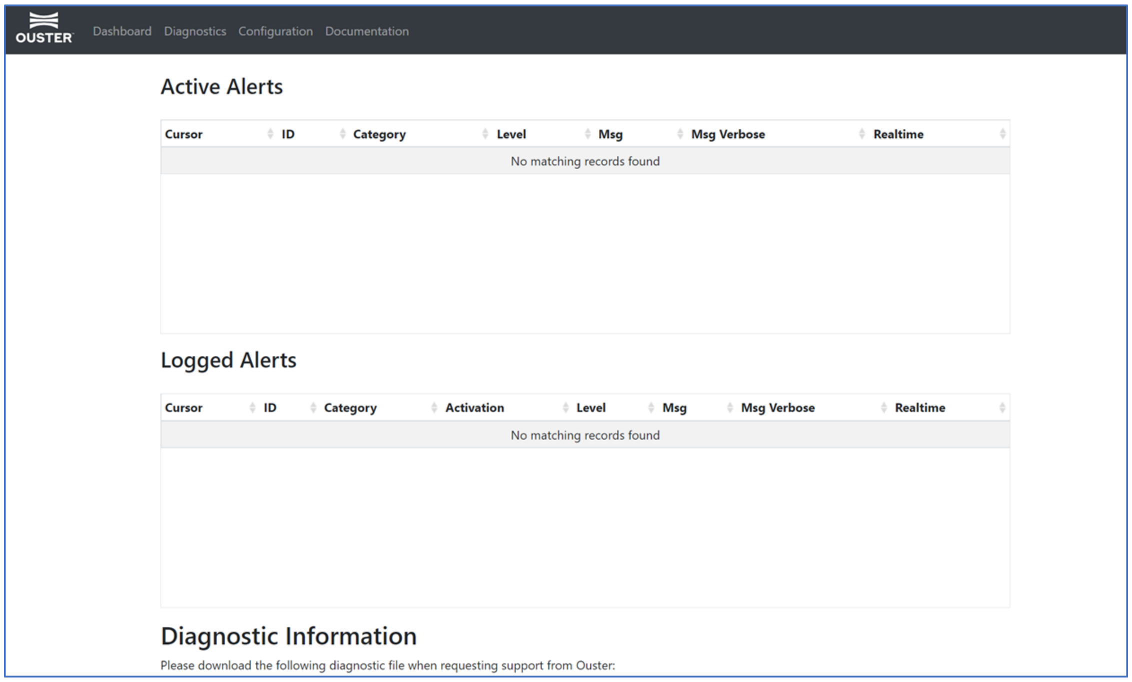
State	RUNNING
Active Alerts	0

Configuration

Lidar Mode	1024x10
Signal Multiplier	1
Active Azimuth Window	[0, 360000]
UDP Profile Lidar	LEGACY

Figure 3.2: Ouster Dashboard

Diagnostics: Contains diagnostic alert and error information about the sensor for troubleshooting purposes. For a list of possible alerts and errors, see *Alerts and Errors*. Some Alerts require the user to reach out to ouster support. Please include a copy of the System Diagnostics file which can be downloaded by clicking the blue tab on this page.



Active Alerts

Cursor	ID	Category	Level	Msg	Msg Verbose	Realtime
No matching records found						

Logged Alerts

Cursor	ID	Category	Activation	Level	Msg	Msg Verbose	Realtime
No matching records found							

Diagnostic Information

Please download the following diagnostic file when requesting support from Ouster:

Figure 3.3: Ouster Alerts & Diagnostics

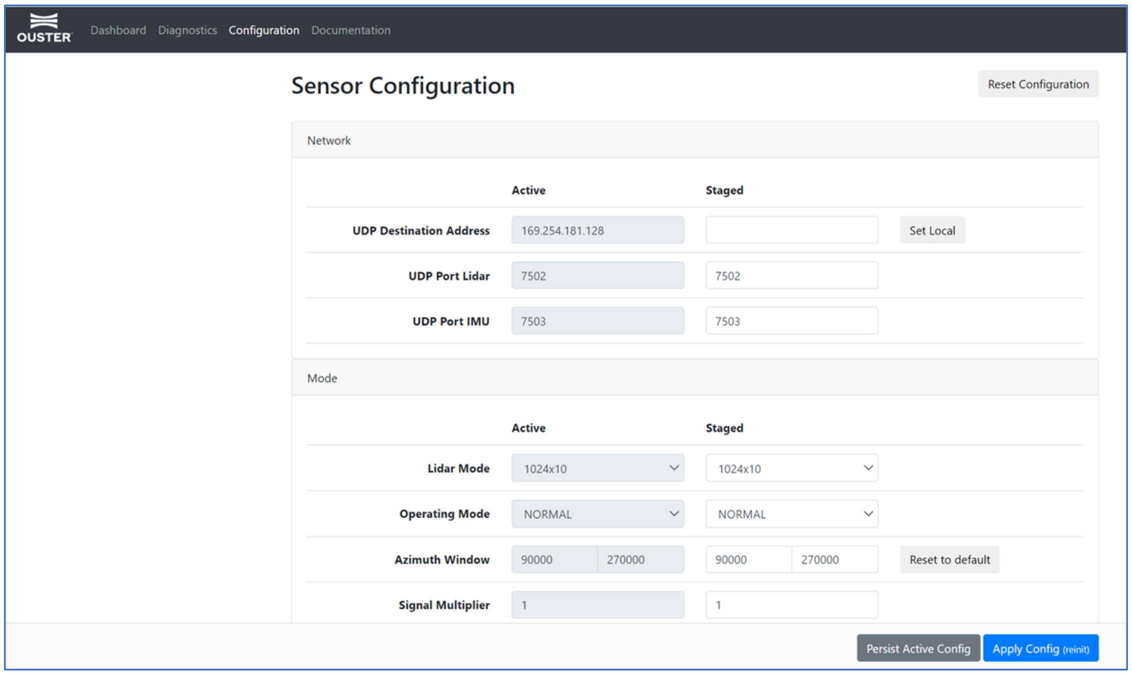
Configuration: This tab contains a user interface to change sensor configuration. While in STANDBY mode, we can update the configuration settings in the WEB UI, but it will not take effect until we switch the sensor back to NORMAL mode

- **Reset Configuration**: Resets sensor to factory configurations and settings. Note that this resets any static IP address given to the sensor.
- **Persist Active Config**: Allows the user to configure the sensor settings and set them as the active configuration without reinitializing the sensor
- **Apply Config (reinit)**: Allows the user to configure the sensor settings. This involves a reinitialization of the sensor, so that the sensor configuration settings can take effect
- **Documentation**: Contains the HTTP and TCP API guides that are compatible with the version of the firmware on the sensor. Visit Ouster Sensor Documentation for latest hardware and software user manuals, along with integration guides and troubleshooting guides.

4 Updating Firmware

Sensor firmware can be updated with an Ouster-provided firmware file from Ouster FW (or directly from the deployment engineering team) by accessing the sensor over http - e.g., http://os-991900123456.local/ and uploading the file as prompted.

Always check your firmware version before attempting an update. Only update to an equal or higher version number.



Sensor Configuration

Network

	Active	Staged
UDP Destination Address	169.254.181.128	
UDP Port Lidar	7502	7502
UDP Port IMU	7503	7503

Mode

	Active	Staged
Lidar Mode	1024x10	1024x10
Operating Mode	NORMAL	NORMAL
Azimuth Window	90000 270000	90000 270000
Signal Multiplier	1	1

Figure 3.4: Sensor Configuration

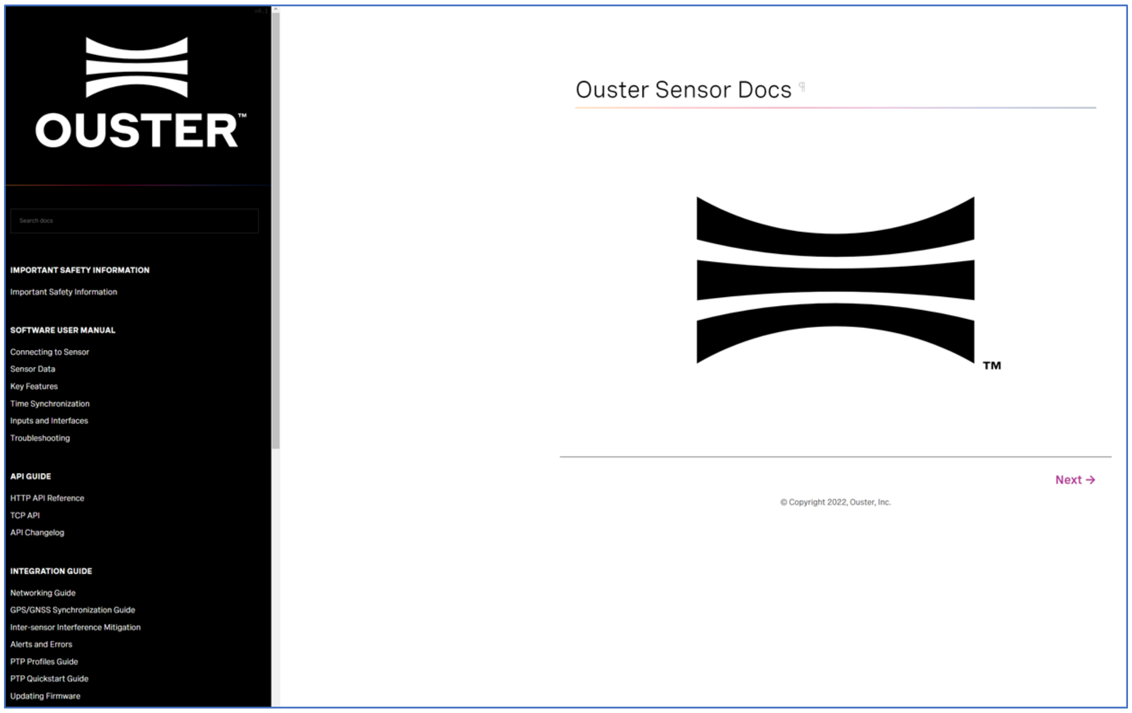
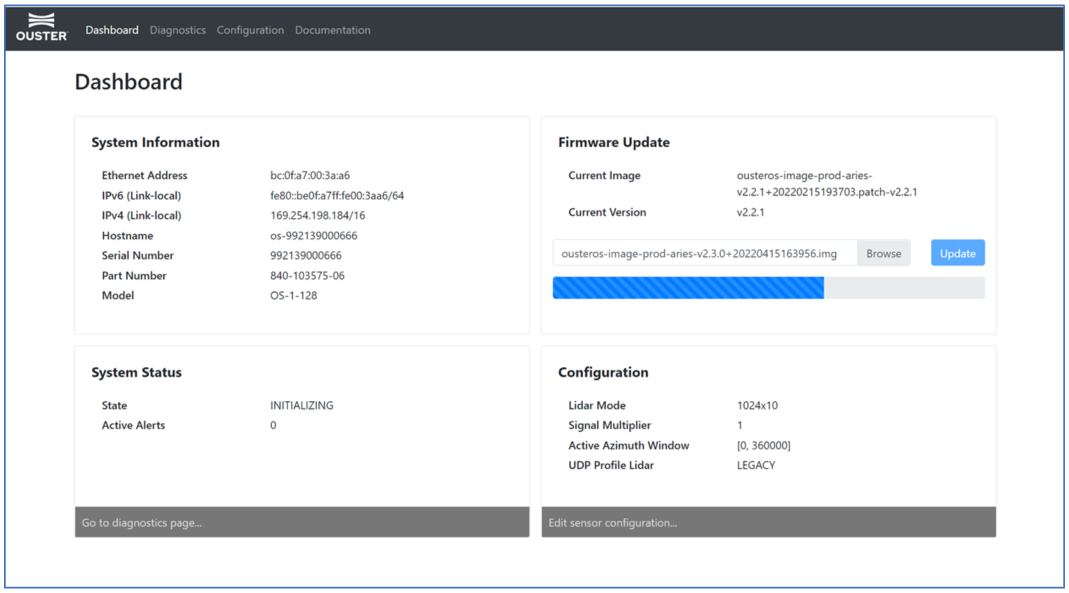


Figure 3.5: Ouster Documentation



Dashboard

System Information

Ethernet Address	bc:0f:a7:00:3a:a6
IPv6 (Link-local)	fe80::be0f:a7ff:fe00:3aa6/64
IPv4 (Link-local)	169.254.198.184/16
Hostname	os-992139000666
Serial Number	992139000666
Part Number	840-103575-06
Model	OS-1-128

Firmware Update

Current Image	ousteros-image-prod-aries-v2.2.1+20220215193703.patch-v2.2.1
Current Version	v2.2.1

ousteros-image-prod-aries-v2.3.0+20220415163956.img

System Status

State	INITIALIZING
Active Alerts	0

Configuration

Lidar Mode	1024x10
Signal Multiplier	1
Active Azimuth Window	[0, 360000]
UDP Profile Lidar	LEGACY

Figure 4.1: Uploading a new firmware image onto the sensor

After the web UI confirms that the update is complete, please allow the sensor to reboot (about 2 minutes) and refresh your webpage to get access to the updated Web UI.

4.1 Software Resources

After connecting to your sensor, you can quickly visualize the point cloud using Ouster Python SDK or using Ouster Studio . Both Ouster Python SDK and Ouster Studio are available for Linux, Mac, and Windows. Please visit Ouster Resources for the latest tools to visualize your sensor output.

Ouster Python SDK Ouster Python SDK provides a high-level interface for interacting with sensor hardware and record sensor data suitable for prototyping, evaluation, and other non-safety critical applications. Example and reference code is provided for several common operations on sensor data. he SDK includes APIs for:

- Querying and setting sensor configuration
- Recording and reading data in pcap format
- Reading and buffering sensor UDP data streams reliably
- Frame-based access to lidar data as numpy datatypes
- Conversion of raw data to range/signal/near_ir/reflectivity images (de-staggering)
- Efficient projection of range measurements to Cartesian (X, Y, Z) coordinates

Ouster Studio Ouster Studio is software provided by Ouster to visualize, record, and analyze data from Ouster lidar sensors. Ouster Studio is cross-platform, with official support for Windows, MacOS and Ubuntu. The software performs real-time visualization, processing, and recording of live 3D lidar data captured from Ouster lidar sensors.

Note: Ouster recommends users to utilize Ouster Python SDK.

5 Sensor Data

5.1 Coordinate Frames and XYZ Calculation

Ouster defines two coordinate frames:

The **Lidar Coordinate Frame** follows the Right Hand Rule convention and is a point cloud-centric frame of reference that is the simplest frame in which to calculate and manipulate point clouds. The X-axis points backwards towards the external connector, which is an unintuitive orientation that was deliberately chosen to meet the following criteria:

- data frames split at the back of the sensor i.e. the external connector
- data frames start with the azimuth angle equal to 0°

All point cloud features including setting an azimuth window and phase locking are defined in the Lidar Coordinate Frame.

The **Sensor Coordinate Frame** follows the Right Hand Rule convention and is a mechanical housing-centric frame of reference that follows robotics convention with X-axis pointing forward. Ouster-provided drivers and visualizers represent data in the Sensor Coordinate Frame.

Note: All Ouster coordinate frames follow the Right Hand Rule, allowing for standard 3D transformation matrix math to convert between them. For multi-sensor systems that require calibration, this Linear Algebra-based approach can be convenient. However, customers with single-sensor systems may find it more intuitive to stay in the Lidar Coordinate Frame and take arithmetic shortcuts.

5.1.1 Lidar Coordinate Frame

The Lidar Coordinate Frame is defined at the intersection of the lidar axis of rotation and the lidar optical midplane (a plane parallel to Sensor Coordinate Frame XY plane and coincident with the 0° elevation beam angle of the sensor).

The Lidar Coordinate Frame axes are arranged with:

- positive X-axis pointed at encoder angle 0° and the external connector
- positive Y-axis pointed towards encoder angle 90°
- positive Z-axis pointed towards the top of the sensor

The Lidar Coordinate Frame is marked in both diagrams below with X_L, Y_L, and Z_L.

5.1.2 Lidar Range to XYZ

Given the following information, range data may be transformed into 3D cartesian XYZ coordinates in the Lidar Coordinate Frame:

From a measurement block from the UDP packet:

- `Measurement ID` value can be found on the lidar data packet
- `scan_width` value of the horizontal resolution
- `r` or `range_mm`[1] value of the data block of the *i*-th channel
- `r'` or `range_to_beam_origin_mm`[2]

From the `get_beam_intrinsics` TCP command:

- `lidar_origin_to_beam_origin_mm`[3] value
- `beam_altitude_angles` array
- `beam_azimuth_angles` array

The corresponding 3D point can be computed by

$$
\begin{aligned}
r &= range\_mm \\
|\vec{n}| &= lidar\_origin\_to\_beam\_origin\_mm \\
r &= |\vec{r'}| + |\vec{n}| \\
\theta_{encoder} &= 2\pi \cdot \left(1 - \frac{measurement\ ID}{scan\_width}\right) \\
\theta_{azimuth} &= -2\pi \frac{beam\_azimuth\_angles[i]}{360} \\
\phi &= 2\pi \frac{beam\_altitude\_angles[i]}{360}
\end{aligned}
$$

$$
\begin{aligned}
x &= (r - |\vec{n}|) \cos(\theta_{encoder} + \theta_{azimuth}) \cos(\phi) + |\vec{n}| \cos(\theta_{encoder}) \\
y &= (r - |\vec{n}|) \sin(\theta_{encoder} + \theta_{azimuth}) \cos(\phi) + |\vec{n}| \sin(\theta_{encoder}) \\
z &= (r - |\vec{n}|) \sin(\phi)
\end{aligned}
$$

[1] `r` or `range_mm` is the sum of the magnitudes of vectors of r′ and n. This value is provided for each measurement in blocks [0-15] of the *i*-th channel.

[2] `r'` or `range_to_beam_origin_mm` is the magnitude of the distance vector from lidar front optics to the detected object. This value is **NOT** provided; It is only to help illustrate the concept.

[3] `n` or `lidar_origin_to_beam_origin_mm` is the magnitude of the distance vector from the center of the lidar origin coordinate frame to lidar front optics. This value is provided from the `get_beam_intrinsics`, please refer to the API Guide for more information.

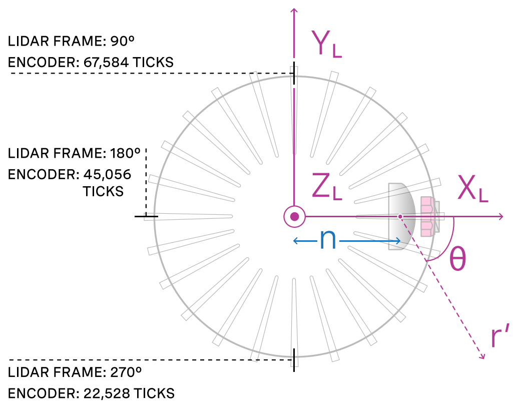


Figure 5.1: Top-down view of Lidar Coordinate Frame

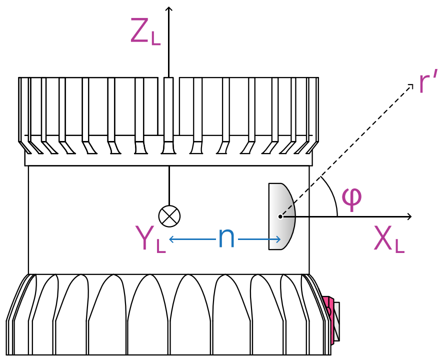


Figure 5.2: Side view of Lidar Coordinate Frame

5.1.3 Sensor Coordinate Frame

The Sensor Coordinate Frame is defined at the center of the sensor housing on the bottom, with the X-axis pointed forward, Y-axis pointed to the left and Z-axis pointed towards the top of the sensor. The external connector is located in the negative x direction. The Sensor Coordinate Frame is marked in the diagram below with X_S, Y_S, Z_S.

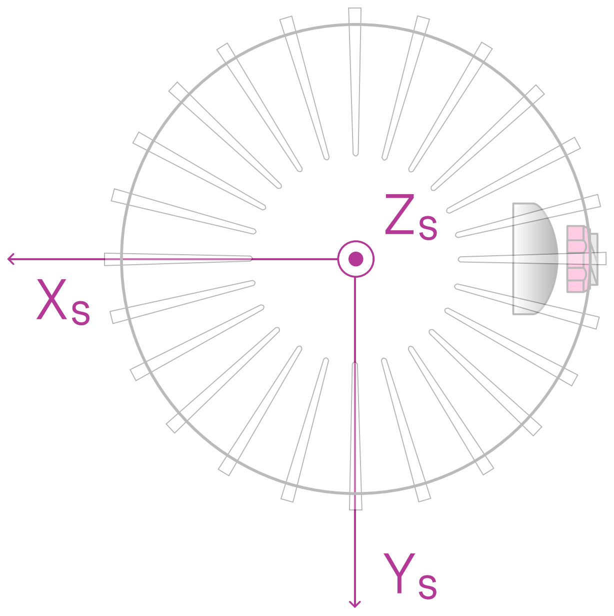


Figure 5.3: Top-down view of Sensor Coordinate Frame

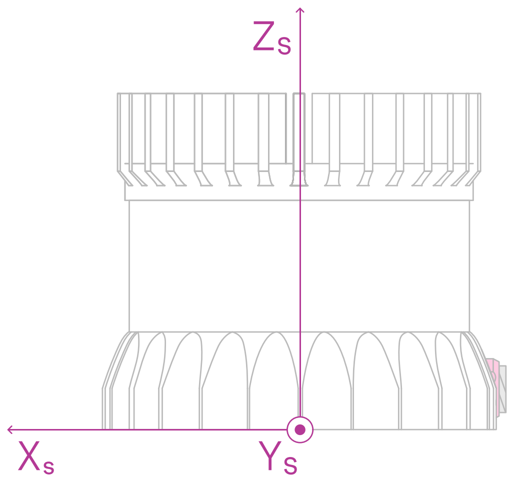


Figure 5.4: Side view of Sensor Coordinate Frame

5.1.4 Combining Lidar and Sensor Coordinate Frame

The Lidar Coordinate Frame's positive X-axis (0 encoder value) is opposite the Sensor Coordinate Frame's positive X-axis to center lidar data about the Sensor Coordinate Frame's positive X-axis. A single measurement frame starts at the Lidar Coordinate Frame's 0º position and ends at the 360º position. This is convenient when viewing a "range image" of the Ouster Sensor measurements, allowing the "range image" to be centered in the Sensor Coordinate Frame's positive X-axis, which is generally forward facing in most robotic systems.

The Ouster Sensor scans in the clockwise direction when viewed from the top, which is a negative rotational velocity about the Z-axis. Thus, as encoder ticks increase from 0 to 90,111, the actual angle about the Z-axis in the Lidar Coordinate Frame will decrease.

5.1.5 Lidar Intrinsic Beam Angles

The intrinsic beam angles for each beam may be queried with a TCP command `get_beam_intrinsics` to provide an azimuth and elevation adjustment offset to each beam. The azimuth adjustment is referenced off of the current encoder angle and the elevation adjustment is referenced from the XY plane in the Sensor and Lidar Coordinate Frames.

5.1.6 Lidar Range Data To Sensor XYZ Coordinate Frame

For applications that require calibration against a precision mount or use the IMU data in combination with the lidar data, the XYZ points should be adjusted to the Sensor Coordinate Frame. This requires a Z translation and a rotation of the X,Y,Z points about the Z-axis. The z translation is the height of the lidar aperture stop above the sensor origin, which varies depending on the sensor you have, and the data must be rotated 180º around the Z-axis. This information can be queried over TCP in the form of a homogeneous transformation matrix in row-major ordering.

Example JSON formatted query using the TCP command `get_lidar_intrinsics`:

```
{
  "lidar_to_sensor_transform": [-1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 1, 36.180, 0, 0, 0, 1]
}
```

Which corresponds to the following matrix

$$M\_lidar\_to\_sensor = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 36.180 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The table below lists all product lines' distances of the aperture stop above the sensor origin for use in the z translation.

Table 5.1: Lidar aperture stop above sensor origin

Product Line	Lidar aperture stop above sensor origin
OS0 Gen 1 & Gen 2 (All Revisions)	36.180 mm
OS1 Gen 1 & Gen 2 (All Revisions)	36.180 mm
OS2 Gen 2 (Revisions A, B, C)	74.296 mm
OS2 Gen 2 (Revisions D and higher)	78.296 mm

5.1.7 IMU Data To Sensor XYZ Coordinate Frame

The IMU is slightly offset in the Sensor Coordinate Frame for practical reasons. The IMU origin in the Sensor Coordinate Frame can be queried over TCP in the form of an homogeneous transformation matrix in row-major ordering.

Example 1- Expected response for TCP command `get_imu_intrinsics` when using Gen1 OS1 (all revisions), Gen2 OS01 (all revisions) and Gen2 OS2 (top-level revisions A, B, C)

```
{
  "imu_to_sensor_transform": [1, 0, 0, 6.253, 0, 1, 0, -11.775, 0, 0, 1, 7.645, 0, 0, 0, 1]
}
```

Which corresponds to the following matrix

$$M\_imu\_to\_sensor = \begin{bmatrix} 1 & 0 & 0 & 6.253 \\ 0 & 1 & 0 & -11.775 \\ 0 & 0 & 1 & 7.645 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example 2- Expected response for TCP command `get_imu_intrinsics` when using Gen2 OS2 (top-level revisions D and higher)

```
{
  "imu_to_sensor_transform": [1, 0, 0, 6.253, 0, 1, 0, -11.775, 0, 0, 1, 11.645, 0, 0, 0, 1]
}
```

Which corresponds to the following matrix

$$M\_imu\_to\_sensor = \begin{bmatrix} 1 & 0 & 0 & 6.253 \\ 0 & 1 & 0 & -11.775 \\ 0 & 0 & 1 & 11.645 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6 Lidar Data Packet Format

With firmware version 2.3 and above, users will have the option to switch between different lidar data packet formats as shown below.

- *Configurable Data Packet Format*
  - *Single Return Profile*
  - *Low Data Rate Profile*
  - *Dual Return Profile*
- *LEGACY Data Packet Format*

By default, the data packet format will be set to `LEGACY`.

Note: In order to enable dual returns the user needs to have both a Rev 06 sensor or later and an upgrade to firmware version 2.2 or later.

6.1 Configurable Data Packet Format

When setting the `udp_profile_lidar` to a value other than `LEGACY`, a new packet format will be automatically activated. This packet format, called the configurable data packet format, allows the users to choose between different options for the channel data profile depending on their usage, while maintaining a uniform packet structure across different profiles. It is described in detail below.

6.1.1 Lidar Data Format

When the config parameter `udp_profile_lidar` is set to a parameter other than `LEGACY`, we will be in the Configurable data packet format. Each data packet consists of Packet Header, Measurement Header, Channel Data Blocks and Packet Footer. The packet rate is dependent on the lidar mode. Words are 32 bits in length and little endian. By default, lidar UDP data is forwarded to Port `7502`. Please refer to the **TCP API** portion of this manual for more information on setting this parameter. Alternately, this mode can also be configured via the Web Interface.

Packet layout

Packet Header [256 bits]

- **Packet type** [16 bit unsigned int] - Identifies lidar data vs. other packets in stream. Packet Type is 0x1 for Lidar packets.
- **Frame ID** [16 bit unsigned int] - Index of the lidar scan, increments every time the sensor completes a rotation, crossing the zero azimuth angle.

- **Init ID** [24 bit unsigned int] - Initialization ID. Updates on every reinit, which may be triggered by the user or an error, and every reboot. This value may also be obtained by running the TCP command `get_sensor_info`.
- **Serial No** [40 bit unsigned int] - Serial number of the sensor. This value is unique to each sensor and can be found on a sticker affixed to the top of the sensor. In addition, this information is also available on the Sensor Web UI and by reading the field `prod_sn` from `get_sensor_info`.

Column Header Block [96 bits]

- **Timestamp** [64 bit unsigned int] - Timestamp of the measurement in nanoseconds.
- **Measurement ID** [16 bit unsigned int] - Sequentially incrementing measurement counting up from 0 to 511, or 0 to 1023, or 0 to 2047 depending on lidar_mode.
- **Status** [1 bit unsigned int] - Indicates validity of the measurements. Status is 0x01 for valid measurements.Status is 0x00 for dropped or disabled columns.

Channel Data Blocks [Varies based on channel data profile]

- The size and the structure of the channel data block varies based on the configurable data packet format chosen by the user. More information on each of these options are described below in the following sections.

Packet Footer [256 bits]

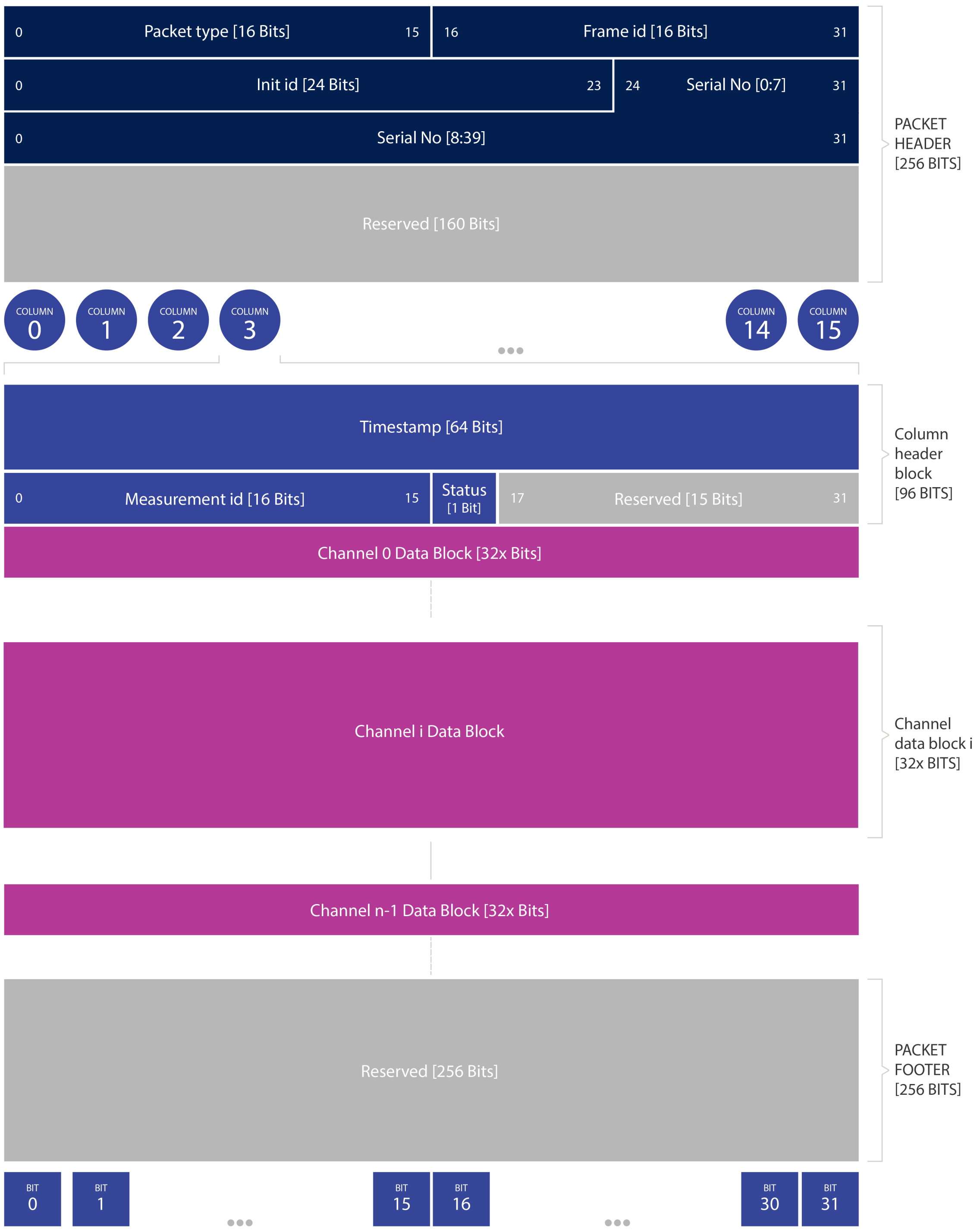


Figure 6.1: Configurable Data Packet Configuration

6.1.2 Channel Data Profiles

This section describes the different channel data profile options that are available to the users as part of the configurable data packet format. Each of these data profiles can be selected by setting the configuration parameter udp_profile_lidar to one of the following options:

- RNG19_RFL8_SIG16_NIR16 (Single Return Profile)
- RNG15_RFL8_NIR8 (Low Data Rate Profile)
- RNG19_RFL8_SIG16_NIR16_DUAL (Dual Return Profile)

More details on how to set the configuration parameters are described in the *TCP API Guide* portion of this Firmware User Manual.

Note: Calibrated reflectivity has certain hardware requirements. Please refer to the *Calibrated Reflectivity* section for more details.

Table 6.1: Configurable Data Packet Profiles

Description	*Single Return Profile*	*Low Data Rate Profile*	*Dual Return Profile*
Profiles	`RNG19_RFL8_SIG16_NIR16`	`RNG15_RFL8_NIR8`	`RNG19_RFL8_SIG16_NIR16_DUAL`
Words per pixel	3	1	4
Range RET1	19 bits	15 bits	19 bits
Reflectivity RET1	8 bits	8 bits	8 bits
Range RET2	Not Available	Not Available	19 bits
Reflectivity RET2	Not Available	Not Available	8 bits
Signal RET1	16 bits	Not Available	16 bits
Signal RET2	Not Available	Not Available	16 bits
NIR	16 bits	8 bits	16 bits

6.1.3 Single Return Profile

This channel data profile can be activated by setting the configuration parameter udp_profile_lidar to RNG19_RFL8_SIG16_NIR16.

Note: This is only available with firmware version 2.3 or later.

```
set_config_param udp_profile_lidar RNG19_RFL8_SIG16_NIR16
```

The above command will set the channel data profile in the configurable data packet format to Single Return mode. This channel data profile is identical to the channel data block present in LEGACY format, but makes use of the configurable data packet format. Users looking to take advantage of the configurable data packet format can use this profile in place of LEGACY. The channel data profile for this is described below.

Channel Data Blocks [96 bits each for RNG19_RFL8_SIG16_NIR16 profile]

For RNG19_RFL8_SIG16_NIR16 profile the channel data block consists of 3 words to accommodate data for porting over the LEGACY profile to configurable Data Packet format. Only a single return will be made available to the user.

- **Range** [19 bit unsigned int] - Range in millimeters, discretized to the nearest 1 millimeters with a maximum range of 524m. Note that range value will be set to 0 if out of range or if no detection is made.
- **Calibrated Reflectivity** [8 bit unsigned int] - Sensor Signal Photons measurements are scaled based on measured range and sensor sensitivity at that range, providing an indication of target reflectivity.
- **Signal Photons** [16 bit unsigned int] - Signal intensity photons in the signal return measurement are reported.
- **Near Infrared Photons** [16 bit unsigned int] - NIR photons related to natural environmental illumination are reported.

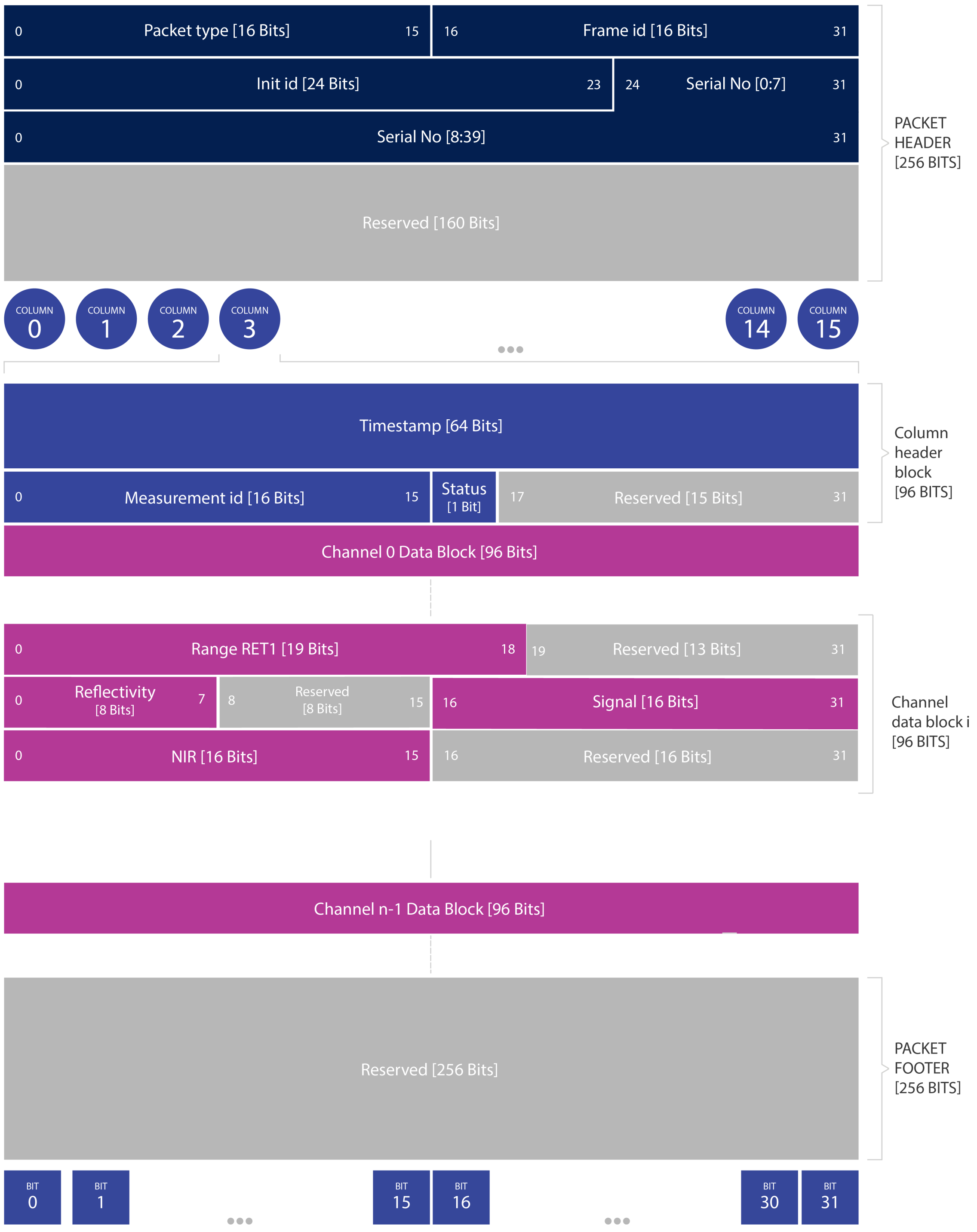


Figure 6.2: Single Return Configuration

6.1.4 Low Data Rate Profile

This channel data profile can be activated by setting the configuration parameter udp_profile_lidar to RNG15_RFL8_NIR8.

Note: This is only available with firmware version 2.3 or later.

```
set_config_param udp_profile_lidar RNG15_RFL8_NIR8
```

The above command will set the channel data profile in the configurable data packet format to Low Data Rate configuration.

This channel data profile is especially useful to users who are looking to adopt a channel data profile to fit with a limited compute capabilities. The data rate and the data packet size that are achieved as a result of using this profile will be smaller as compared to the other channel data profile options that are available.

The channel data profile for this is described below.

Channel Data Blocks [32 bits each for RNG15_RFL8_NIR8 profile]

For the RNG15_RFL8_NIR8 profile the channel data block consists of only 1 word to accommodate data for optimizing information at a low data rate. Only a single return will be made available to the user.

- **Range** [15 bit unsigned int] - Range scaled down by a factor of 8 mm, for a maximum range of (2^15*8) = 262 mm in 15 bits. **Note** The range value will be set to 0 if out of range or if no detection is made.
- **Calibrated Reflectivity** [8 bit unsigned int] - Sensor Signal Photons measurements are scaled based on measured range and sensor sensitivity at that range, providing an indication of target reflectivity.
- **Near Infrared Photons** [8 bit unsigned int] - NIR photons related to natural environmental illumination are reported. Measurements are taken similar to LEGACY and other data profiles (Single Data Profile and Dual Return Profile) but it is scaled down by a factor of 16.

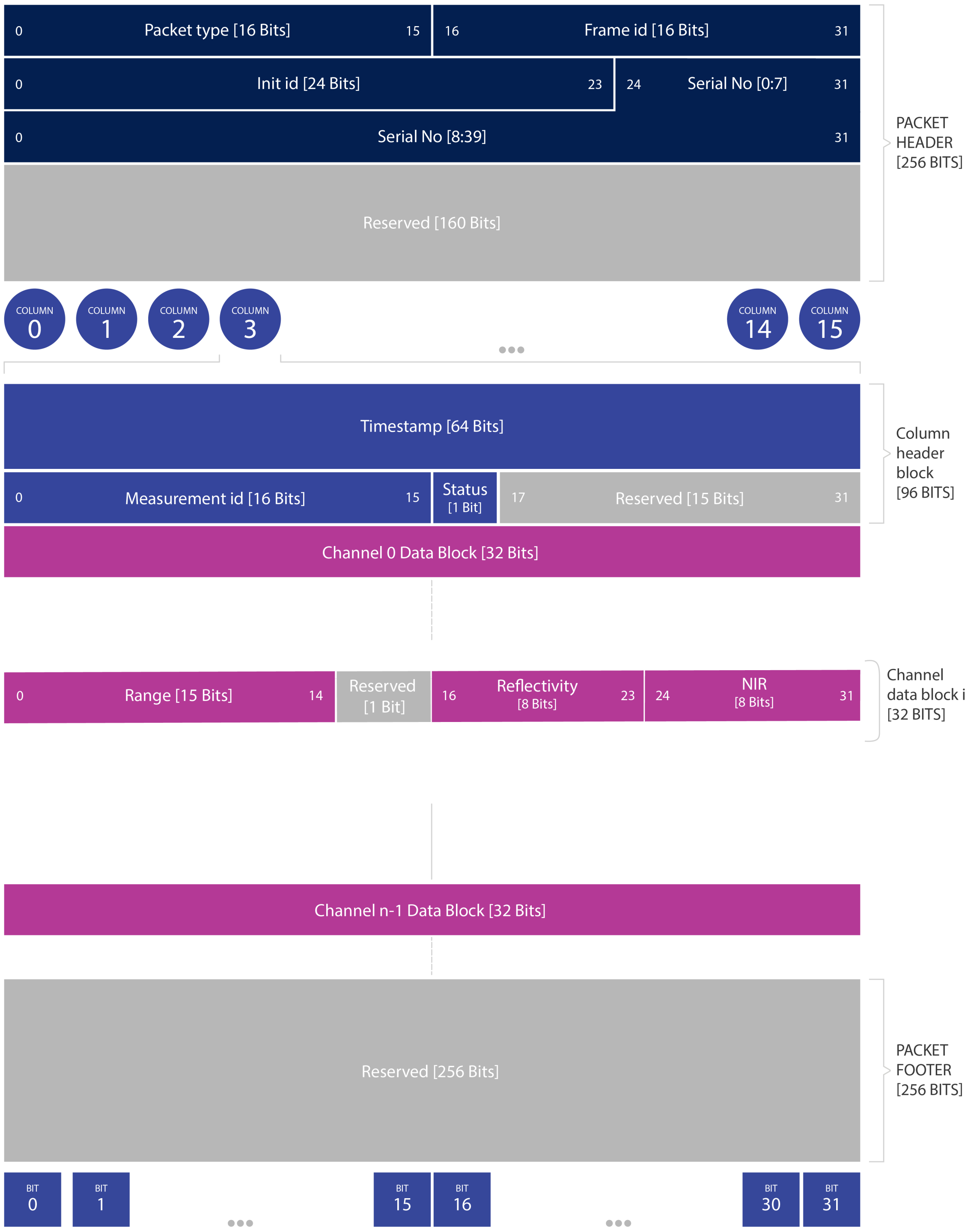


Figure 6.3: Low Data Rate Configuration

6.1.5 Dual Return Profile

This channel data profile can be activated by setting the configuration parameter udp_profile_lidar to RNG19_RFL8_SIG16_NIR16_DUAL. Please note in order to enable dual returns the user needs to have both a Rev 06 sensor or later and an upgrade to firmware version 2.2 or later.

```
set_config_param udp_profile_lidar RNG19_RFL8_SIG16_NIR16_DUAL
```

This feature is meant to allow the sensor to provide an output up to 2 returns (Strongest and Second Strongest).

This feature will allow Ouster sensors to operate in scenarios with semi-transparent obscurants, such as rain, fog, or even a chain-link fence. In these scenarios, the strongest and second strongest returns are required to see both the semi-transparent object, as well as whatever may lie behind it.

Channel Data Blocks [128 bits each for RNG19_RFL8_SIG16_NIR16_DUAL profile]

For RNG19_RFL8_SIG16_NIR16_DUAL profile the channel data block consists of 4 words to accommodate data for the multiple returns. A total of up to two returns will be made available to the user.

- **Range RET1/2** [19 bit unsigned int] - range in millimeters, discretized to the nearest 1 millimeters with a maximum range of 524m. Note that range value will be set to 0 if out of range or if no detection is made.
- **Calibrated Reflectivity RET1/2** [8 bit unsigned int] - Sensor Signal Photons measurements are scaled based on measured range and sensor sensitivity at that range, providing an indication of target reflectivity.
- **Signal Photons RET1/2** [16 bit unsigned int] - Signal intensity photons in the signal return measurement are reported.
- **Near Infrared Photons** [16 bit unsigned int] - NIR photons related to natural environmental illumination are reported.

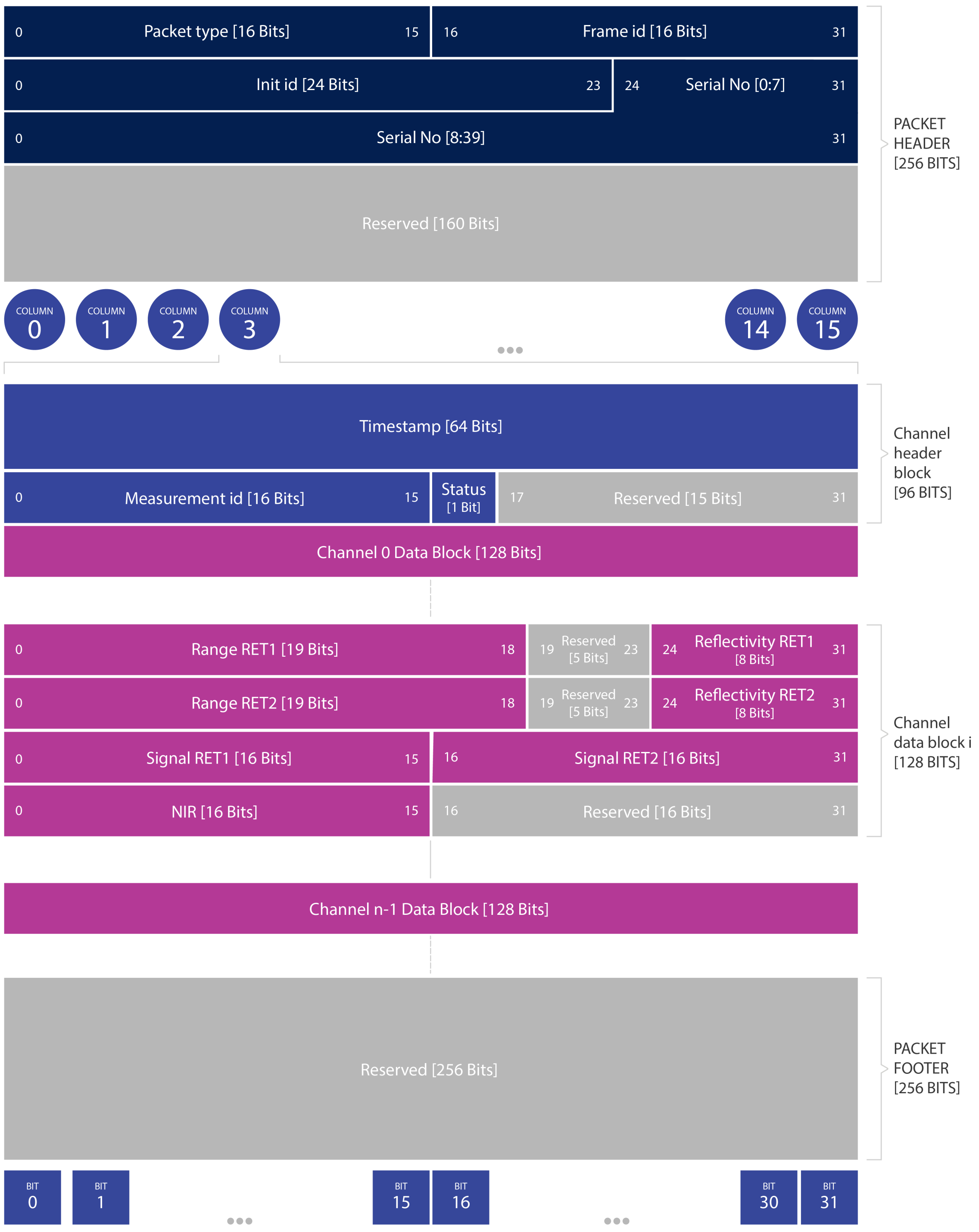


Figure 6.4: Dual Return Data Packet Configuration

6.1.6 Packet Size Calculation (Configurable)

Packet size can be calculated by the following formula:

`packet_header_size` + `columns_per_packet` * (`measurement_header_size` + `pixels_per_column` * `channel_data_block_size`) + `packet_footer_size`

For example:

- 32 + 16 * (12 + n * s) + 32 bytes
  - `packet_header_size` = 32 bytes
  - `columns_per_packet` = 16
  - `measurement_header_size` = 12 bytes
  - **n** is `pixels_per_column` (correspond to the number of channels: 128 for OS1-128)
  - **s** is the size of a channel data block (16 bytes for RNG19_RFL8_SIG16_NIR16_DUAL configuration)
  - `packet_footer_size` = 32 bytes

The table below calculates the data of all products operating at the highest lidar modes, 2048x10 or 1024x20 for Single Return and Low Data Rate profiles and 1024x10 for Dual Return profile and assuming a default azimuth window of 360º. Providing a custom azimuth window can further lower data rate. See the *Azimuth Window* section for details on setting a custom azimuth window.

Table 6.2: Packet Size (Bytes) and Data Rate (Mbps) Breakdown - Configurable Data Packet Format

Product	Single Return Lidar Packet size (bytes)	Dual Return Lidar Packet size (bytes)	Low Data Rate Lidar Packet size (bytes)	Single Return Data rate (Mbps)	Dual Return Data rate (Mbps)	Low Data Rate (Mbps)
OS0-32, OS1-32, OS2-32	6400	8448	2304	65.57	43.62	23.63
OS0-64, OS1-64, OS2-64	12544	16640	4352	128.49	85.24	44.60
OS0-128, OS1-128, OS2-128	24832	33024	8448	254.32	169.12	86.55

Table 6.3: Packet Rate (Hz) Breakdown - Configurable Data Packet Format

Product	Single Return Lidar Packet rate (Hz)	Dual Return Lidar Packet rate (Hz)	Low Data Rate Lidar Packet rate (Hz)
OS0-32, OS1-32, OS2-32	1280	640	1280
OS0-64, OS1-64, OS2-64	1280	640	1280
OS0-128, OS1-128, OS2-128	1280	640	1280

6.2 LEGACY Data Packet Format

On firmware version 2.3 the data packet format by default is set to `LEGACY`. This option is backward compatible and is supported on earlier hardware versions as well. The data profile can be modified by changing the configuration parameter `udp_profile_lidar`.

6.2.1 Lidar Data Format

Note: Gen 1 OS1-16 and OS1-32 customers should note that upgrading to firmware v2.0.0 or higher will change their lidar packet format which reduces their data rates which is not backwards compatible with pre-v2.0.0 clients.

By default the configuration parameter `udp_profile_lidar` is set to `LEGACY`. In this mode, lidar packets consist of 16 Measurement Blocks and vary in size relative to the number of channels in the sensor. The packet rate is dependent on the lidar mode. Words are 32 bits in length and little endian. By default, lidar UDP data is forwarded to Port `7502`. Please refer to the **TCP API** portion of this manual for more information on setting this parameter. Alternately, this mode can also be configured via the Web Interface.

Lidar frames are composed of 512, 1024, or 2048 measurement blocks, depending upon lidar mode and are completely deterministic in number per frame and their monotonic order and position within lidar data packets. This determinism allows for efficient lookup table-based decoding in clients.

Each Measurement Block contains:

Header Block [128 bits]

- **Timestamp** [64 bit unsigned int] - timestamp of the measurement in nanoseconds.
- **Measurement ID** [16 bit unsigned int] - a sequentially incrementing measurement counting up from 0 to 511, or 0 to 1023, or 0 to 2047 depending on lidar_mode.
- **Frame ID** [16 bit unsigned int] - index of the lidar scan. Increments every time the sensor completes a rotation, crossing the zero point of the encoder.

- **Encoder Count** [32 bit unsigned int] - an azimuth angle as a raw encoder count, starting from 0 with a max value of 90,111 - incrementing 44 ticks every azimuth angle in 2048 mode, 88 ticks in 1024 mode, and 176 ticks in 512 mode. Note: the encoder count is redundant with the Measurement ID and will be deprecated in the future.

Channel Data Blocks [96 bits each]

- **Range** [32 bit unsigned int - only 20 bits used] - range in millimeters, discretized to the nearest 1 millimeters.
- **Calibrated Reflectivity** [8 bit unsigned int] - sensor Signal Photons measurements are scaled based on measured range and sensor sensitivity at that range, providing an indication of target reflectivity. Note that calibrated reflectivity has certain hardware requirements. Please refer to the *Calibrated Reflectivity* section for more details.
- **Signal Photons** [16 bit unsigned int] - signal intensity photons in the signal return measurement are reported.
- **Near Infrared Photons** [16 bit unsigned int] - NIR photons related to natural environmental illumination are reported.

Measurement Block Status [32 bits]- indicates whether the measurement block contains valid or zero-padded data in its channels' Data Blocks. Valid = 0xFFFFFFFF, Padded = 0x0. If the Measurement Block Status is Padded (e.g. in the case of channel data being dropped or if the Measurement Block is outside of the azimuth window), values within the Channel Data Blocks will be 0, but values within the Header Block remain valid.

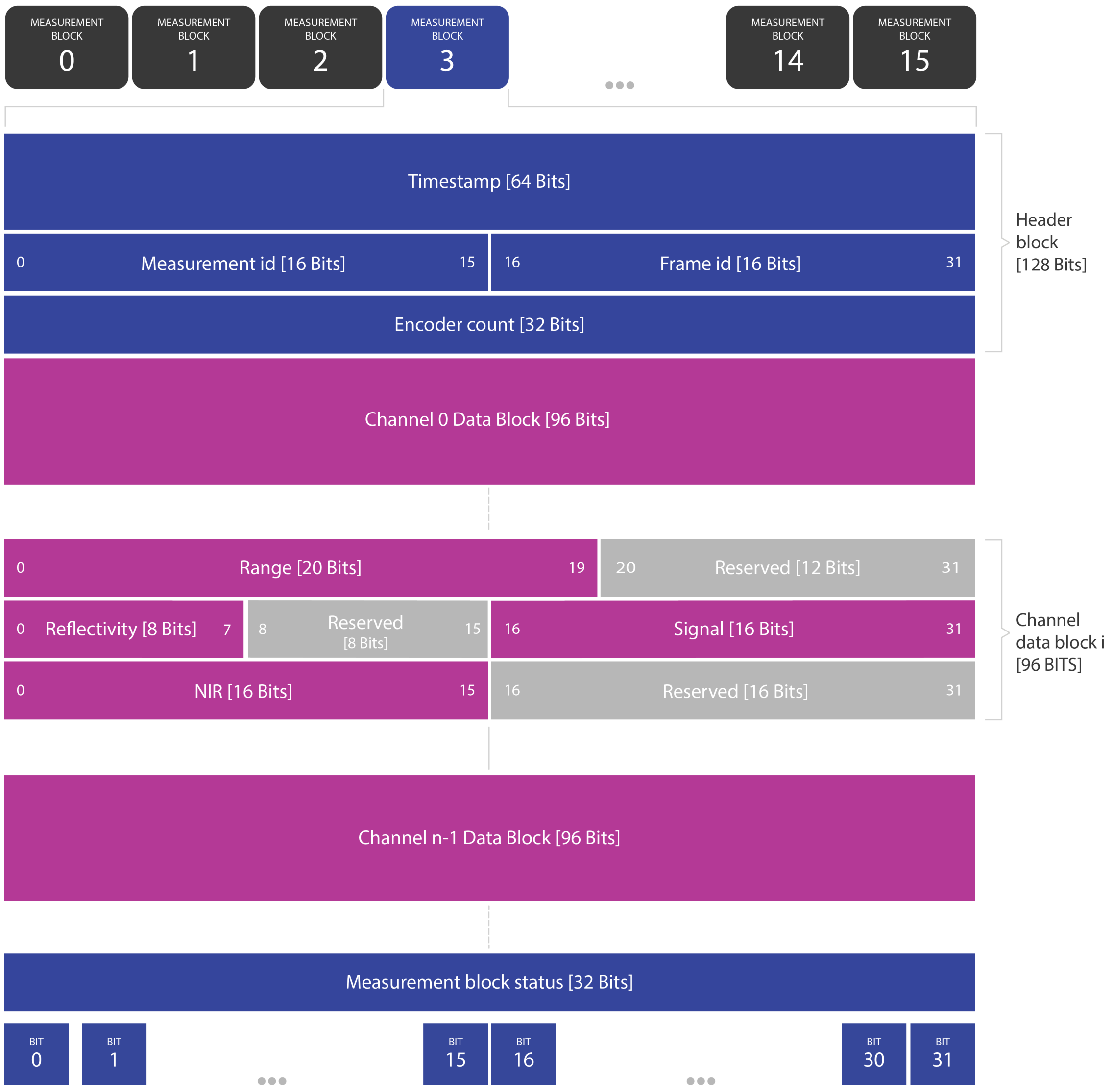


Figure 6.5: LEGACY Packet Data Configuration

6.2.2 Packet Size Calculation (LEGACY)

The table below shows the lidar data packet size breakdown for all products when `LEGACY` profile is configured. Since the size of the measurement block varies proportional to the number of channels in a sensor, all sensors with the same number of channels have the same lidar packet data structure and size.

Table 6.4: Packet Size Breakdown- LEGACY Profile

Product	**Number of words in Measurement Block**	**Size of single Measurement Block (Bytes)**	**Size of lidar packet (Bytes)**
OS1-16	53	212	3,392
OS0-32, OS1-32, OS2-32	101	404	6,464
OS0-64, OS1-64, OS2-64	197	788	12,608
OS0-128, OS1-128, OS2-128	389	1,556	24,896

The table below calculates the data of all products operating at the highest lidar modes, `2048x10` or `1024x20` for `LEGACY` profile assuming a default azimuth window of 360º. Providing a custom azimuth window can further lower data rate. See the *Azimuth Window* section for details on setting a custom azimuth window.

Table 6.5: Data Rate- LEGACY Profile

Product	**LEGACY Lidar packet size (Bytes)**	**LEGACY Lidar packets rate (Hz)**	**LEGACY Data Rate (Mbps)**
OS1-16	3392	1280	34.77
OS0-32, OS1-32, OS2-32	6464	1280	66.23
OS0-64, OS1-64, OS2-64	12608	1280	129.14
OS0-128, OS1-128, OS2-128	24896	1280	254.97

6.3 Calibrated Reflectivity

Starting in firmware v2.1.0, sensors have an 8-bit reflectivity data field. Existing sensors in the field that update to v2.1.0 will have default calibration values pushed to them. Sensors that have been factory calibrated for reflectivity will have a higher accuracy of reflectivity.

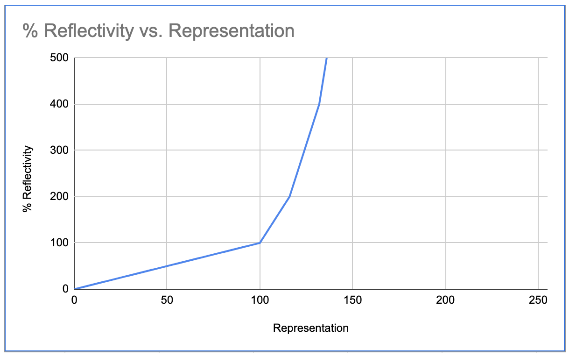
The command `get_calibration_status` will return the status of your sensor calibration. The calibration status is returned with the following format:

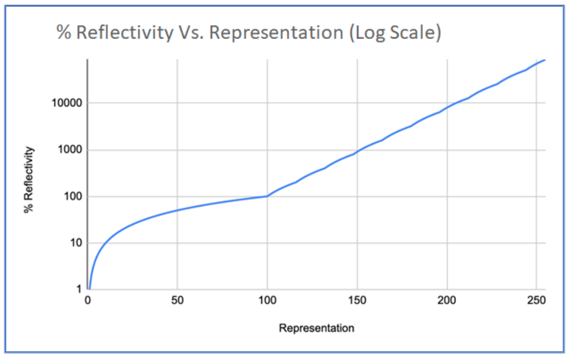
```
{
  "reflectivity":
  {
      "valid": "true: if factory calibrated for better accuracy, false: if not calibrated -- using default
                      values and likely has less accuracy",
      "timestamp": "Date when the calibration has been performed"
  }
}
```

Please contact your support@ouster.io if you have questions on whether your sensor is hardware-enabled for calibrated reflectivity.

6.3.1 Reflectivity Data Mapping

Reflectivity values between 0-100 are linearly mapped for lambertian targets with values between 0% and 100% reflectivity. Values between 101-255 are mapped as log $_2$ with linear interpolation between logarithmic points for retroreflective targets. The 255 value corresponds to a retroreflector 864x stronger than a 100% lambertian target. The charts below show the mapping functions.





Representation	% Reflectivity	Representation	% Reflectivity	Representation	% Reflectivity	Representation	% Reflectivity
0-100	0-100	139	575	178	3000	217	16800
101	106.25	140	600	179	3100	218	17600
102	112.5	141	625	180	3200	219	18400
103	118.75	142	650	181	3400	220	19200
104	125	143	675	182	3600	221	20000
105	131.25	144	700	183	3800	222	20800
106	137.5	145	725	184	4000	223	21600
107	143.75	146	750	185	4200	224	22400
108	150	147	775	186	4400	225	23200
109	156.25	148	800	187	4600	226	24000
110	162.5	149	850	188	4800	227	24800
111	168.75	150	900	189	5000	228	25600
112	175	151	950	190	5200	229	27200
113	181.25	152	1000	191	5400	230	28800
114	187.5	153	1050	192	5600	231	30400
115	193.75	154	1100	193	5800	232	32000
116	200	155	1150	194	6000	233	33600
117	212.5	156	1200	195	6200	234	35200
118	225	157	1250	196	6400	235	36800
119	237.5	158	1300	197	6800	236	38400
120	250	159	1350	198	7200	237	40000
121	262.5	160	1400	199	7600	238	41600
122	275	161	1450	200	8000	239	43200
123	287.5	162	1500	201	8400	240	44800
124	300	163	1550	202	8800	241	46400
125	312.5	164	1600	203	9200	242	48000
126	325	165	1700	204	9600	243	49600
127	337.5	166	1800	205	10000	244	51200
128	350	167	1900	206	10400	245	54400
129	362.5	168	2000	207	10800	246	57600
130	375	169	2100	208	11200	247	60800
131	387.5	170	2200	209	11600	248	64000
132	400	171	2300	210	12000	249	67200
133	425	172	2400	211	12400	250	70400
134	450	173	2500	212	12800	251	73600
135	475	174	2600	213	13600	252	76800
136	500	175	2700	214	14400	253	80000
137	525	176	2800	215	15200	254	83200
138	550	177	2900	216	16000	255	86400

6.4 IMU Data Format

IMU UDP Packets are 48 Bytes long and by default are sent to Port `7503` at 100 Hz. Values are little endian.

Note: IMU data format is the same regardless of the lidar data profile selected by the user.

Each IMU data block contains:

- **IMU Diagnostic Time** [64 bit unsigned int] - timestamp of monotonic system time since boot in nanoseconds.
- **Accelerometer Read Time** [64 bit unsigned int] - timestamp for accelerometer time relative to *timestamp_mode* in nanoseconds.
- **Gyroscope Read Time** [64 bit unsigned int] - timestamp for gyroscope time relative to *timestamp_mode* in nanoseconds.
- **Acceleration in X-axis** [32 bit float] - acceleration in g.
- **Acceleration in Y-axis** [32 bit float] - acceleration in g.
- **Acceleration in Z-axis** [32 bit float] - acceleration in g.
- **Angular Velocity about X-axis** [32 bit float] - Angular velocity in deg per sec.
- **Angular Velocity about Y-axis** [32 bit float] - Angular velocity in deg per sec.
- **Angular Velocity about Z-axis** [32 bit float] - Angular velocity in deg per sec.

Note that the first timestamp (Words 0,1) is for diagnostics only and is rarely used under normal operation.

The second two timestamps, (Words 2,3) and (Words 4,5), are sampled on the same clock as the lidar data, so should be used for most applications.

Ouster provides timestamps for both the gyro and accelerometer in order to give access to the lowest level information. In most applications it is acceptable to use the average of the two timestamps.

Table 6.6: Data Rate - IMU Data Packet

Product	**IMU packet size** (Bytes)	**IMU packets per second**
OS1-16	48	100
OS0-32, OS1-32, OS2-32	48	100
OS0-64, OS1-64, OS2-64	48	100
OS0-128, OS1-128, OS2-128	48	100

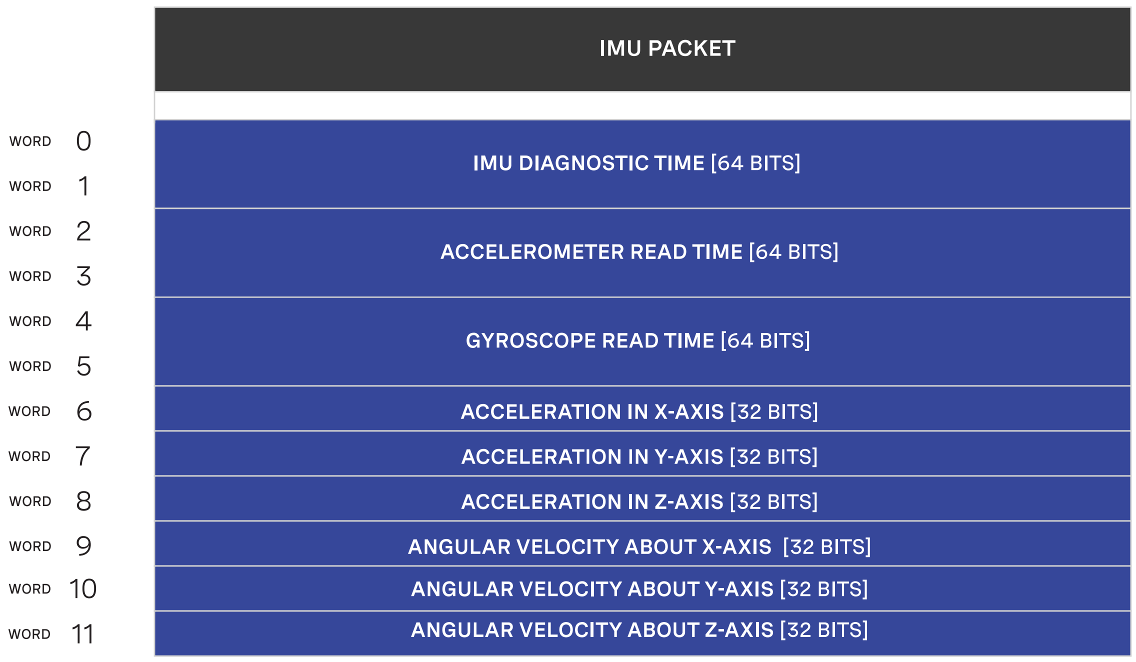

IMU PACKET
WORD 0
WORD 1
IMU DIAGNOSTIC TIME [64 BITS]
WORD 2
WORD 3
ACCELEROMETER READ TIME [64 BITS]
WORD 4
WORD 5
GYROSCOPE READ TIME [64 BITS]
WORD 6
ACCELERATION IN X-AXIS [32 BITS]
WORD 7
ACCELERATION IN Y-AXIS [32 BITS]
WORD 8
ACCELERATION IN Z-AXIS [32 BITS]
WORD 9
ANGULAR VELOCITY ABOUT X-AXIS [32 BITS]
WORD 10
ANGULAR VELOCITY ABOUT Y-AXIS [32 BITS]
WORD 11
ANGULAR VELOCITY ABOUT Z-AXIS [32 BITS]

Figure 6.6: IMU Packet Format

7 Sensor Operations

7.1 Typical Sensor Operation

Described below is the typical sensor state machine operation. When the sensor is powered ON, the sensors start in the initialization phase.

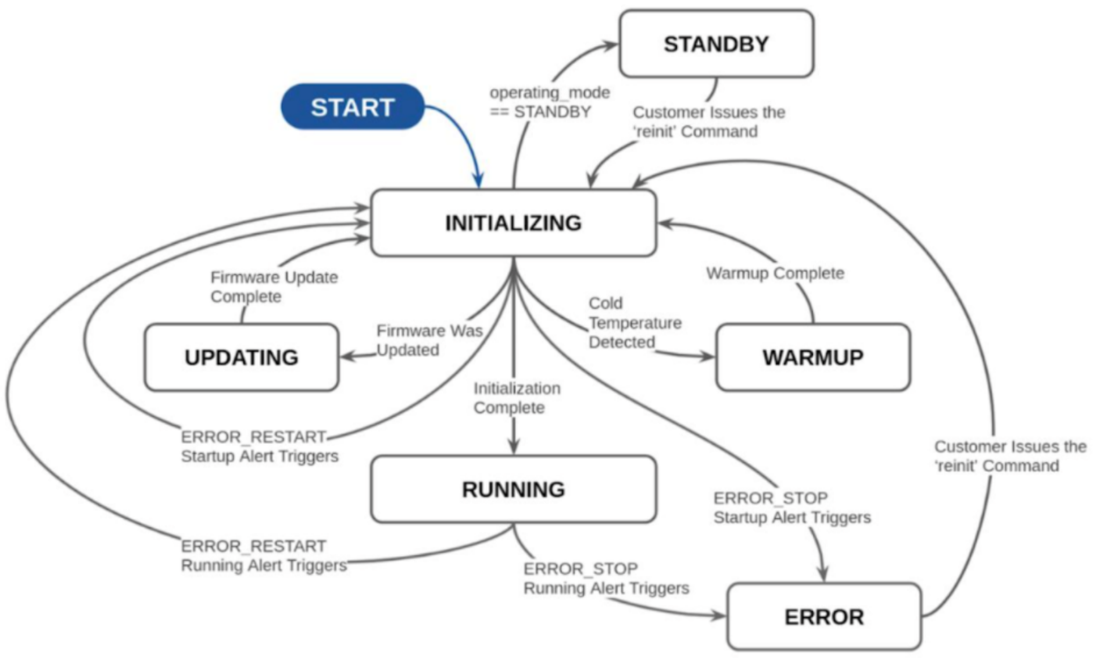

STANDBY
START
operating_mode == STANDBY
Customer Issues the 'reinit' Command
INITIALIZING
Firmware Update Complete
Warmup Complete
Cold Temperature Detected
Firmware Was Updated
UPDATING
WARMUP
Initialization Complete
ERROR_RESTART Startup Alert Triggers
Customer Issues the 'reinit' Command
RUNNING
ERROR_STOP Startup Alert Triggers
ERROR_RESTART Running Alert Triggers
ERROR_STOP Running Alert Triggers
ERROR

Table 7.1: Sensor Operation

Operating State	**Description**
Warm-up	If the sensor detects that its environmental temperature is low it will attempt to self-heat in a warmup state (*Cold Start*) before entering a normal operating state.
Initializing	Startup of Ouster Lidar.
Updating	Only remains in this state temporarily to update the firmware.
Error	If an exception is thrown during initialization or running state, the lidar logs the error.
OFF	Ouster Lidar shut off.
Running	Sensor has completed initialization phase and is now running.
Standby	User enabled low power operating mode of the sensor.

7.2 Sensor Telemetry

Sensor telemetry refers to sensor system state information that changes with time i.e., temperature, voltage, etc. Users can monitor this data live or for diagnostics and take precautionary measures if needed. This feature is only available on FW 2.3 and later. This information can be obtained from running the command `get_telemetry` as shown in the example below.

```
{
    "input_current_ma": 758,
    "input_voltage_mv": 23606,
    "internal_temperature_deg_c": 45,
    "phase_lock_status": "DISABLED",
    "timestamp_ns": 2962666299310
}
```

Table 7.2: Example Sensor Telemetry

Fields	**Notes**
Timestamp	Timestamp from the FPGA measured in `ns` (Nanoseconds)
Lidar Input Voltage	Input voltage `mv` (Millivolt) that is provided to the sensor
Lidar Input Current	Input current `ma` (Milliamp) that is provided to the sensor
Internal Temperature	Internal base board temperature `°C` (Degree Celsius).
Phase Lock Status	Different codes to specify phase lock status and issues related to phase locking (`LOCKED`, `LOST`, `DISABLED`)

Note: Sensor telemetry will be available on all sensor revisions but internal base board temperature value can only be measured with **Rev 06** and above sensors.

Note: **Phase lock** output will not indicate loss of lock if the PTP source is lost.

7.3 Cold Start

There is software-enabled capability starting with firmware version 2.0.0 for the Ouster sensor to power-up from lower temperatures. If the sensor detects that its environmental temperature is low, it will attempt to self-heat in a warmup state before entering a normal operating state.

7.3.1 Hardware Requirements

Gen 1 sensors are not cold start-compatible on any firmware. While all sensors will attempt to start at lower exhibit cold start behavior by going into the warmup state, only Gen 2 sensors are able to successfully exit the warmup state into the normal operating state.

7.3.2 Cold Start Operation

There is nothing for the user to change about the sensor configuration to use this feature. The sensor will automatically begin its warmup process in the coldest parts of its operating temperature range.

Table 7.3: Cold Start

Product Line	Min Temp Specs
OS0	• -40°C min operating temp • 8 mins to `SENSOR_RUNNING` • 12 mins to lasers at temp (full range) • 28W peak power
OS1	• -40°C min operating temp • 8 mins to `SENSOR_RUNNING` • 12 mins to lasers at temp (full range) • 28W peak power
OS2	• -20°C min operating temp • 15 mins to `SENSOR_RUNNING` • 15 mins to lasers at temp (full range) • 30W peak power

7.3.3 Indications and Alerts

In a cold start scenario, the sensor will have a short warmup phase; we've added in the additional `"WARMUP"` status to indicate when the sensor is warming up.

```
$ nc os-122201000998 7501
get_sensor_info
     {
     "build_date": "2022-07-29T23:56:15Z",
     "build_rev": "v2.4.0-omega.2",
     "image_rev": "ousteros-image-prod-aries-v2.4.0-omega.2+20220730010801.staging",
     "initialization_id": 9599937,
     "prod_line": "OS-1-128",
     "prod_pn": "840-103575-06",
     "prod_sn": "122201000998",
     "status": "RUNNING"
 }
```

The following alerts are related to cold start

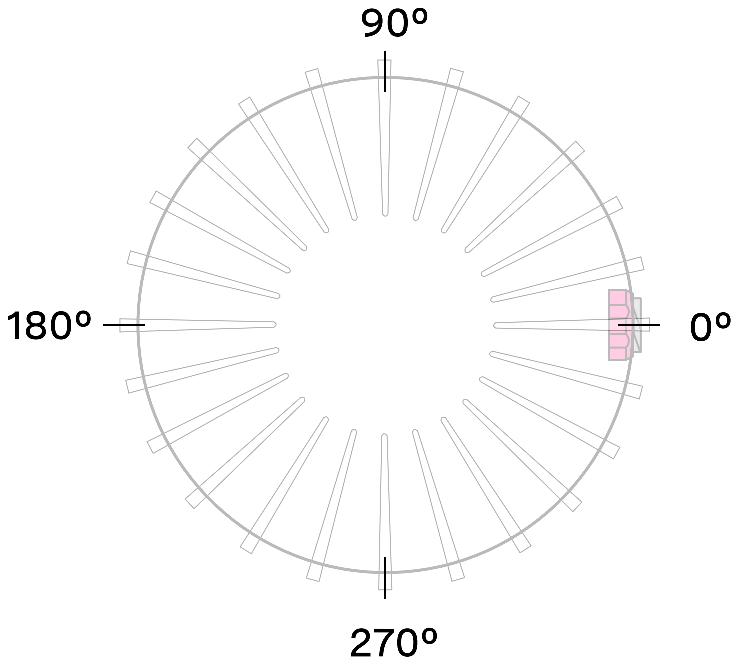
Table 7.4: Cold Start Alerts

ID	Category	Level	Description
0x01000053	WARMUP_ISSUE	ERROR	Sensor warmup process has failed.
0x0100004F	WARMUP_ISSUE	WARNING	Sensor warmup process is taking longer than expected; please ensure sensor is thermally constrained per requirements.

7.4 Azimuth Window

Configuring the azimuth window is a feature to only turn on the UDP lidar data within a region of interest. The region of interest is defined by a min bound and a max bound, both in millidegrees. As a reminder, angles in this frame increment counterclockwise when viewed from the top. Below is the Lidar Coordinate Frame from a top-down perspective:

- 0° towards the external connector
- 90° a quarter turn counterclockwise from the connector
- 180° opposite the connector
- 270° three quarter turns counterclockwise from the connector



LIDAR COORDINATE FRAME TOP-DOWN VIEW

Configuring the azimuth window lowers the average output data rate of the sensor but does not affect the peak output data rate of the sensor. It also does not stop the lasers from firing and thus does not have an effect on power consumption or thermals.

7.4.1 Expected Sensor Behavior

The sensor will round the input azimuth window bounds to the nearest *Measurement Block* IDs generating new ID-based bounds. The new bounds are used to mask *Measurement Blocks* in the lidar data packets. Lidar packets containing only masked *Measurement Blocks* are not output, and there may be partially masked *Measurement Blocks* in the two bookended lidar packets in each frame. The *Measurement Block Status* field will indicate the valid or masked/padded *Measurement Blocks* in any partially masked lidar packets. (See the *Lidar Data Packet Format* section for details on the lidar data format.)

The visualized output will contain jagged edges caused by the staggered, nonzero nature of the beam azimuth angles. It is necessary to set more conservative (wider) bounds to push the jagged edges beyond the desired window. This can be determined through trial and error or calculated deterministically with knowledge of the queryable beam azimuth angles.

7.4.2 Azimuth Window Examples

The TCP API Guide lists the command for setting an azimuth window. Below are example settings.

The command syntax is as follows:

```
set_config_param azimuth_window [min_bound_millidegrees, max_bound_millidegrees]
```

Default settings of 360º window:

```
set_config_param azimuth_window [0, 360000]
```

Set a region of interest between 0º to 180º:

```
set_config_param azimuth_window [0, 180000]
```

Set a region of interest between 270º to 90º with 180º field of view:

```
set_config_param azimuth_window [270000, 90000]
```

Set a region of interest 90º to 270º with 180º field of view:

```
set_config_param azimuth_window [90000, 270000]
```

Set a region of interest between 0º to 90º with 90º field of view:

```
set_config_param azimuth_window [0, 90000]
```

Set a region of interest 90º to 360º with 270º field of view:

```
set_config_param azimuth_window [90000, 0]
```

7.5 Standby Operating Mode

Starting with firmware v2.0.0, the sensor can be commanded in and out of a low-power Standby Operating Mode that can be useful for power, battery, or thermal-conscious applications of the sensor.

The TCP config param `operating_mode` has a default value of `NORMAL`. Setting it to `STANDBY` puts the sensor into Standby Operating Mode upon reinitialization.

7.5.1 Expected Sensor Behavior

Power draw in Standby mode 5W. The motor does not spin, and light is not visible from the window. However, the sensor is on and listening to commands. The sensor status will be `STANDBY`.

7.5.2 Standby Operating Mode Examples

Below are example netcat console command input and responses for several use cases of the Standby mode.

Note: In the examples below, to distinguish between the command and expected response, a dash has been added before the expected response. The actual response will be without the dash.

Set sensor into Standby mode and keep sensor in Standby mode upon power-up at next use:

```
$ nc os-991900123456 7501
set_config_param operating_mode STANDBY
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Set sensor into Standby mode but have sensor start in the default Running mode upon power-up at next use:

```
$ nc os-991900123456 7501
set_config_param operating_mode STANDBY
-set_config_param
reinitialize
-reinitialize
```

Command sensor back into Running mode and save config:

```
$ nc os-991900123456 7501
set_config_param operating_mode NORMAL
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Note: `auto_start_flag` has been **deprecated** with **Firmware 2.4 and later**. For prior Firmware versions `auto_start_flag 0` is equivalent to `operating_mode STANDBY` and `auto_start_flag 1` is equivalent to `operating_mode NORMAL`. Please use `operating_mode` wherever possible in client code.

Warning: Usage of `auto_start_flag` in firmware prior to v2.0.0 has unexpected behavior.

7.6 Signal Multiplier

For Gen 2 sensors with firmware v2.1 or higher, the `signal_multiplier` config parameter allows the user to set a multiplier for the signal strength of the sensor, which corresponds to a maximum allowable azimuth window. Lasers are disabled outside of the maximum allowable azimuth window. By default the sensor has a signal multiplier value of `1`.

7.6.1 Use Cases

The config parameter `signal_multiplier <1/2/3>` sets the signal multiplier value. For 2x and 3x multipliers, the `azimuth_window [int, int]` parameter sets the azimuth window that the lasers will be enabled in. The higher the signal multiplier value, the smaller the maximum azimuth window can be.

Table 7.5: Relation

Signal Multiplier Value	Max Azimuth Window
1 (Default)	360º
2	180º
3	120º

All sensors have equivalent power draw and thermal output when operating at the max azimuth window for a particular signal multiplier value. Therefore, using an azimuth window that is smaller than the maximum allowable azimuth window with a particular signal multiplier value (excluding 1x) can reduce the power draw and thermal output of the sensor. However, while this can increase the max operating temp of the sensor, it can also degrade the performance at low temps. This discrepancy will be resolved in a future firmware. The table below outlines some example use cases.

Table 7.6: Example Use Cases

Use Case	`signal_multiplier` Parameter	`azimuth_window` Parameter
Signal boost	3	[0,120000]
Signal boost with power draw reduction	2	[0,90000]

7.6.2 Expected Behavior

If the sensor has signal multiplier of `1`, lasers will be enabled for all 360º of the window, regardless of the `azimuth_window` set.

If an invalid pair of signal multiplier and azimuth window values are set, the sensor will throw an error. If a valid pair of values are set, upon reinitializing, the sensor will operate in the signal multiplier mode.

7.6.3 Examples

The following shows the netcat console input commands and responses for some configuration examples.

Note: In the examples below, to distinguish between the command and expected response, a dash has been added before the expected response. The actual response will be without the dash.

Set sensor in 3x signal mode with 120º HFoV:

```
$ nc sensor1_hostname 7501
set_config_param set_config_param signal_multiplier 3
-set_config_param
set_config_param azimuth_window [120000, 240000]
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Sensor will throw an error if invalid parameters are set:

```
$ nc sensor1_hostname 7501
set_config_param signal_multiplier 5
-error: signal_multiplier must be between 1 and 3, inclusive
set_config_param signal_multiplier 3
-set_config_param
set_config_param azimuth_window [120000, 300000]
-set_config_param
reinitialize
-error: for signal_multiplier value of 3, azimuth_window must span a maximum of 120000 millidegrees.
        Current azimuth_window [120000, 300000] spans 180000 millidegrees.
```

7.7 Sensor Performance by Operating Configuration

Depending upon the sensor's lidar mode and signal multiplier setting, the sensor performance will vary from its baseline as listed on the datasheet. This section will present the estimated performance multiplier depending on the sensor and the operating configuration.

7.7.1 Estimated range multiplier

When using a signal multiplier higher than 1x and depending on the lidar mode, the sensor will get a range increase. The following tables present an estimated range multiplier depending on the operating configuration.

OS0 and OS1

For the OS0 and OS1 sensors the baseline is the 1024x10 mode

Frame Rate / Horiz. Res.	**512**			**1024**			**2048**		
Signal multiplier	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**
10 Hz	1.19	1.41	1.57	1.00	1.19	1.32	0.84	1.00	1.11
20 Hz	1.00	1.19	1.32	0.84	1.00	1.11	NA		

OS2

For OS2 sensors the baseline is the 2048x10 mode.

Frame Rate / Horiz. Res.	**512**			**1024**			**2048**		
Signal multiplier	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**
10 Hz	1.41	1.68	1.86	1.19	1.41	1.57	1.00	1.19	1.32
20 Hz	1.19	1.41	1.57	1.00	1.19	1.32	NA		

Note: The values in the tables above are given for guidance only. The only specs guaranteed are the ones defined in the sensor datasheet for a specific mode.

Maximal representable range

Depending upon the signal multiplier, the maximal representable range of the sensor will be different. The table below shows the maximal representable range values for each sensor type and multiplier value.

Table 7.7: Maximum Range

Signal Multiplier Value	**OS0**	**OS1**	**OS2**
1x	270 m	270 m	465 m
2x	135 m	135 m	232 m
3x	90 m	90 m	155 m

Range returns beyond the maximal representable range will experience range aliasing. Therefore, these modes are only recommended in scenarios where there will not be any returns beyond the maximal representable range.

7.7.2 Estimated precision multiplier

When using a signal multiplier higher than 1x and depending on the lidar mode, the sensor will get a precision improvement. The following tables present an estimated precision multiplier depending on the operating configuration. Please refer to the *Signal Multiplier* section for more details.

OS0 and OS1

Frame Rate / Horiz. Res.	**512**			**1024**			**2048**		
Signal multiplier	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**
10 Hz	0.71	0.50	0.41	1.00	0.71	0.58	1.41	1.00	0.82
20 Hz	1.00	0.71	0.58	1.41	1.00	0.82	NA		

OS2

Frame Rate / Horiz. Res.	**512**			**1024**			**2048**		
Signal multiplier	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**	**1x**	**2x**	**3x**
10 Hz	0.50	0.35	0.29	0.71	0.50	0.41	1.00	0.71	0.58
20 Hz	0.71	0.50	0.41	1.00	0.71	0.58	NA		

8 Multi-Sensor Synchronization

8.1 Phase Lock

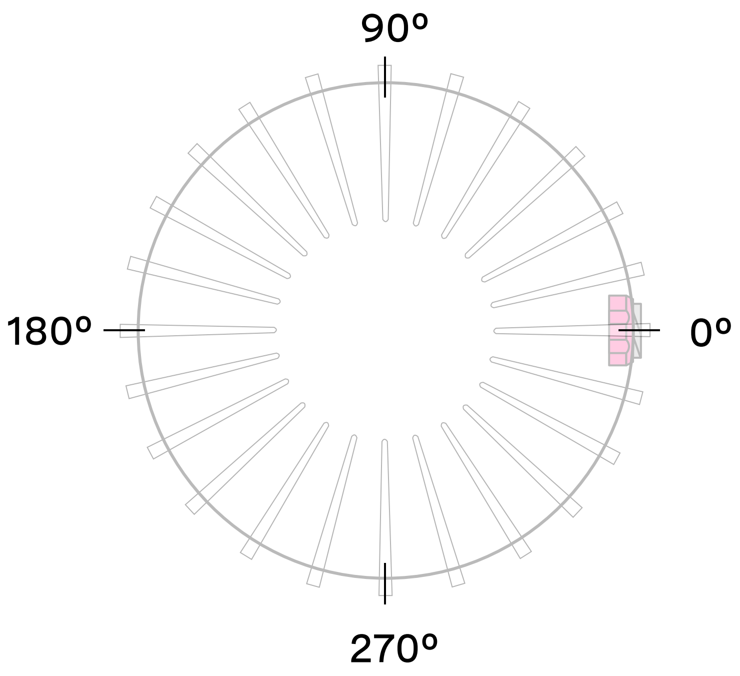
Phase locking allows a sensor to consistently pass through a specific angle at the top, tenth (10 Hz mode), or fifth (20 Hz mode) of a second on each rotation. The phase lock control loop runs at 1000 Hz. Phase locking is useful for synchronizing a sensor with other devices including camera, radar, and other lidar.

A sensor must first be time-synchronized from an external source and must be in either the `TIME_FROM_PTP_1588` or `TIME_FROM_SYNC_PULSE_IN` *timestamp_mode* before entering phase lock.

8.1.1 Phase Locking Reference Frame

Phase locking commands use angles defined in the Lidar Coordinate Frame in millidegrees. As a reminder, angles in this frame increment counterclockwise when viewed from the top. Below is the Lidar Coordinate Frame from a top-down perspective:

- 0º towards the external connector
- 90º a quarter turn counterclockwise from the connector
- 180º opposite the connector
- 270º three quarter turns counterclockwise from the connector



LIDAR COORDINATE FRAME TOP-DOWN VIEW

8.1.2 Phase Locking Commands

The TCP API Guide lists the two commands needed to achieve phase lock.

Command to enable or disable phase lock:

By default, `phase_lock_enable` is `false`

```
set_config_param phase_lock_enable <true/false>
```

Command to set the phase lock offset angle in the Lidar Coordinate Frame:

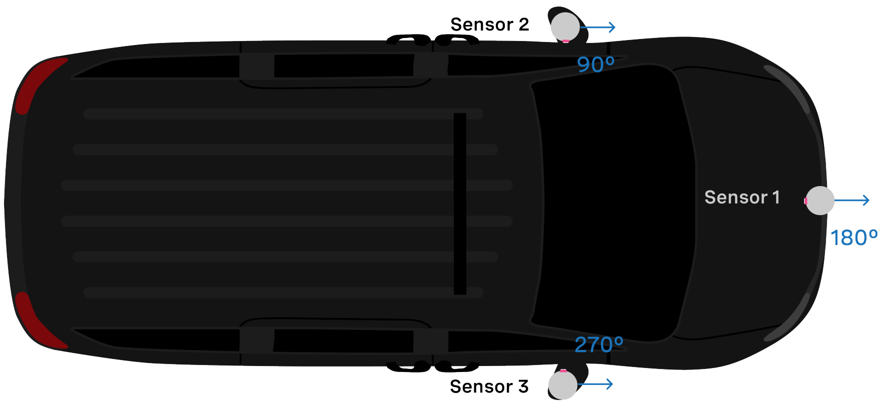
By default, `phase_lock_offset` value is `0`

Note: `<angle_in_millidegrees>` is an integer from `0` to `360000`

```
set_config_param phase_lock_offset <angle_in_millidegrees>
```

8.1.3 Multi-sensor Example

In this example below, we are trying to phase lock all three sensors on the car so that they point towards the front of the car at the same time. Note that their external connectors point in different directions.



Assuming the three sensors are properly time synchronized via an external source, the following shows the netcat console input commands and responses from configuring the sensors so that they point forward at the same time.

Note: In the examples below, to distinguish between the command and expected response, a dash has been added before the expected response. The actual response will be without the dash.

Set Sensor 1 to phase lock at 180º:

```
$ nc sensor1_hostname 7501
set_config_param phase_lock_enable true
```

(continues on next page)

(continued from previous page)

```
-set_config_param
set_config_param phase_lock_offset 180000
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Set Sensor 2 to phase lock at 90º:

```
$ nc sensor2_hostname 7501
set_config_param phase_lock_enable true
-set_config_param
set_config_param phase_lock_offset 90000
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Set Sensor 2 to phase lock at 270º:

```
$ nc sensor3_hostname 7501
set_config_param phase_lock_enable true
-set_config_param
set_config_param phase_lock_offset 270000
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

8.1.4 Accuracy

The following chart shows the expected angular position accuracy under normal operating conditions.

Product Line	**Accuracy**	
	10 Hz	**20 Hz**
OS0 and OS1 (Gen 1 and Gen 2)	0.5˚	0.5˚
OS2	5˚	10˚

8.1.5 Phase Locking Alerts

The following alerts related to phase locking errors are listed below. For the full list of alerts and errors see the *Alerts and Errors* section in the Appendix.

Table 8.1: Phase Lock Alerts

id	category	level	description
0x01000050	MOTOR_CONTROL	WARN-ING	The phase lock offset error has exceeded the threshold.
0x01000051	MOTOR_CONTROL	ERROR	The phase lock control failed to achieve a lock multiple times; please contact Ouster at https://ouster.com/tech-support.
0x01000024	STARTUP	ERROR	The phase lock control failed to achieve a lock during startup.

Note: For information on how to mitigate crosstalk between different Ouster lidars in the same system refer to *Inter-sensor Interference Mitigation* section of this manual.

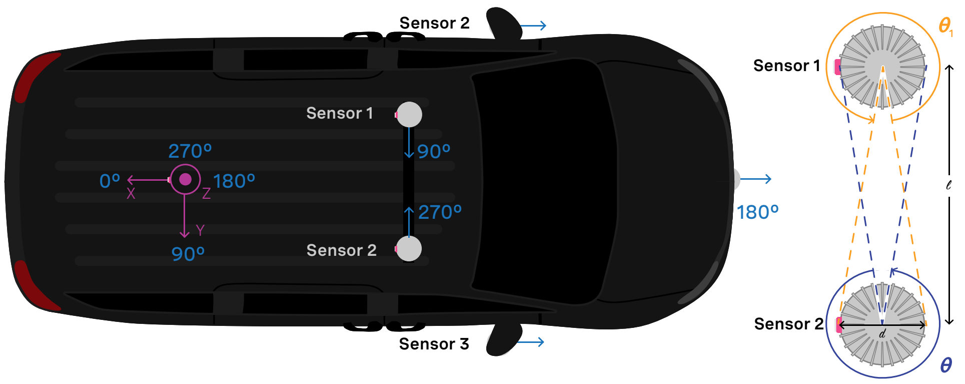
8.2 Inter-sensor Interference Mitigation

Inter-sensor crosstalk occurs when two sensors are operating close together and they interpret each other's laser pulses as their own. Mitigating crosstalk between two sensors is a two step process:

1) Phase lock the two sensors
2) Set azimuth window on each sensor so that they don't send data when they are pointing at each other

8.2.1 Two Sensor Example

In this example below, we are trying to mitigate inter-sensor crosstalk between Sensor 1 and Sensor 2 on the car. Both of their connectors are facing towards the back of the car. The Lidar Coordinate Frame is printed on the back of the vehicle for reference.



First and foremost, placing a physical barrier between the two sensors is the best option to mitigate cross talk in this example and most scenarios. If this is not possible, we can use the phase locking feature to eliminate the problem. Crosstalk only occurs when one sensor shines its lasers into the window of another sensor. The goal of phase locking is to force the sensors to point at each other simultaneously so that crosstalk occurs when sensors aren't generating important data about the environment.

1. Time synchronize the two sensors via an external source. See the *Time Synchronization* section for more details on time synchronizing sensors with an external GPS or via PTP.
2. Phase lock both sensors such that they point directly at each other at the same time. In this case, we want Sensor 1 to be pointing at 90° at the same time that Sensor 2 is pointing at 270°. The example netcat console output would look like below.

Note: In the examples below, to distinguish between the command and expected response, a dash has been added before the expected response. The actual response will be without the dash.

Example: Set Sensor 1 to phase lock at 90°:

```
$ nc sensor1_hostname 7501
set_config_param phase_lock_enable true
```

(continues on next page)

(continued from previous page)

```
    -set_config_param
    set_config_param phase_lock_offset 90000
    -set_config_param
    reinitialize
    -reinitialize
    save_config_params
    -save_config_params
```

Example: Set Sensor 2 to phase lock at 270º:

```
$ nc sensor2_hostname 7501
    set_config_param phase_lock_enable true
    -set_config_param
    set_config_param phase_lock_offset 270000
    -set_config_param
    reinitialize
    -reinitialize
    save_config_params
    -save_config_params
```

3. Set an azimuth window for both sensors. In this case, the region of interest for Sensor 1 is θ_1 and the region of interest for Sensor 2 is θ_2

The calculation for θ_1 and θ_2 is as follows:

$$\theta_1 = \theta_2 = 360^\circ - 2 \cdot \text{arctan}\,\frac{d}{l}$$

In this case, if the two sensors were placed a distance of 100 mm apart, $360^\circ - 2 \cdot \text{arctan}\,\frac{81}{1000} = 360^\circ - 78^\circ = 282^\circ$ We want to set azimuth window of size 282º for the two sensors, so that they do not send data in the 78º where they would point at each other. Sensor 1's azimuth window is the 282º centered around 270º. Sensor 2's region of interest is the 282º centered around 90º.

Sensor 1's azimuth window starts at 129º and follows the CCW direction to end at 51º:

```
$ nc sensor1_hostname 7501
set_config_param azimuth_window [129000, 51000]
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Sensor 2's azimuth window starts at 309º and follows the CCW direction to end at 231º:

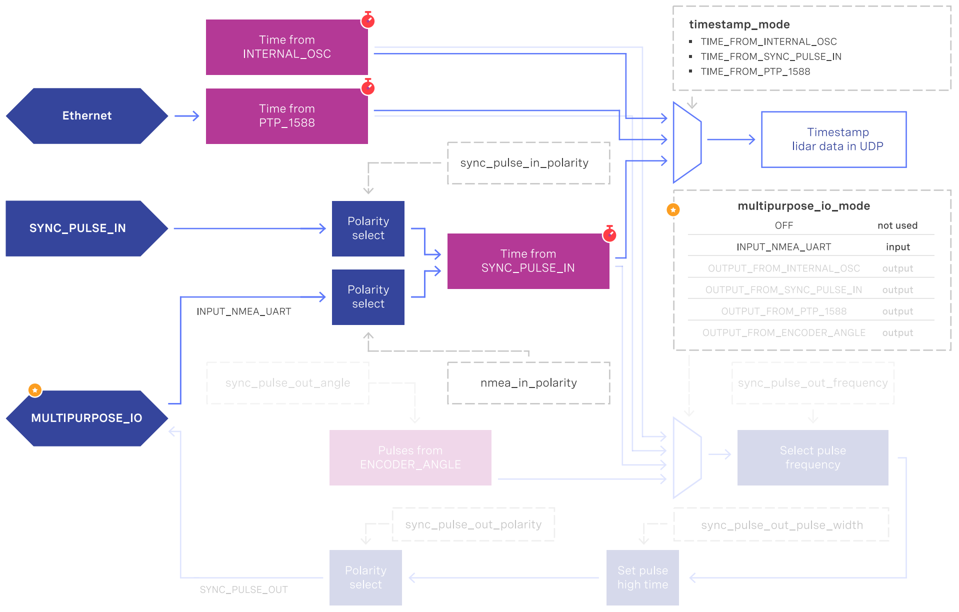
```
$ nc sensor2_hostname 7501
set_config_param azimuth_window [309000, 231000]
-set_config_param
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Product Line	Diameter	
	At window	**At base including fins**
OS0 and OS1 (Gen1 and Gen2)	81 mm	88 mm
OS2	111 mm	121 mm

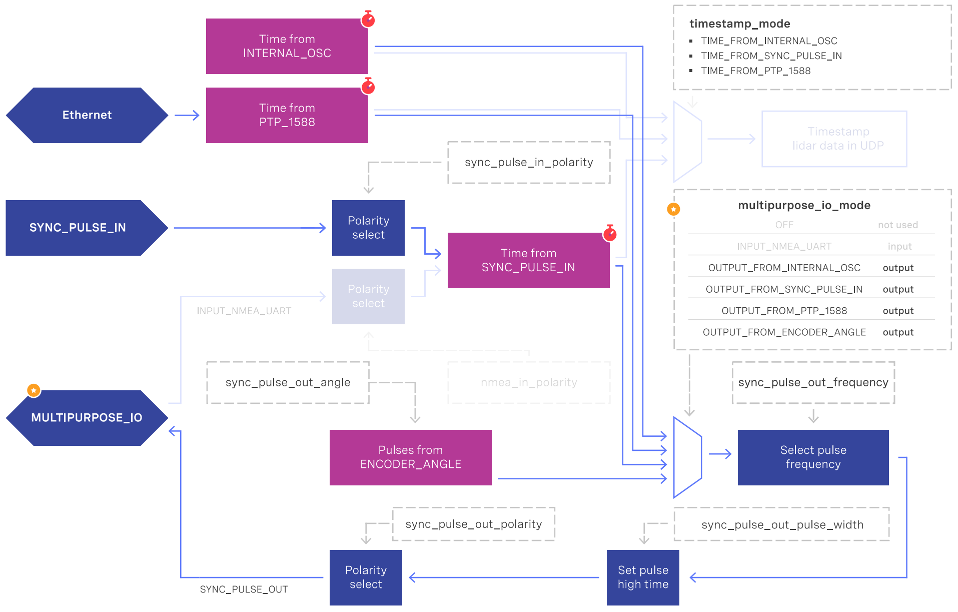
9 Time Synchronization

9.1 Timing Overview Diagram

Signal path with MULTIPURPOSE_IO set as input



Signal path with MULTIPURPOSE_IO set as output



9.2 Sensor Time Source

- All lidar and IMU data are timestamped to a common timer with 10 nanosecond precision.
- The common timer can be programmed to run off one of three clock sources:
  - An internal clock derived from a high accuracy, low drift oscillator.
  - An opto-isolated digital input from the external connector for timing off an external hardware trigger such as a GPS. The polarity of this input signal is programmable. For instance, both a GPS PPS pulse and a 30 Hz frame sync from an industrial camera can supply a timing signal to the sensor
  - Using the IEEE 1588 Precision Time Protocol. PTP provides the convenience of configuring timing over a network that supports IEEE 1588 with no additional hardware signals.

9.2.1 Setting Ouster Sensor Time Source

The source for measurement timestamps can be configured using the `timestamp_mode` TCP command. The available modesare described below:

Table9.1: Timestamp Modes

Command	Response
`TIME_FROM_INTERNAL_OSC`	Use the internal clock. Measurements are time stamped with ns since power-on. Free running counter based on the sensor's internal oscillator. Counts seconds and nanoseconds since sensor turn on, reported at ns resolution (both a second and nanosecond register in every UDP packet), but min increment is on the order of 10 ns.
`TIME_FROM_SYNC_PULSE_IN`	A free running counter synced to the SYNC_PULSE_IN input counts seconds (# of pulses) and nanoseconds since sensor turn on. If `multipurpose_io_mode` is set to `INPUT_NMEA_UART` then the seconds register jumps to time extracted from a NMEA $GPRMC message read on the `multipurpose_io` port. Reported at ns resolution (both a second and nanosecond register in every UDP packet), but min increment is on the order of 10 ns.
`TIME_FROM_PTP_1588`	Synchronize with an external PTP master. A monotonically increasing counter that will begin counting seconds and nanoseconds since startup. As soon as a 1588 sync event happens, the time will be updated to seconds and nanoseconds since 1970. The counter must always count forward in time. If another 1588 sync event happens the counter will either jump forward to match the new time, or slow itself down. It is reported at ns resolution (there is both a second and nanosecond register in every UDP packet), but the minimum increment varies.

If configuring the sensor to synchronize time from an external sync pulse, the pulse polarity can be specified as described in the TCP API Guide. Pulse-in frequency is assumed to be 1 Hz. For example, the below commands will set the sensor to expect an active low pulse and configure the seconds timestamp to be pulse count since sensor startup:

- `set_config_param timestamp_mode TIME_FROM_SYNC_PULSE_IN`
- `set_config_param sync_pulse_in_polarity ACTIVE_LOW`
- `reinitialize`

To configure the multipurpose-io port of the sensor to accept an external NMEA UART message, the `multipurpose_io_mode` parameter must be set to `INPUT_NMEA_UART` as described in *External Trigger Clock Source*. Once a valid UART message is received by the sensor, the seconds timestamp will snap to the latest timestamp received. The expected NMEA UART message is configurable as described in TCP API Guide. For example, the below commands will set the sensor to accept an NMEA UART message that is active high with a baud rate of 115200 bits per second, add 27 additional leap seconds, and accept messages even with a valid character not set:

- `set_config_param multipurpose_io_mode INPUT_NMEA_UART`
- `set_config_param nmea_in_polarity ACTIVE_HIGH`
- `set_config_param nmea_baud_rate BAUD_115200`
- `set_config_param nmea_leap_seconds 27`

- `set_config_param nmea_ignore_valid_char 1`
- `reinitialize`

9.2.2 External Trigger Clock Source

Additionally, the sensor can be configured to output a SYNC_PULSE_OUT signal from a variety of sources. See example commands in the *TCP API Guide* section. Pulses will always be evenly spaced.

This can be enabled through the `multipurpose_io_mode` configuration parameter.

Configuration	**Response**
`OFF`	Do not output a SYNC_PULSE_OUT signal.
`INPUT_NMEA_UART`	Reconfigures the MULTIPURPOSE_IO port as an input. See *Setting Ouster Sensor Time Source* for more information.
`OUTPUT_FROM_INTERNAL_OSC`	Output a SYNC_PULSE_OUT signal synchronized with the internal clock.
`OUTPUT_FROM_SYNC_PULSE_IN`	Output a SYNC_PULSE_OUT signal synchronized with a SYNC_PULSE_IN provided to the unit.
`OUTPUT_FROM_PTP_1588`	Output a SYNC_PULSE_OUT signal synchronized with an external PTP IEEE 1588 master.
`OUTPUT_FROM_ENCODER_ANGLE`	Output a SYNC_PULSE_OUT signal with a user defined rate in an integer number of degrees.

When the sensor's `multipurpose_io_mode` is set to `OUTPUT_FROM_INTERNAL_OSC`, `OUTPUT_FROM_SYNC_PULSE_IN`, or `OUTPUT_FROM_PTP_1588`, then `sync_pulse_out_frequency` (Hz) parameter can be used to define the output rate. It defaults to 1 Hz. It should be greater than 0 Hz and maximum `sync_pulse_out_frequency` is limited by the criterion below.

When the sensor is set to `OUTPUT_FROM_ENCODER_ANGLE`, then the `sync_pulse_out_angle` (deg) parameter can be used to define the output pulse rate. This allows the user to output a SYNC_PULSE_OUT signal when the encoder passes a specified angle, or multiple of the angle, indexed from 0 crossing, in degrees. It should be an integer between 0 and 360 degrees, inclusive. However, the minimum `sync_pulse_out_angle` is also limited by the criterion below.

In all modes, the output pulse width is defined by `sync_pulse_out_pulse_width` (ms).

Note: If `sync_pulse_out_pulse_width` x `sync_pulse_out_frequency` is close to 1 second, the output pulses will not function (will not return to 0). For example, at 10 Hz rotation and a 10 ms pulse width, the limitation on the number of pulses per rotation is 9.

Example Commands

Here are example commands and their effect on output pulse when `lidar_mode` is `1024x10`, and assuming `sync_pulse_out_pulse_width` is 10 ms.

Command	Response

`set_config_param multipurpose_io_mode OUTPUT_FROM_SYNC_PULSE_IN`
`set_config_param sync_pulse_out_pulse_width 10`
`set_config_param sync_pulse_out_frequency 1`
`reinitialize`
	The output pulse frequency is 1 Hz. Each pulse is 10 ms wide. `sync_pulse_out_pulse_width` and `sync_pulse_out_frequency` commands are optional because they just re-command the default values

`set_config_param multipurpose_io_mode OUTPUT_FROM_SYNC_PULSE_IN`
`set_config_param sync_pulse_out_frequency 50`
`reinitialize`
	The output pulse frequency is 50 Hz. Each pulse is 10 ms wide.

`set_config_param multipurpose_io_mode OUTPUT_FROM_ENCODER_ANGLE`
`set_config_param sync_pulse_out_angle 360`
`reinitialize`
	The output pulse frequency is 10 Hz, since the sensor is in 10 Hz mode (10 rotations / sec) and the angle is set to 360º, a full rotation. Each pulse is 10 ms wide.

`set_config_param multipurpose_io_mode OUTPUT_FROM_ENCODER_ANGLE`
`set_config_param sync_pulse_out_angle 45`
`reinitialize`
	The output pulse frequency is 80 Hz, since the sensor is in 10 Hz mode (10 rotations / sec) and the angle is set to 45º. Each full rotation will have 8 pulses. Each pulse is 10 ms wide.

9.3 NMEA Message Format

The Ouster Sensor expects a standard NMEA $GPRMC UART message. Data (called a sentence) is a simple ASCII string starting with a '$' character and ending with a return character. Fields of the sentence are separated with a ',' character, and the last field (a checksum) is separated by a '*' character.

The max character length of a standard message is 80 characters; however, the Ouster Sensor can support non-standard messages up to 85 characters (see Example 2 below).

The Ouster Sensor will deliver time in the UDP packet by calculating seconds since 00:00:00 Thursday, 1 January 1970. `nmea_leap_seconds` by default is 0, meaning this calculation will not take into account any leap seconds. If `nmea_leap_seconds` is 0 then the reported time is Unix Epoch time. As of February, 2019 Coordinated Universal Time (UTC) lags behind International Atomic Time (TAI) by an offset of 37 seconds (10 seconds from the initial UTC offset when UTC was introduced in 1972 + 27 leap seconds announced in the intervening years). Therefore, setting `nmea_leap_seconds` to 37 in February of 2019 would make the timestamps match the TAI standard.

`nmea_in_polarity` by default is `ACTIVE_HIGH`. This means that a UART start bit will occur directly after a falling edge. If using RS-232, the UART signal may be inverted (where a start bit occurs directly after a rising edge). In this case, `nmea_in_polarity` should be set to `ACTIVE_LOW`.

9.3.1 Example 1 Message:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
```

Field	Description
$GPRMC	Recommended Minimum sentence C
123519	Fix taken at 12:35:19 UTC
A	Status A=active or V=Void
4807.038	Latitude 48 deg 07.038′
N	Latitude cardinal reference
01131.000	Longitude 11 deg 31.000′
E	Longitude cardinal reference
022.4	Speed over the ground in knots
084.4	Track angle in degrees True
230394	Date - 23rd of March 1994
003.1	Magnetic Variation
W	Magnetic cardinal reference
A	[Optional] A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator
*6A	The checksum data, always begins with *

9.3.2 Example 2 Message:

```
$GPRMC,042901.00,A,3745.871698,N,12224.825960,W,0.874,327.72,130219,13.39,E,A,*60
```

Field	Description
$GPRMC	Recommended Minimum sentence C
042901.00	Fix taken at 4:29:01 UTC
A	Status A=active or V=Void
3745.871698	Latitude 37 deg 45.871698′
N	Latitude cardinal reference
12224.825960	Longitude 12 deg 24.825960′
W	Longitude cardinal reference
0.874	Speed over the ground in knots
327.72	Track angle in degrees True
130219	Date - 13th of February 2019
13.39	Magnetic Variation
E	Magnetic cardinal reference
A	[Optional] A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator
*60	The checksum data, always begins with *

10 GPS/GNSS Synchronization Guide

For more information on how to physically connect a GPS to your Ouster sensor and synchronise the Ouster sensor timestamp to an NMEA sentence, please refer to your sensor's Hardware User Manual.

10.1 Configuring the Ouster Sensor

Now that everything is configured and verified on the GPS side and you have connected everything to the Ouster sensor, it is time to configure the Ouster sensor to synchronize its timestamp with the GPS.

- Set the `timestamp_mode` to `TIME_FROM_SYNC_PULSE_IN`
  - TCP command: `set_config_param timestamp_mode TIME_FROM_SYNC_PULSE_IN`
- Set the `multipurpose_io_mode` to `INPUT_NMEA_UART`
  - TCP command: `set_config_param multipurpose_io_mode INPUT_NMEA_UART`
- Set the polarity of the `sync_pulse_in` pin to match the GPS PPS polarity
  - TCP command: `set_config_param sync_pulse_in_polarity <ACTIVE_HIGH or ACTIVE_LOW>`
- Set the polarity of the `multipurpose_io` pin to match the GPS NMEA UART polarity
  - TCP command: `set_config_param nmea_in_polarity <ACTIVE_HIGH or ACTIVE_LOW>`
- Set the `nmea_baud_rate` to match the GPS NMEA baud rate
  - TCP command: `set_config_param nmea_baud_rate <BAUD_11520 or BAUD_9600>`
- Set the `nmea_leap_seconds` to match the current leap seconds as defined by TIA at this website, at time of writing this the leap seconds are `37`
  - TCP command: `set_config_param nmea_leap_seconds 37`
- Reinitialize and write the configuration
  - TCP command: `reinitialize`
  - TCP command: `save_config_params`

10.1.1 Checking for Sync

Once you have completed all the above you should be able to check for synchronization

- Check the output from the TCP command: `get_time_info`
  - Verify that the sensor is `locked` onto the PPS signal.
    - "sync_pulse_in": { "locked": 1 }
    - if not check the polarity and change it if necessary.
  - Verify that the sensor is `locked` on the NMEA signal.
    - "nmea": { "locked": 1 }

        - if not check the polarity and baud rate and change them if necessary.
- Verify that `last_read_message` looks like a valid GPRMC sentence.
    - "decoding": {"last_read_message": "GPRMC,024041.00,A,5107.0017737,N,11402.3291611, W,0.080,323.3,020420,0.0,E,A*20"}
- Verify that `timestamp` time has updated to a reasonable GPS time.
    - "timestamp": {"time": 1585881641.96139565999999, "mode": "TIME_FROM_SYNC_PUSLE_IN", "time_options": { "sync_pulse_in": 1585881641}}

Example output from get_time_info:

```
{
   "timestamp":{
      "time":1585881641.96139565999999,
      "mode":"TIME_FROM_SYNC_PUSLE_IN",
      "time_options":{
         "sync_pulse_in":1585881641,
         "internal_osc":302,
         "ptp_1588":309
      }
   },
   "sync_pulse_in":{
      "locked":1,
      "diagnostics":{
         "last_period_nsec":10,
         "count_unfiltered":832,
         "count":832
      },
      "polarity":"ACTIVE_HIGH"
   },
   "multipurpose_io":{
      "mode":"INPUT_NMEA_UART",
      "sync_pulse_out":{
         "pulse_width_ms":10,
         "angle_deg":360,
         "frequency_hz":1,
         "polarity":"ACTIVE_HIGH"
      },
      "nmea":{
         "locked":1,
         "baud_rate":"BAUD_9600",
         "diagnostics":{
            "io_checks":{
               "bit_count":2938457,
               "bit_count_unfilterd":2938457,
               "start_char_count":832,
               "char_count":66526
            },
            "decoding":{
               "last_read_message":"GPRMC,024041.00,A,5107.0017737,N,11402.3291611,W,
                                   0.080,323.3,020420,0.0,E,A*20",
               "date_decoded_count":832,
               "not_valid_count":0,
```

(continues on next page)

(continued from previous page)

```
                "utc_decoded_count":832
            }
        },
        "leap_seconds":37,
        "ignore_valid_char":0,
        "polarity":"ACTIVE_HIGH"
      }
   }
}
```

11 Sensor Configuration

11.1 Configurable Parameters

Please find a list of all **config_param** that can be used to configure the sensor below. For detailed description of each parameter refer to *Description- Configurable Parameters*.

Table11.1: Overview

Parameter	Type	Valid Values
`udp_dest`	String	`""` (default)
`udp_port_lidar`	Integer	`7502` (default)
`udp_port_imu`	Integer	`7503` (default)
`sync_pulse_in_polarity`	Keyword	`ACTIVE_HIGH` (default) `ACTIVE_LOW`
`sync_pulse_out_polarity`	Keyword	`ACTIVE_LOW` (default) `ACTIVE_HIGH`
`sync_pulse_out_frequency`	Integer >= 1	`1` (default)
`sync_pulse_out_angle`	Integer [0 ... 360]	`360` (default)
`sync_pulse_out_pulse_width`	Integer >= 0	`10` (default)
`nmea_in_polarity`	Keyword	`ACTIVE_HIGH` (default) `ACTIVE_LOW`
`nmea_ignore_valid_char`	integer [0 ... 1]	`0` (default) `1`
`nmea_baud_rate`	Keyword	`BAUD_9600` (default) `BAUD_115200`
`nmea_leap_seconds`	Integer >= 0	`0` (default)
`azimuth_window`	List	`[0,360000]` (default)

`signal_multiplier`	number [1 ... 3]	`1` (default)
		`2`
		`3`

`udp_profile_lidar`	Keyword	`LEGACY` (default)
		`RNG19_RFL8_SIG16_NIR16`
		`RNG19_RFL8_SIG16_NIR16_DUAL`
		`RNG15_RFL8_NIR8`

`udp_profile_imu`	Keyword	`LEGACY` (default)
`phase_lock_enable`	Boolean	`false` (default) `true`

continues on next page

Table 11.1 - continued from previous page

Parameter	Type	Valid Values
`phase_lock_offset`	Integer [0 ... 360]	`0` (default)

`lidar_mode`	Keyword	`512x10`
		`1024x10` (default)
		`2048x10`
		`512x20`
		`1024x20`

`timestamp_mode`	Keyword	`TIME_FROM_INTERNAL_OSC` (default)
		`TIME_FROM_PTP_1588`
		`TIME_FROM_SYNC_PULSE_IN`

`multipurpose_io_mode`	Keyword	`OFF` (default)
		`INPUT_NMEA_UART`
		`OUTPUT_FROM_INTERNAL_OSC`
		`OUTPUT_FROM_SYNC_PULSE_IN`
		`OUTPUT_FROM_PTP_1588`
		`OUTPUT_FROM_ENCODER_ANGLE`

`operating_mode`	Keyword	`NORMAL` (default) `STANDBY`

11.2 Description- Configurable Parameters

11.2.1 `udp_dest`

Description:

- Type: String
- Default: "169.254.198.184"
- Destination to which the sensor sends UDP traffic.

Note: As of now, setting the udp_dest to "@auto" is not supported through the set_config_param TCP command. It is only supported through `HTTP endpoint` **POST** `/api/v1/sensor/config`, and the `set_udp_dest_auto` using TCP command. `udp_ip` flag has been **deprecated** in **Firmware 2.4**, please use `udp_dest` flag.

11.2.2 `udp_port_lidar`

Description:

- Type: Integer [0 ... 65535]
- Default: 7502
- The `<port>` on `udp_dest` to which lidar data will be sent (`7502`, default).

11.2.3 `udp_port_imu`

Description:

- Type: Integer [0 ... 65535]
- Default: 7503
- The `<port>` on `udp_dest` to which IMU data will be sent (`7503`, default).

11.2.4 `sync_pulse_in_polarity`

Description:

- Type: Keyword
- Default: "ACTIVE_HIGH"
- **Enum:**
    - "ACTIVE_HIGH"
    - "ACTIVE_LOW"
- The polarity of SYNC_PULSE_IN input, which controls polarity of SYNC_PULSE_IN pin when timestamp_mode is set in `TIME_FROM_SYNC_PULSE_IN`.

11.2.5 `sync_pulse_out_polarity`

Description:

- Type: Keyword
- Default: "ACTIVE_HIGH"
- The polarity of SYNC_PULSE_OUT output, if the sensor is set as the master sensor used for time synchronization.

11.2.6 `sync_pulse_out_frequency`

Description:

- Type: Integer >= 1
- Default: 1
- The output SYNC_PULSE_OUT pulse rate in Hz. Valid inputs are integers >0 Hz, but also limited by the criteria described in the Time Synchronization section of the Software User Manual.

11.2.7 `sync_pulse_out_angle`

Description:

- Type: Integer [0 ... 360]
- Default: 360
- The angle in terms of degrees that the sensor traverses between each SYNC_PULSE_OUT pulse. E.g. a value of 180 means a sync pulse is sent out every 180º for a total of two pulses per revolution and angular frequency of 20 Hz if the sensor is 1024x10 Hz lidar mode. Valid inputs are integers between 0 and 360 inclusive but also limited by the criteria described in the Time Synchronization section of Software User Manual.

11.2.8 `sync_pulse_out_pulse_width`

Description:

- Type: Integer >= 0
- Default: 10
- The polarity of SYNC_PULSE_OUT output, if the sensor is set as the master sensor used for time synchronization. Output SYNC_PULSE_OUT pulse width is in ms, increments in 1 ms. Valid inputs are integers greater than 0 ms, but also limited by the criteria described in the Time Synchronization section of Software User Manual.

11.2.9 `nmea_in_polarity`

Description:

- Type: Keyword
- Default: "ACTIVE_HIGH"
- **Enum:**
  - "ACTIVE_HIGH"
  - "ACTIVE_LOW"
- Set the polarity of NMEA UART input $GPRMC messages. See Time Synchronization section in sensor user manual for NMEA use case. Use `ACTIVE_HIGH` if UART is active high, idle low, and start bit is after a falling edge.

11.2.10 `nmea_ignore_valid_char`

Description:

- Type: Integer [0 ... 1]
- Default: 0
- Set `0` if NMEA UART input $GPRMC messages should be ignored if valid character is not set, and `1` if messages should be used for time syncing regardless of the valid character.

11.2.11 `nmea_baud_rate`

Description:

- Type: Keyword
- Default: "BAUD_9600"
- **Enum:**
    - "BAUD_9600"
    - "BAUD_115200"
- `BAUD_9600` (default) or `BAUD_115200` for the expected baud rate the sensor is attempting to decode for NMEA UART input $GPRMC messages.

11.2.12 `nmea_leap_seconds`

Description:

- Type: Integer >= 0
- Default: 0
- Set an integer number of leap seconds that will be added to the UDP timestamp when calculating seconds since 00:00:00 Thursday, 1 January 1970. For Unix Epoch time, this should be set to `0`.

11.2.13 `azimuth_window`

Description:

- Type: List
- Default: [0,360000]
- Set the visible region of interest of the sensor in millidegrees. Only data from within the specified azimuth window bounds is sent. The value should be provisioned as: [min_bound_millideg, max_bound_millideg]

11.2.14 `signal_multiplier`

Description:

- Type: Number [1 ... 3]
- Default: 1
- The value that the signal_multiplier is configured. By default the sensor has a signal multiplier value of `1`.

  For 2x and 3x multipliers, the `azimuth_window` parameter sets the azimuth window that the lasers will be enabled in.

  The higher the signal multiplier value, the smaller the maximum azimuth window can be.

  Signal Multiplier Value Max Azimuth Window 1: (Default) 360º, 2: 180º, 3: 120º.

  All sensors have equivalent power draw and thermal output when operating at the max azimuth window for a particular signal multiplier value. Therefore, using an azimuth window that is smaller than the maximum allowable azimuth window with a particular signal multiplier value (excluding 1x) can reduce the power draw and thermal output of the sensor.

  However, while this can increase the max operating temp of the sensor, it can also degrade the performance at low temps. This discrepancy will be resolved in a future firmware. The table below outlines some example use cases.

11.2.15 `udp_profile_lidar`

Description:

- Type: Keyword
- Default: "LEGACY"
- **Enum:**
  - "LEGACY"
  - "RNG19_RFL8_SIG16_NIR16"
  - "RNG19_RFL8_SIG16_NIR16_DUAL"
  - "RNG15_RFL8_NIR8"
- The configuration of the LIDAR data packets. Valid values are `LEGACY` *[Default]*, `RNG19_RFL8_SIG16_NIR16`, `RNG19_RFL8_SIG16_NIR16_DUAL`, `RNG15_RFL8_NIR8`.

11.2.16 `udp_profile_imu`

Description:

- Type: Keyword
- Default: "LEGACY"
- Value: "LEGACY"
- The configuration of the IMU data packets. Valid value is `LEGACY`.

11.2.17 `phase_lock_enable`

Description: Whether phase locking is enabled. Refer to Phase Lock Section in the Firmware User Manual for more details on using phase lock.

- Type: Boolean
- Default: False
- Whether phase locking is enabled. Refer to Phase Lock Section in the Firmware User Manual for more details on using phase lock.

11.2.18 `phase_lock_offset`

Description:

- Type: Integer [0 ... 360000]
- Default: 0
- The angle in the Lidar Coordinate Frame that sensors are locked to in millidegrees if phase locking is enabled. Angle is traversed at the top of the second.

11.2.19 `lidar_mode`

Description:

- Type: Keyword
- Default: "1024x10"
- **Enum:**
  - "512x10"
  - "1024x10"
  - "2048x10"
  - "512x20"
  - "1024x20"
- The horizontal resolution and rotation rate of the sensor. The effective range of the sensor is increased by 15-20% for every halving of the number of points gathered e.g. 512x10 has 15-20% longer range than 512x20.

11.2.20 `timestamp_mode`

Description:

The method used to timestamp measurements. Valid modes are `TIME_FROM_INTERNAL_OSC`, `TIME_FROM_SYNC_PULSE_IN`, or `TIME_FROM_PTP_1588`.

11.2.21 `multipurpose_io_mode`

Description:

- Type: Keyword
- Default: "OFF"
- **Enum:**
    - "OFF"
    - "INPUT_NMEA_UART"
    - "OUTPUT_FROM_INTERNAL_OSC"
    - "OUTPUT_FROM_SYNC_PULSE_IN"
    - "OUTPUT_FROM_PTP_1588"
    - "OUTPUT_FROM_ENCODER_ANGLE"
- Configure the mode of the MULTIPURPOSE_IO pin. Refer to Time Synchronization section in Firmware user manual for a detailed description of each option.

11.2.22 `operating_mode`

Description:

- Type: Any
- Default: "NORMAL"
- Set `NORMAL` to put the sensor into a normal operating mode or `STANDBY` to put the sensor into a low power (5W) operating mode where the motor does not spin and lasers do not fire.

Note: `auto_start_flag` is deprecated parameter in Firmware 2.4 and later. `auto_start_flag 0` is equivalent to `operating_mode STANDBY` and `auto_start_flag 1` is equivalent to `operating_mode NORMAL`.

---

12 TCP API Guide

12.1 Querying Sensor Info and Intrinsic Calibration

The sensor can be queried and configured using a simple plaintext protocol over TCP on port 7501.

An example session using the unix netcat utility is shown below. Note: "xxx" refers to the sensor serial number. The hostname of the sensor can look like "os-xxx" or "os1-xxx".

```
$ nc os-122201000998.local 7501
get_sensor_info

 {"prod_line": "OS-1-128", "prod_pn": "840-103575-06", "prod_sn": "122201000998",
  "image_rev": "ousteros-image-prod-aries-v2.4.0-omega.2+20220730010801.staging",
  "build_rev": "v2.4.0-omega.2", "build_date": "2022-07-29T23:56:15Z",
  "status": "RUNNING", "initialization_id": 9599937}
```

A sensor may have one of the following statuses:

Table 12.1: Sensor Status

Status	Description
`INITIALIZING`	When the sensor is booting and not yet outputting data
`WARMUP`	Sensor has gone into thermal warmup state
`UPDATING`	When the sensor is updating the FPGA firmware on the first reboot after a firmware upgrade
`RUNNING`	When the sensor has reached the final running state where it can output data
`STANDBY`	The sensor has been configured into a low-power state where sensor is on but not spinning
`ERROR`	Check error codes in the `errors` field for more information
`UNCONFIGURED`	An error with factory calibration that requires a manual power cycle or reboot

Note: If the sensor is set to **STANDBY** mode some of these commands will not return the expected values.

If the sensor is in an `ERROR` or `UNCONFIGURED` state, please contact Ouster support with the diagnostic file found at http://os-9919xxxxxxxx/diag for support.

12.2 Sensor Configuration Parameter Interface

12.2.1 `get_config_param`

Description: Returns all active or staged JSON-formatted sensor configuration.

Sensor configurations and operating modes can also be queried over TCP. Below is the command format:

- `get_config_param active <parameter>` will return the current active configuration parameter values.
- `get_config_param staged <parameter>` will return the parameter values that will take place after issuing a `reinitialize` command.

Warning: The command `get_config_txt` is deprecated and superseded by `get_config_param active`, which provides the same response. `get_config_txt` will be removed in a future firmware.

Example 1: Shows how a user can `get` information of a particular parameter below using the unix netcat utility:

```
$ nc os-991900123456 7501
get_config_param active lidar_mode
1024x10
```

Example 2: Shows how a user can `get` all the `<active> parameters` information below using the unix netcat utility:

```
$ nc os-991900123456 7501
  get_config_param active
  {
    "udp_dest": "169.254.111.53",
    "udp_port_lidar": 7502,
    "udp_port_imu": 7503,
    "udp_profile_lidar": "RNG19_RFL8_SIG16_NIR16_DUAL",
    "udp_profile_imu": "LEGACY",
    "columns_per_packet": 16,
    "timestamp_mode": "TIME_FROM_INTERNAL_OSC",
    "sync_pulse_in_polarity": "ACTIVE_HIGH",
    "nmea_in_polarity": "ACTIVE_HIGH",
    "nmea_ignore_valid_char": 0,
    "nmea_baud_rate": "BAUD_9600",
    "nmea_leap_seconds": 0,
    "multipurpose_io_mode": "OFF",
    "sync_pulse_out_polarity": "ACTIVE_HIGH",
    "sync_pulse_out_frequency": 1,
    "sync_pulse_out_angle": 360,
    "sync_pulse_out_pulse_width": 10,
    "operating_mode": "NORMAL",
    "lidar_mode": "1024x10",
    "azimuth_window": [0, 360000],
    "signal_multiplier": 1,
```

(continues on next page)

(continued from previous page)

```
    "phase_lock_enable": false,
    "phase_lock_offset": 0
  }
```

12.2.2 `set_config_param`

`set_config_param <parameter> <value>` will set new values for configuration parameters, which will take effect after issuing the `reinitialize` command.

`reinitialize` will reinitialize the sensor so the staged values of the parameters will take effect immediately.

`save_config_params` will write new values of active parameters into a configuration file, so they will persist after sensor reset. In order to permanently change a parameter in the configuration file, first use `set_config_param` to update the parameter in a staging area, then use `reinitialize` to make that parameter active. Only after the parameter is made active will `save_config_params` capture it to persist after reset.

Warning: The command `write_config_txt` will be deprecated in a future firmware. The command `save_config_params` provides the same response.

While in **STANDBY** mode, we can set the config parameters, but it will not take effect until we switch the sensor back to **NORMAL** mode.

`set_udp_dest_auto` will automatically determine the sender's IP address at the time the command was sent, and set it as the destination of UDP traffic. This takes effect after issuing a `reinitialize` command. Using this command has the same effect as using `set_config_param udp_dest <ip address>`.

An example session using the unix netcat utility is shown below.

Note: In the example below, to distinguish between the command and expected response, a dash has been added before the expected response. The actual response will be without the dash.

```
$ nc os-991900123456.local 7501
set_config_param lidar_mode 512x20
-set_config_param
set_udp_dest_auto
-set_udp_dest_auto
reinitialize
-reinitialize
save_config_params
-save_config_params
```

Note: In the example below, see the `error` message when an invalid value is set i.e., `lidar_mode` = **511x10**.

```
$ nc os-122201000998.local 7501
set_config_param lidar_mode 511x10
error: '511x10' is not supported
```

The following commands will set sensor configuration parameters:

Note: Each of the following commands have two responses: * `set_config_param` on Success * `error:` Otherwise

Table12.2: Reinitialize, Save Sensor Config, and Auto Setting of Destination IP

Command	Command Description
`reinitialize` or `reinit`	Restarts the sensor. Changes to lidar, multipurpose_io, and timestamp modes will only take effect after reinitialization. **Response on success:** `reinitialize` or `reinit`
`save_config_params`	Makes all current parameter settings persist after reboot. **Response on success:** `save_config_params`
`set_udp_dest_auto`	Set the destination of UDP traffic to the destination address that issued the command. **Response on success:** `set_udp_dest_auto`

Note: Refer to *Sensor Configuration* for detailed information on all configurable parameters using TCP commands.

12.3 Sensor Status and Calibration

12.3.1 `get_sensor_info`

Description: Returns JSON-formatted sensor metadata: serial number, hardware and software revision, and sensor status.

```
{
   "prod_line": "OS-1-128",
   "prod_pn": "840-103575-06",
   "prod_sn": "122201000998",
   "image_rev": "ousteros-image-prod-aries-v2.4.0-omega.2+20220730010801.staging",
   "build_rev": "v2.4.0-omega.2",
   "build_date": "2022-07-29T23:56:15Z",
   "status": "RUNNING",
   "initialization_id": 9599937
}
```

12.3.2 `get_time_info`

Description: Returns JSON-formatted sensor timing configuration and status of udp `timestamp`, `sync_pulse_in`, and `multipurpose_io`.

```
{
   "timestamp":
     {
       "time": 3709.04727264,
       "mode": "TIME_FROM_INTERNAL_OSC",
      "time_options":
         {
            "ptp_1588": 3718,
            "sync_pulse_in": 1,
            "internal_osc": 3709
         }
     },
   "sync_pulse_in":
     {
       "locked": 0,
       "polarity": "ACTIVE_HIGH",
       "diagnostics":
         {
            "last_period_nsec": 0,
            "count": 1,
            "count_unfiltered": 0
         }
     },
   "multipurpose_io":
     {
       "mode": "OFF",
       "sync_pulse_out":
        {
          "polarity": "ACTIVE_HIGH",
          "frequency_hz": 1,
```

(continues on next page)

(continued from previous page)

```
          "angle_deg": 360,
          "pulse_width_ms": 10
        },
      "nmea":
        {
          "locked": 0,
          "polarity": "ACTIVE_HIGH",
          "ignore_valid_char": 0,
          "baud_rate": "BAUD_9600",
          "leap_seconds": 0,
          "diagnostics":
           {
             "decoding":
               {
                 "utc_decoded_count": 0,
                 "date_decoded_count": 0,
                 "not_valid_count": 0,
                 "last_read_message": ""
               },
             "io_checks":
               {
                 "start_char_count": 0,
                 "char_count": 0,
                 "bit_count": 1,
                 "bit_count_unfiltered": 0
               }
           }
        }
     }
}
```

12.3.3 get_beam_intrinsics

Description: Returns JSON-formatted beam altitude and azimuth offsets, in degrees. Length of arrays is equal to the number of channels in the sensor. Also returns distance between lidar origin and beam origin in mm, to be used for point cloud calculations.

```
{
  "beam_altitude_angles": [21.01, 20.72, 20.42,
    20.1, 19.79, 19.48, 19.17,
    18.84, 18.55, 18.22, 17.9, 17.6,
    17.27, 16.96, 16.65, 16.32, 15.97,
    15.65, 15.34, 15.01, 14.67, 14.35,
    14.01, 13.68, 13.33, 13.02, 12.67,
    12.34, 11.99, 11.65, 11.33, 10.98,
    10.64, 10.28, 9.949999999999999,
    9.609999999999999, 9.27, 8.92, 8.57,
    8.23, 7.88, 7.54, 7.18, 6.84, 6.47,
    6.13, 5.78, 5.45, 5.09, 4.73,
    4.41, 4.05, 3.69, 3.32, 2.98,
    2.63, 2.27, 1.93, 1.57, 1.22,
    0.85, 0.5, 0.15, -0.19, -0.55, -0.92,
```

(continues on next page)

(continued from previous page)

```
  -1.25, -1.63, -1.98, -2.31, -2.67,
  -3.04, -3.4, -3.74, -4.09, -4.45, -4.8,
  -5.15, -5.5, -5.87, -6.21, -6.57, -6.91,
  -7.25, -7.62, -7.95, -8.300000000000001,
  -8.65, -9.01, -9.35, -9.69, -10.05,
  -10.39, -10.74, -11.09, -11.42, -11.77,
  -12.11, -12.45, -12.8, -13.14, -13.47,
  -13.81, -14.15, -14.48, -14.82, -15.13,
  -15.47, -15.81, -16.15, -16.48, -16.8,
  -17.15, -17.48, -17.79, -18.12, -18.47,
  -18.77, -19.09, -19.42, -19.73, -20.06,
  -20.36, -20.69, -21, -21.32, -21.62, -21.94],
 "beam_azimuth_angles": [4.23, 1.41, -1.4,
  -4.21, 4.22, 1.42, -1.41, -4.22, 4.23,
   1.41, -1.42, -4.21, 4.23, 1.42, -1.4,
  -4.2, 4.23, 1.41, -1.39, -4.21, 4.25,
   1.43, -1.41,-4.22, 4.24, 1.44, -1.41,
  -4.22, 4.23, 1.42, -1.38, -4.22, 4.24,
   1.42, -1.4, -4.23, 4.26, 1.44, -1.41,
  -4.23, 4.24, 1.44, -1.41, -4.23, 4.24,
   1.42, -1.41, -4.24, 4.25, 1.42, -1.39,
  -4.22, 4.25, 1.41, -1.4, -4.23, 4.24,
   1.43, -1.41, -4.23, 4.24, 1.42, -1.41,
  -4.23, 4.25, 1.42, -1.4, -4.24, 4.24,
   1.44, -1.4, -4.24, 4.24, 1.43, -1.4, -4.24,
   4.25, 1.43, -1.41, -4.24, 4.25, 1.42, -1.42,
  -4.22, 4.24, 1.43, -1.4, -4.24, 4.25, 1.44,
  -1.41, -4.24, 4.25, 1.42, -1.41, -4.24,
   4.25, 1.42, -1.41, -4.25, 4.26, 1.43, -1.41,
  -4.26, 4.27, 1.43, -1.4, -4.24, 4.25, 1.43,
  -1.4, -4.24, 4.26, 1.42, -1.4, -4.25, 4.26,
   1.43, -1.41, -4.26, 4.26, 1.42, -1.42,
   -4.26, 4.25, 1.42, -1.42, -4.27],
 "lidar_origin_to_beam_origin_mm": 15.806
}
```

12.3.4 `get_imu_intrinsics`

Description: Returns JSON-formatted IMU transformation matrix needed to transform to the Sensor Coordinate Frame.

```
{
  "imu_to_sensor_transform":
    [
     1, 0, 0, 6.253, 0, 1, 0, -11.775,
     0, 0, 1, 7.645, 0, 0, 0, 1
    ]
}
```

12.3.5 `get_lidar_intrinsics`

Description: Returns JSON-formatted lidar transformation matrix needed to transform to the Sensor Coordinate Frame.

```
{
  "lidar_to_sensor_transform":
  [
    -1, 0, 0, 0, 0, -1, 0, 0, 0, 0,
     1, 36.18, 0, 0, 0, 1
  ]
}
```

12.3.6 `get_alerts`

Example 1: `get_alerts <START_CURSOR><MODE>`

Description: Returns JSON-formatted sensor diagnostic information.

Two lists will be returned, an **active** list and a **log** list. The **active** list will contain alert-trigger events for alerts that are currently active. An alert-trigger event will by-definition always have its active attribute set to true. There is no limit on the number of alert-trigger events that are displayed in the **active** event list. All currently active alert-trigger events will be displayed in the **active** event list.

The **log** list will contain all current and past alert-trigger and alert-clear events. An alert-clear event will by-definition have the exact same attributes and attribute values as its corresponding trigger event, with the exception of the realtime and cursor attributes which should have higher values, since an alert-clear event will always be received after an alert-trigger event. The **log** list has a length limit of 32 events with the oldest events automatically removed from the log list once a new event needs to be added to the **log** list and the **log** list length limit is reached, essentially acting as a FIFO (First In First Out) queue.

In addition to the **active** and **log** lists, `get_alerts` also returns a **next_cursor** field. Every alert event has a cursor attribute, which increments for every alert event logged. This can be used to track the alert activity that has been viewed and reduce message bandwidth. To do this, users are recommended to save the **next_cursor** field when calling `get_alerts` and then use that value as the `START_CURSOR` argument on the next `get_alerts` call to fetch only new **log** entries.

A valid value for **MODE** is either `summary` or `default`.

Note: Valid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=1

  `get_alerts 1`

- Example: Calling get_alerts with START_CURSOR=2 and MODE=summary

  `get_alerts 2 summary`

Invalid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=summary and MODE=2

  `get_alerts summary 2`

Note: The position of `START_CURSOR` and `MODE` is important to follow when using `TCP Commands` and is irrelevant when calling the equivalent **/api/v1/sensor/alerts** `HTTP endpoint`, meaning for an HTTP GET call the position is insensitive.

The example in the code block is for `get_alerts`.

```
{
   "log":
  [
    {
        "active": true,
        "category": "UDP_TRANSMISSION",
        "cursor": 0,
        "id": "0x01000018",
        "level": "WARNING",
        "msg": "Client machine announced
            it is not reachable on the
            provided not reachable on
            IMU data port; check that
            udp_dest and udp_port_imu
            configured on the sensor matches
            client IP and port.",
        "msg_verbose": "Failed to send
            imu UDP data to destination
            host 169.254.175.254:7503",
        "realtime": "39850161524"
    },
    {
        "active": true,
        "category": "UDP_TRANSMISSION",
        "cursor": 1,
        "id": "0x01000015",
        "level": "WARNING",
        "msg": "Client machine announced
            it is not reachable on the
            provided lidar data port;
```

(continues on next page)

(continued from previous page)

```
            check that udp_dest and
            udp_port_lidar configured on
            the sensor matches client IP
            and port.",
        "msg_verbose": "Failed to send
            lidar UDP data to destination
            host 169.254.175.254:7502",
        "realtime": "40842065146"
    },
    {
        "active": true,
        "category": "ETHERNET_LINK_BAD",
        "cursor": 2,
        "id": "0x01000011",
        "level": "WARNING",
        "msg": "Ethernet link bad, please
            check network switch and
            harnessing can support
            1 Gbps Ethernet.",
        "msg_verbose": "Link transitioned
            to 0/Unknown",
        "realtime": "414257307390"
    },
    {
        "active": true,
        "category": "ETHERNET_LINK_BAD",
        "cursor": 2,
        "id": "0x01000011",
        "level": "WARNING",
        "msg": "Ethernet link bad, please
            check network switch and
            harnessing can support
            1 Gbps Ethernet.",
        "msg_verbose": "Link transitioned
            to 0/Unknown",
        "realtime": "414257307390"
    },
    {
        "active": true,
        "category": "UDP_TRANSMISSION",
        "cursor": 3,
        "id": "0x01000016",
        "level": "WARNING",
        "msg": "Could not send lidar data
            UDP packet to host; check that
            network is up.",
        "msg_verbose": "Failed to send
            lidar UDP data to destination
            host 169.254.175.254:7502",
        "realtime": "414261086316"
    },
    {
        "active": true,
        "category": "UDP_TRANSMISSION",
```

(continues on next page)

(continued from previous page)

```
        "cursor": 4,
        "id": "0x01000019",
        "level": "WARNING",
        "msg": "Could not send IMU UDP
            packet to host; check
            that network is up.",
        "msg_verbose": "Failed to send imu
            UDP data to destination
            host 169.254.175.254:7503",
        "realtime": "414266339945"
    },
    {
        "active": false,
        "category": "ETHERNET_LINK_BAD",
        "cursor": 5,
        "id": "0x01000011",
        "level": "WARNING",
        "msg": "Ethernet link bad,
            please check network switch
            and harnessing can support
            1 Gbps Ethernet.",
        "msg_verbose": "Link transitioned
            to 1000/Full",
        "realtime": "416337486469"
    }
  ],
  "next_cursor": 6,
  "active":
    [
      {
        "active": true,
        "category": "UDP_TRANSMISSION",
        "cursor": 1,
        "id": "0x01000015",
        "level": "WARNING",
        "msg": "Client machine announced
            it is not reachable on the
            provided lidar data port;
            check that udp_dest and
            udp_port_lidar configured
            on the sensor matches
            client IP and port.",
        "msg_verbose": "Failed to send
            lidar UDP data to destination
            host 169.254.175.254:7502",
        "realtime": "40842065146"
      },
      {
        "active": true,
        "category": "UDP_TRANSMISSION",
        "cursor": 3,
        "id": "0x01000016",
        "level": "WARNING",
        "msg": "Could not send lidar data
```

(continues on next page)

(continued from previous page)

```
            UDP packet to host;
            check that network is up.",
        "msg_verbose": "Failed to send
            lidar UDP data to destination
            host 169.254.175.254:7502",
        "realtime": "414261086316"
      },
      {
        "active": true,
        "category": "UDP_TRANSMISSION",
        "cursor": 0,
        "id": "0x01000018",
        "level": "WARNING",
        "msg": "Client machine announced
            it is not reachable on the provided
            not reachable on IMU data port;
            check that udp_dest and
            udp_port_imu configured on the
            sensor matches client IP
            and port.",
        "msg_verbose": "Failed to send imu
            UDP data to destination host
            169.254.175.254:7503",
        "realtime": "39850161524"
      },
      {
        "active": true,
        "category": "UDP_TRANSMISSION",
        "cursor": 4,
        "id": "0x01000019",
        "level": "WARNING",
        "msg": "Could not send IMU UDP
            packet to host; check that
            network is up.",
        "msg_verbose": "Failed to send
            imu UDP data to destination
            host 169.254.175.254:7503",
        "realtime": "414266339945"
      }
    ]
}
```

Example 2: `get_alerts <START_CURSOR><MODE>`

Description: Returns JSON-formatted sensor diagnostic information. A valid value for **MODE** is either `summary` or `default`. Setting the **MODE** argument to `SUMMARY` provides a less verbose version of the `get_alerts <START_CURSOR>` command as referenced above.

Note: Valid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=1

  `get_alerts 1`

- Example: Calling get_alerts with START_CURSOR=2 and MODE=summary

  `get_alerts 2 summary`

Invalid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=summary and MODE=2

  `get_alerts summary 2`

Note: The order in which the START_CURSOR and MODE arguments are specified for the get_alerts TCP command must be followed when providing both arguments, with START_CURSOR preceding the MODE. The arguments can be specified in any order when calling the equivalent /api/v1/sensor/alerts HTTP endpoint.

The example shown in the code block is for `get_alerts summary`.

```
{
   "log":
  [
    {
        "cursor": 0,
        "id": "0x01000018",
        "realtime": "80395661548",
        "active": true
    },
    {
        "cursor": 1,
        "id": "0x01000015",
        "realtime": "81386289401",
        "active": true
    },
    {
        "cursor": 2,
        "id": "0x01000014",
        "realtime": "2730854039127",
        "active": true
    },
    {
```

(continues on next page)

(continued from previous page)

```
        "cursor": 3,
        "id": "0x01000014",
        "realtime": "2740849252064",
        "active": false
    },
    {
        "cursor": 4,
        "id": "0x01000011",
        "realtime": "8835209059341",
        "active": true
    },
    {
        "cursor": 5,
        "id": "0x01000016",
        "realtime": "8835240707086",
        "active": true
    },
    {
        "cursor": 6,
        "id": "0x01000019",
        "realtime": "8835245794318",
        "active": true
    },
    {
        "cursor": 7,
        "id": "0x01000011",
        "realtime": "8837298086954",
        "active": false
    },
    {
        "cursor": 8,
        "id": "0x01000015",
        "realtime": "10742075130316",
        "active": false
    },
    {
        "cursor": 9,
        "id": "0x01000016",
        "realtime": "10742075535820",
        "active": false
    },
    {
        "cursor": 10,
        "id": "0x01000018",
        "realtime": "10742075793868",
        "active": false
    },
    {
        "cursor": 11,
        "id": "0x01000019",
        "realtime": "10742076051916",
        "active": false
    },
    {
```

(continues on next page)

(continued from previous page)

```
        "cursor": 12,
        "id": "0x01000015",
        "realtime": "10799083353303",
        "active": true
    },
    {
        "cursor": 13,
        "id": "0x01000018",
        "realtime": "10799092477143",
        "active": true
    },
    {
        "cursor": 14,
        "id": "0x01000016",
        "realtime": "11782640857349",
        "active": true
    }
  ],
        "next_cursor": 15,
        "active":
  [
    {
        "cursor": 12,
        "id": "0x01000015",
        "realtime": "10799083353303"
    },
    {
        "cursor": 14,
        "id": "0x01000016",
        "realtime": "11782640857349"
    },
    {
        "cursor": 13,
        "id": "0x01000018",
        "realtime": "10799092477143"
    }
  ]
}
```

12.3.7 `get_lidar_data_format`

Description: Returns JSON-formatted response that describes the structure of a lidar packet.

- `columns_per_frame`: Number of measurement columns per frame. This can be 512, 1024, or 2048, depending upon the set lidar mode.
- `columns_per_packet`: Number of measurement blocks contained in a single lidar packet. Currently in v2.2.0 and earlier, this is 16. **Note:** This is not user configurable.
- `pixel_shift_by_row`: Offset in terms of pixel count. Can be used to destagger image. Varies by lidar mode. Length of this array is equal to the number of channels of the sensor. **Note:** The new value is centered around zero with both positive and negative values such that the destaggered image corresponds to the sensor's azimuth angles. The old destagger values introduced an offset to shift the *pixel_shift_by_row* values so they are all non-negative.

- `pixels_per_column`: Number of channels of the sensor.
- `column_window`: Index of measurement blocks that are active. Default is [0, lidar_mode-1], e.g. [0,1023]. If there is an azimuth window set, this parameter will reflect which measurement blocks of data are within the region of interest.
- `udp_profile_lidar` - Lidar data profile format [Default LEGACY].
- `udp_profile_lidar` - Lidar data profile format [Default LEGACY].
- `udp_profile_imu` - IMU data profile format [Default LEGACY].

Note: This command only works when the sensor is in **RUNNING** status.

```
{
  "column_window": [0, 1023],
  "columns_per_frame": 1024,
  "columns_per_packet": 16,
  "pixel_shift_by_row": [
    12, 4, -4, -12, 12, 4, -4,
   -12, 12, 4, -4, -12, 12, 4,
   -4, -12, 12, 4, -4, -12, 12,
    4, -4, -12, 12, 4, -4, -12,
    12, 4, -4, -12, 12, 4, -4,
   -12, 12, 4, -4, -12, 12, 4,
   -4, -12, 12, 4, -4, -12, 12,
    4, -4, -12, 12, 4, -4, -12,
    12, 4, -4, -12, 12, 4, -4,
   -12, 12, 4, -4, -12, 12, 4,
   -4, -12, 12, 4, -4, -12, 12,
    4, -4, -12, 12, 4, -4, -12,
    12, 4, -4, -12, 12, 4, -4,
   -12, 12, 4, -4, -12, 12, 4,
   -4, -12, 12, 4, -4, -12, 12,
    4, -4, -12, 12, 4, -4, -12,
    12, 4, -4, -12, 12, 4, -4,
   -12, 12, 4, -4, -12, 12, 4,
   -4, -12],
  "pixels_per_column": 128,
  "udp_profile_imu": "LEGACY",
  "udp_profile_lidar": "LEGACY"
}
```

12.3.8 `get_calibration_status`

Description: Returns JSON-formatted calibration status of the sensor reflectivity. `valid`: true/false depending on calibration status. `timestamp`: if valid is true; time at which the calibration was completed.

```
{
  "reflectivity":
   {
     "valid": true,
     "timestamp": "2021-10-05T00:02:36"
   }
}
```

12.3.9 `get_telemetry`

Description: Returns JSON-formatted response that provides sensor system state information. This includes the FPGA **Timestamp** in `ns` (Nanoseconds) at which the information was collected from the FPGA, **Lidar Input Voltage** in `mv` (Millivolt), **Lidar Input Current** in `ma` (Milliamp), **Internal Temperature** of the sensor in `°C` (Degree Celsius) and **Phase Lock status** namely `LOCKED`, `LOST`, `DISABLED`.

Note: Using `get_telemetry`, Internal temperature can only be measured with **Rev 06** and above sensors.

Note: **Phase lock** output will not indicate loss of lock if the PTP source is lost.

```
{
  "input_current_ma": 758,
  "input_voltage_mv": 23606,
  "internal_temperature_deg_c": 45,
  "phase_lock_status": "DISABLED",
  "timestamp_ns": 2962666299310
}
```

13 HTTP API Reference Guide

This reference guide documents the interface for HTTP API and is accessible via `/api/v1` on the sensor hosted HTTP server.

The sensor can be queried and configured using an HTTP GET requests. This can be done using several different tools such as HTTPie, cURL, Advanced REST Client, etc.

Here is an example using **curl** command:

```
$ curl --request GET --url http://169.254.198.184/api/v1/sensor/metadata/lidar_intrinsics

 {
   "lidar_to_sensor_transform": [-1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 1, 36.18, 0, 0, 0, 1]
 }
```

13.1 Sensor Metadata

13.1.1 GET /api/v1/sensor/metadata/sensor_info

`GET 169.254.198.184/api/v1/sensor/metadata/sensor_info`

Get the sensor information

```
GET /api/v1/sensor/metadata/sensor_info HTTP/1.1
 Host: 169.254.198.184

 HTTP/1.1 200 OK
 content-length: 283
 content-type: application/json

 {
   "build_date": "2022-07-29T23:56:15Z",
   "build_rev": "v2.4.0-omega.2",
   "image_rev": "ousteros-image-prod-aries-v2.4.0-omega.2+20220730010801.staging",
   "initialization_id": 9599937,
   "prod_line": "OS-1-128",
   "prod_pn": "840-103575-06",
   "prod_sn": "122201000998",
   "status": "RUNNING"
}
```

statuscode: 200 No error

Description: Returns JSON-formatted response that includes serial number, product number, FW image revision and sensor status along with other parameters as shown is provided.

13.1.2 GET /api/v1/sensor/metadata/lidar_data_format

`GET 169.254.198.184/api/v1/sensor/metadata/lidar_data_format`
Get the sensor lidar data format

```
GET /api/v1/sensor/metadata/lidar_data_format HTTP/1.1
  Host: 169.254.198.184

  HTTP/1.1 200 OK
  content-type: application/json
  content-length: 661

  {
     "column_window": [0, 1023],
     "columns_per_frame": 1024,
     "columns_per_packet": 16,
     "pixel_shift_by_row": [
       12, 4, -4, -12, 12, 4, -4,
      -12, 12, 4, -4, -12, 12, 4,
      -4, -12, 12, 4, -4, -12, 12,
       4, -4, -12, 12, 4, -4, -12,
       12, 4, -4, -12, 12, 4, -4,
      -12, 12, 4, -4, -12, 12, 4,
      -4, -12, 12, 4, -4, -12, 12,
       4, -4, -12, 12, 4, -4, -12,
       12, 4, -4, -12, 12, 4, -4,
      -12, 12, 4, -4, -12, 12, 4,
      -4, -12, 12, 4, -4, -12, 12,
       4, -4, -12, 12, 4, -4, -12,
       12, 4, -4, -12, 12, 4, -4,
      -12, 12, 4, -4, -12, 12, 4,
       4, -12, 12, 4, -4, -12, 12,
       4, -4, -12, 12, 4, -4, -12,
       12, 4, -4, -12, 12, 4, -4,
      -12, 12, 4, -4, -12, 12, 4,
      -4, -12],
     "pixels_per_column": 128,
     "udp_profile_imu": "LEGACY",
     "udp_profile_lidar": "LEGACY"
}
```

statuscode: 200 No error

Description: Returns JSON-formatted response that describes the structure of a lidar packet.

`columns_per_frame`: Number of measurement columns per frame. This can be 512, 1024, or 2048, depending upon the set lidar mode.

`columns_per_packet`: Number of measurement blocks contained in a single lidar packet. Currently in v2.2.0 and earlier, this is 16. **Note:** This is not user configurable.

`pixel_shift_by_row`: Offset in terms of pixel count. Can be used to destagger image. Varies by lidar mode. Length of this array is equal to the number of channels of the sensor.

`pixels_per_column`: Number of channels of the sensor.

`column_window`: Index of measurement blocks that are active. Default is [0, lidar_mode-1], e.g. [0,1023]. If there is an azimuth window set, this parameter will reflect which measurement blocks of data are within the region of interest.

`udp_profile_lidar`: Lidar data profile format. Default LEGACY.

`udp_profile_lidar`: Lidar data profile format. Default LEGACY.

`udp_profile_imu`: IMU data profile format. Default LEGACY.

NOTE: This command only works when the sensor is in **RUNNING** status.

13.1.3 GET /api/v1/sensor/metadata/beam_intrinsics

`GET 169.254.198.184/api/v1/sensor/metadata/beam_intrinsics`

Get the sensor beam intrinsics

```
GET /api/v1/sensor/metadata/beam_intrinsics HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-type: application/json
content-length: 4506

 {
   "beam_azimuth_angles": [4.2300000000000004, 1.4099999999999999, -1.3999999999999999, -4.21,
     4.2199999999999998, 1.4199999999999999, -1.4099999999999999, -4.2199999999999998, 4.2300000000000004,
     1.4099999999999999, -1.4199999999999999, -4.21, 4.2300000000000004, 1.4199999999999999,
    -1.3999999999999999, -4.2000000000000002, 4.2300000000000004, 1.4099999999999999, -1.3899999999999999,
    -4.21, 4.25, 1.4299999999999999, -1.4099999999999999, -4.2199999999999998, 4.2400000000000002,
     1.4399999999999999, -1.4099999999999999, -4.2199999999999998, 4.2300000000000004, 1.4199999999999999,
    -1.3799999999999999, -4.2199999999999998, 4.2400000000000002, 1.4199999999999999, -1.3999999999999999,
    -4.2300000000000004, 4.2599999999999998, 1.4399999999999999, -1.4099999999999999, -4.2300000000000004,
     4.2400000000000002, 1.4399999999999999, -1.4099999999999999, -4.2300000000000004, 4.2400000000000002,
     1.4199999999999999, -1.4099999999999999, -4.2400000000000002, 4.25, 1.4199999999999999,
    -1.3899999999999999, -4.2199999999999998, 4.25, 1.4099999999999999, -1.3999999999999999,
    -4.2300000000000004, 4.2400000000000002, 1.4299999999999999, -1.4099999999999999, -4.2300000000000004,
     4.2400000000000002, 1.4199999999999999, -1.4099999999999999, -4.2300000000000004, 4.25,
     1.4199999999999999, -1.3999999999999999, -4.2400000000000002, 4.2400000000000002, 1.4399999999999999,
    -1.3999999999999999, -4.2400000000000002, 4.2400000000000002, 1.4299999999999999, -1.3999999999999999,
    -4.2400000000000002, 4.25, 1.4299999999999999, -1.4099999999999999, -4.2400000000000002, 4.25,
     1.4199999999999999, -1.4199999999999999, -4.2199999999999998, 4.2400000000000002, 1.4299999999999999,
    -1.3999999999999999, -4.2400000000000002, 4.25, 1.4399999999999999, -1.4099999999999999,
    -4.2400000000000002, 4.25, 1.4199999999999999, -1.4099999999999999, -4.2400000000000002, 4.25,
     1.4199999999999999, -1.4099999999999999, -4.25, 4.2599999999999998, 1.4299999999999999,
    -1.4099999999999999, -4.2599999999999998, 4.2699999999999996, 1.4299999999999999, -1.3999999999999999,
    -4.2400000000000002, 4.25, 1.4299999999999999, -1.3999999999999999, -4.2400000000000002,
     4.2599999999999998, 1.4199999999999999, -1.3999999999999999, -4.25, 4.2599999999999998,
     1.4299999999999999, -1.4099999999999999, -4.2599999999999998, 4.2599999999999998, 1.4199999999999999,
    -1.4199999999999999, -4.2599999999999998, 4.25, 1.4199999999999999, -1.4199999999999999,
    -4.2699999999999996],
   "beam_altitude_angles": [21.010000000000002, 20.719999999999999,
     20.420000000000002, 20.100000000000001, 19.789999999999999, 19.48, 19.170000000000002, 18.84,
```
(continues on next page)

(continued from previous page)

```
     18.550000000000001, 18.219999999999999, 17.899999999999999, 17.600000000000001, 17.27,
     16.960000000000001, 16.649999999999999, 16.32, 15.970000000000001, 15.65, 15.34, 15.01, 14.67, 14.35,
     14.01, 13.68, 13.33, 13.02, 12.67, 12.34, 11.99, 11.65, 11.33, 10.98, 10.640000000000001,
     10.279999999999999, 9.9499999999999993, 9.6099999999999994, 9.2699999999999996, 8.9199999999999999,
     8.5700000000000003, 8.2300000000000004, 7.8799999999999999, 7.54, 7.1799999999999997,
     6.8399999999999999, 6.4699999999999998, 6.1299999999999999, 5.7800000000000002, 5.4500000000000002,
     5.0899999999999999, 4.7300000000000004, 4.4100000000000001, 4.0499999999999998, 3.6899999999999999,
     3.3199999999999998, 2.98, 2.6299999999999999, 2.27, 1.9299999999999999, 1.5700000000000001, 1.22,
     0.84999999999999998, 0.5, 0.14999999999999999, -0.19, -0.55000000000000004, -0.92000000000000004,
    -1.25, -1.6299999999999999, -1.98,-2.3100000000000001, -2.6699999999999999, -3.04,
    -3.3999999999999999, -3.7400000000000002, -4.0899999999999999, -4.4500000000000002,
    -4.7999999999999998, -5.1500000000000004, -5.5, -5.8700000000000001, -6.21, -6.5700000000000003,
    -6.9100000000000001, -7.25, -7.6200000000000001, -7.9500000000000002, -8.3000000000000007,
    -8.6500000000000004, -9.0099999999999998, -9.3499999999999996, -9.6899999999999995,
    -10.050000000000001, -10.390000000000001, -10.74, -11.09, -11.42, -11.77, -12.109999999999999,
    -12.449999999999999, -12.800000000000001, -13.140000000000001, -13.470000000000001,
    -13.81, -14.15, -14.48, -14.82, -15.130000000000001, -15.470000000000001, -15.81,
    -16.149999999999999, -16.48, -16.800000000000001, -17.149999999999999, -17.48, -17.789999999999999,
    -18.120000000000001, -18.469999999999999, -18.77, -19.09, -19.420000000000002, -19.73,
    -20.059999999999999, -20.359999999999999, -20.690000000000001, -21, -21.32, -21.620000000000001,
    -21.940000000000001],
   "lidar_origin_to_beam_origin_mm": 15.805999999999999
 }
```

status code: 200 No error

Description: Returns JSON-formatted beam altitude and azimuth offsets, in degrees. Length of arrays is equal to the number of channels in the sensor. Also returns distance between lidar origin and beam origin in mm, to be used for point cloud calculations.

13.1.4 GET /api/v1/sensor/metadata/imu_intrinsics

`GET 169.254.198.184/api/v1/sensor/metadata/imu_intrinsics`
    Get the sensor imu intrinsics

```
GET /api/v1/sensor/metadata/imu_intrinsics HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-type: application/json
content-length: 117
 {
   "imu_to_sensor_transform": [1, 0, 0, 6.2530000000000001,
    0, 1, 0, -11.775, 0, 0, 1, 7.6449999999999996, 0, 0, 0, 1]
 }
```

status code: 200 No error

Description: Returns JSON-formatted IMU transformation matrix needed to transform to the Sensor Coordinate Frame.

13.1.5 GET /api/v1/sensor/metadata/lidar_intrinsics

`GET 169.254.198.184/api/v1/sensor/metadata/lidar_intrinsics`
Get the sensor lidar intrinsics

```
GET /api/v1/sensor/metadata/lidar_intrinsics HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-type: application/json
content-length: 85

 {
   "lidar_to_sensor_transform": [-1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 1, 36.18, 0, 0, 0, 1]
 }
```

status code: 200 No error

Description: Returns JSON-formatted lidar transformation matrix needed to transform to the Sensor Coordinate Frame.

13.1.6 GET /api/v1/sensor/metadata/calibration_status

`GET 169.254.198.184/api/v1/sensor/metadata/calibration_status`
Get the sensor calibration status

```
GET /api/v1/sensor/metadata/calibration_status HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-type: application/json
content-length: 69

 {
   "reflectivity":
     {
       "timestamp": "2021-10-05T00:02:36",
       "valid": true
     }
 }
```

status code: 200 No error

Description: Returns JSON formatted calibration status of the sensor reflectivity. `valid`: true/false depending on calibration status. `timestamp`: if valid is true; time at which the calibration was completed.

13.1.7 GET /api/v1/sensor/config

`GET 169.254.198.184/api/v1/sensor/config`

Get sensor configuration parameter

```
GET /api/v1/sensor/config HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-length: 715
content-type: application/json

   {
       "azimuth_window": [
           0,
           360000
       ],
       "columns_per_packet": 16,
       "lidar_mode": "1024x10",
       "multipurpose_io_mode": "OFF",
       "nmea_baud_rate": "BAUD_9600",
       "nmea_ignore_valid_char": 0,
       "nmea_in_polarity": "ACTIVE_HIGH",
       "nmea_leap_seconds": 0,
       "operating_mode": "NORMAL",
       "phase_lock_enable": false,
       "phase_lock_offset": 0,
       "signal_multiplier": 1,
       "sync_pulse_in_polarity": "ACTIVE_HIGH",
       "sync_pulse_out_angle": 360,
       "sync_pulse_out_frequency": 1,
       "sync_pulse_out_polarity": "ACTIVE_HIGH",
       "sync_pulse_out_pulse_width": 10,
       "timestamp_mode": "TIME_FROM_INTERNAL_OSC",
       "udp_dest": "169.254.198.184",
       "udp_port_imu": 7503,
       "udp_port_lidar": 7502,
       "udp_profile_imu": "LEGACY",
       "udp_profile_lidar": "RNG19_RFL8_SIG16_NIR16_DUAL"
   }
```

status code: 200 No error

Description: Please refer to *Description- Configurable Parameters* section for detailed description on sensor configurable parameters.

13.1.8 POST /api/v1/sensor/config

Description

- Currently the lidar_mode=1024x10, to change the lidar mode to "512X10", use the command: `http POST 169.254.198.184/api/v1/sensor/config "parameter"="value"`.
- Note: To identify all parameters that can be changed using this command please refer to *get_config_param*.

Example 1: `Valid` sensor configuration to change lidar_mode shown below:

`POST http POST 169.254.198.184/api/v1/sensor/config "lidar_mode"="512x10"`

```
POST /api/v1/sensor/config HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 204 No Content
```

Sensor config after `POST` command:

```
HTTP/1.1 200 OK
content-length: 767
content-type: application/json
 {
   "auto_start_flag": 1,
   "azimuth_window": [90000, 270000],
   "columns_per_packet": 16,
   "lidar_mode": "512x10",
   "multipurpose_io_mode": "OFF",
   "nmea_baud_rate": "BAUD_9600",
   "nmea_ignore_valid_char": 0,
   "nmea_in_polarity": "ACTIVE_HIGH",
   "nmea_leap_seconds": 0,
   "operating_mode": "NORMAL",
   "phase_lock_enable": false,
   "phase_lock_offset": 0,
   "signal_multiplier": 1,
   "sync_pulse_in_polarity": "ACTIVE_HIGH",
   "sync_pulse_out_angle": 360,
   "sync_pulse_out_frequency": 1,
   "sync_pulse_out_polarity": "ACTIVE_HIGH",
   "sync_pulse_out_pulse_width": 10,
   "timestamp_mode": "TIME_FROM_INTERNAL_OSC",
   "udp_dest": "169.254.198.184",
   "udp_port_imu": 7503,
   "udp_port_lidar": 7502,
   "udp_profile_imu": "LEGACY",
   "udp_profile_lidar": "RNG19_RFL8_SIG16_NIR16_DUAL"
 }
```

status code: 204 server has successfully fulfilled the request.

Example 2: `Invalid` sensor configuration to change lidar_mode shown below:

- Currently the lidar_mode=1024x10, to change the lidar mode to "512X10", use the command: `http POST 169.254.216.107/api/v1/sensor/config "parameter"="value"`.
- Note: An incorrect value for a valid key i.e.., "lidar_mode" is given to show the error message that would be prompted. In this case Lidar_mode is set to **511x10**.

`POST http POST 169.254.216.107/api/v1/sensor/config "lidar_mode"="511x10"`

```
POST /api/v1/sensor/config HTTP/1.1
Host: 169.254.216.107
```

Sensor config after `POST` command:

```
HTTP/1.1 400 Bad Request
content-length: 102
content-type: application/json

{
  "error": {
  "title": "While processing key 'lidar_mode' encountered error: '511x10' is not supported"
           }
}
```

status code: 400 Server cannot or will not process the request due to something that is perceived to be a client error.

13.1.9 GET /api/v1/sensor/metadata

`GET 169.254.198.184/api/v1/sensor/metadata`
Get the sensor metadata information

```
GET /api/v1/sensor/metadata HTTP/1.1
 Host: 169.254.198.184

HTTP/1.1 200 OK
content-length: 4163
content-type: application/json

        {
        "beam_intrinsics": {
            "beam_altitude_angles": [
                20.95, 20.67, 20.36, 20.03,
                19.73, 19.41, 19.11, 18.76,
                18.47, 18.14, 17.82, 17.5,
                17.19, 16.86, 16.53, 16.2,
                15.89, 15.56, 15.23, 14.9,
                14.57, 14.23, 13.9,  13.57,
                13.25, 12.91, 12.57, 12.22,
                11.9,  11.55, 11.2,  10.87,
                10.54, 10.18, 9.84, 9.51,
                9.15, 8.81, 8.47, 8.11,
                7.78, 7.43, 7.08, 6.74,
```

(continues on next page)

(continued from previous page)

```
            6.39, 6.04, 5.7,  5.34,
            4.98, 4.64, 4.29, 3.93,
            3.58, 3.24, 2.88, 2.53,
            2.17, 1.82, 1.47, 1.12,
            0.78, 0.41, 0.07, -0.28,
            -0.64, -0.99, -1.35, -1.7,
            -2.07, -2.4, -2.75, -3.11,
            -3.46, -3.81, -4.15, -4.5,
            -4.86, -5.22, -5.57, -5.9,
            -6.27, -6.61, -6.97, -7.3,
            -7.67, -8.01, -8.35, -8.69,
            -9.05, -9.38, -9.71, -10.07,
            -10.42, -10.76, -11.09, -11.43,
            -11.78, -12.12, -12.46, -12.78,
            -13.15, -13.46, -13.8, -14.12,
            -14.48, -14.79, -15.11, -15.46,
            -15.79, -16.12, -16.45, -16.76,
            -17.11, -17.44, -17.74, -18.06,
            -18.39, -18.72, -19.02, -19.32,
            -19.67, -19.99, -20.27, -20.57,
            -20.92, -21.22, -21.54, -21.82
        ],
        "beam_azimuth_angles": [
            4.21, 1.41, -1.4, -4.22, 4.22,
            1.41, -1.4, -4.23, 4.21,
            1.4, -1.42, -4.2, 4.22,
            1.41, -1.4, -4.23, 4.21,
            1.41, -1.41, -4.21, 4.22,
            1.4, -1.41, -4.2, 4.22,
            1.42, -1.4, -4.2, 4.22,
            1.41, -1.42, -4.21, 4.22,
            1.41, -1.4, -4.21, 4.2,
            1.4, -1.4, -4.22, 4.21,
            1.41, -1.41, -4.21, 4.22,
            1.41, -1.4, -4.21, 4.21,
            1.41, -1.4, -4.21, 4.2,
            1.41, -1.4, -4.21, 4.2,
            1.4, -1.41, -4.21, 4.22,
            1.4, -1.4, -4.21, 4.22,
            1.42, -1.4, -4.2, 4.2,
            1.42, -1.4, -4.22, 4.22,
            1.41, -1.4, -4.2, 4.23,
            1.41, -1.4, -4.2, 4.21,
            1.41, -1.4, -4.21, 4.21,
            1.41, -1.4, -4.21, 4.22,
            1.41, -1.39, -4.21, 4.23,
            1.41, -1.39, -4.22, 4.23,
            1.4, -1.4, -4.2, 4.21,
            1.41, -1.41, -4.2, 4.22,
            1.42, -1.39, -4.22, 4.24,
            1.41, -1.41, -4.22, 4.23,
            1.41, -1.39, -4.21, 4.23,
            1.41, -1.39, -4.2, 4.23,
            1.4, -1.39, -4.2, 4.22,
```

(continues on next page)

(continued from previous page)

```
                1.42, -1.39, -4.2
            ],
            "lidar_origin_to_beam_origin_mm": 15.806
        },
        "calibration_status": {
            "reflectivity": {
                "timestamp": "2022-01-08T10:00:38",
                "valid": true
            }
        },
        "config_params": {
            "azimuth_window": [
                0,
                360000
            ],
            "columns_per_packet": 16,
            "lidar_mode": "1024x10",
            "multipurpose_io_mode": "OFF",
            "nmea_baud_rate": "BAUD_9600",
            "nmea_ignore_valid_char": 0,
            "nmea_in_polarity": "ACTIVE_HIGH",
            "nmea_leap_seconds": 0,
            "operating_mode": "NORMAL",
            "phase_lock_enable": false,
            "phase_lock_offset": 0,
            "signal_multiplier": 1,
            "sync_pulse_in_polarity": "ACTIVE_HIGH",
            "sync_pulse_out_angle": 360,
            "sync_pulse_out_frequency": 1,
            "sync_pulse_out_polarity": "ACTIVE_HIGH",
            "sync_pulse_out_pulse_width": 10,
            "timestamp_mode": "TIME_FROM_INTERNAL_OSC",
            "udp_dest": "169.254.162.214",
            "udp_port_imu": 7503,
            "udp_port_lidar": 7502,
            "udp_profile_imu": "LEGACY",
            "udp_profile_lidar": "LEGACY"
        },
        "imu_intrinsics": {
            "imu_to_sensor_transform": [
                1,
                0,
                0,
                6.253,
                0,
                1,
                0,
                -11.775,
                0,
                0,
                1,
                7.645,
                0,
                0,
```

(continues on next page)

(continued from previous page)

```
            0,
            1
        ]
    },
    "lidar_data_format": {
        "column_window": [
            0,
            1023
        ],
        "columns_per_frame": 1024,
        "columns_per_packet": 16,
        "pixel_shift_by_row": [12, 4, -4, -12, 12,
            4, -4, -12, 12, 4, -4, -12, 12, 4, -4,
            -12, 12, 4, -4, -12, 12, 4, -4, -12, 12,
            4, -4, -12, 12, 4, -4, -12, 12, 4, -4,
            -12, 12, 4, -4, -12, 12, 4, -4, -12, 12,
            4, -4, -12, 12, 4, -4, -12, 12, 4, -4,
            -12, 12, 4, -4, -12, 12, 4, -4, -12, 12,
            4, -4, -12, 12, 4, -4, -12, 12, 4, -4,
            -12, 12, 4, -4, -12, 12, 4, -4, -12, 12,
            4, -4, -12, 12, 4, -4, -12, 12, 4, -4,
            -12, 12, 4, -4, -12, 12, 4, -4, -12, 12,
            4, -4, -12, 12, 4, -4, -12, 12, 4, -4,
            -12, 12, 4, -4, -12, 12, 4, -4, -12, 12,
            4, -4, -12
        ],
        "pixels_per_column": 128,
        "udp_profile_imu": "LEGACY",
        "udp_profile_lidar": "LEGACY"
    },
    "lidar_intrinsics": {
        "lidar_to_sensor_transform": [
            -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 1, 36.18, 0, 0, 0, 1
        ]
    },
    "sensor_info": {
        "build_date": "2022-07-29T23:56:15Z",
        "build_rev": "v2.4.0-omega.2",
        "image_rev": "ousteros-image-prod-aries-v2.4.0-omega.2+20220730010801.staging",
        "initialization_id": 9599937,
        "prod_line": "OS-1-128",
        "prod_pn": "840-103575-06",
        "prod_sn": "122201000998",
        "status": "RUNNING"
    }
}
```

status code: 200 No error

13.2 System

13.2.1 GET /api/v1/system/firmware

`GET 192.0.2.123/api/v1/system/firmware`

Get the firmware version of the sensor

```
GET /api/v1/system/firmware HTTP/1.1
Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
Host: 192.0.2.123
content-type: application/json; charset=UTF-8

{
  "fw": "ousteros-image-prod-aries-v2.0.0"
}
```

>json string fw Running firmware image name and version.

statuscode: 200 No error

13.2.2 GET /api/v1/system/network

`GET 192.0.2.123/api/v1/system/network`

Get the system network configuration.

```
GET /api/v1/system/network HTTP/1.1
Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8

{
   "carrier": true,
   "duplex": "full",
   "ethaddr": "bc:0f:a7:00:01:2c",
   "hostname": "os-991900123456",
   "ipv4": {
      "addr": "192.0.2.123/24",
      "link_local": "169.254.245.183/16",
      "override": null
   },
   "ipv6": {
      "link_local": "fe80::be0f:a7ff:fe00:12c/64"
   },
   "speed": 1000
}
```

>json boolean carrier: State of Ethernet link, `true` when physical layer is connected.

>json string duplex: Duplex mode of Ethernet link, `half` or `full`.

>json string ethaddr: Ethernet hardware (MAC) address.

>json string hostname: Hostname of the sensor, also used when requesting *DHCP* address and registering mDNS hostname.

>json object ipv4: See *ipv4 object*

>json string ipv6.link_local: Link-local IPv6 address.

>json integer speed: Ethernet physical layer speed in Mbps, should be 1000 Mbps.

statuscode: 200 No error

13.2.3 GET /api/v1/system/network/ipv4

`GET 192.0.2.123/api/v1/system/network/ipv4`
Get the IPv4 network configuration.

```
GET /api/v1/system/network/ipv4 HTTP/1.1
Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8

{
  "addr": "192.0.2.123/23",
  "link_local": "169.254.245.183/16",
  "override": null
}
```

>json string addr: Current global or private IPv4 address.

>json string link_local: Link-local IPv4 address.

>json string override: Static IP override value, this should match `addr`. This value will be `null` when unset and operating in *DHCP* or *link-local* modes.

statuscode: 200 No error

13.2.4 GET /api/v1/system/network/ipv4/override

`GET 192.0.2.123/api/v1/system/network/ipv4/override`
Get the current IPv4 static IP address override.

```
GET /api/v1/system/network/ipv4/override HTTP/1.1
Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8

null
```

>json string Static IP override value, this should match `addr`. This value will be `null` when unset and operating in *DHCP* mode.

statuscode: 200 No error

13.2.5 PUT /api/v1/system/network/ipv4/override

`PUT 192.0.2.123/api/v1/system/network/ipv4/override`
Override the default dynamic behavior and set a static IP address.

Note: The sensor will reset the network configuration after a short sub second delay (to allow for the HTTP response to be sent). After this delay the sensor will only be reachable on the newly set IPv4 address.

The sensor needs to be reachable either by *link-local* or dynamic *DHCP* configuration or by an existing static IP override from the host reconfiguring the sensor.

Warning: If an unreachable network address is set, the sensor will become unreachable. Tools such as avahi-browse, dns-sd, or mDNS browser can help with finding a sensor on a network.

Static IP override should only be used in special use cases. The *link-local* configuration is recommended where possible.

```
PUT /api/v1/system/network/ipv4/override HTTP/1.1
Content-Type: application/json
Host: 192.0.2.123

"192.0.2.100/24"
```

<json string: Static IP override value with subnet mask

>json string: Static IP override value that system will set after a short delay.

statuscode: 200 No error

13.2.6 DELETE /api/v1/system/network/ipv4/override

`DELETE 192.0.2.123/api/v1/system/network/ipv4/override`
Delete the static IP override value and return to dynamic configuration.

Note: The sensor will reset the network configuration after a short sub second delay (to allow for the HTTP response to be sent). After this delay the sensor will only be reachable on the newly set IPv4 address.

The sensor may be unreachable for several seconds while a *link-local* lease is obtained from the network or client machine.

```
DELETE /api/v1/system/network/ipv4/override HTTP/1.1
Host: 192.0.2.123
```

statuscode: 204 No error, no content

13.3 Time

13.3.1 GET /api/v1/time

`GET 192.0.2.123/api/v1/time`
Get the system time configuration for all timing components of the sensor.

```
GET /api/v1/time HTTP/1.1
  Host: 192.0.2.123


  HTTP/1.1 200 OK
  content-type: application/json; charset=UTF-8

  {
   "ptp": {
    "current_data_set": {
        "mean_path_delay": 0.0,
        "offset_from_master": 0.0,
        "steps_removed": 0
      },
    "parent_data_set": {
        "gm_clock_accuracy": 254,
        "gm_clock_class": 255,
        "gm_offset_scaled_log_variance": 65535,
        "grandmaster_identity": "bc0fa7.fffe.003aa6",
        "grandmaster_priority1": 128,
        "grandmaster_priority2": 128,
        "observed_parent_clock_phase_change_rate": 2147483647,
        "observed_parent_offset_scaled_log_variance": 65535,
        "parent_port_identity": "bc0fa7.fffe.003aa6-0",
        "parent_stats": 0
      },
    "port_data_set": {
        "announce_receipt_timeout": 3,
        "delay_mechanism": 1,
        "log_announce_interval": 1,
        "log_min_delay_req_interval": 0,
        "log_min_pdelay_req_interval": 0,
        "log_sync_interval": 0,
        "peer_mean_path_delay": 0,
        "port_identity": "bc0fa7.fffe.003aa6-1",
        "port_state": "LISTENING",
        "version_number": 2
      },
    "profile": "default",
    "time_properties_data_set": {
        "current_utc_offset": 37,
```

(continues on next page)

(continued from previous page)

```
            "current_utc_offset_valid": 0,
            "frequency_traceable": 0,
            "leap59": 0,
            "leap61": 0,
            "ptp_timescale": 1,
            "time_source": 160,
            "time_traceable": 0
        },
        "time_status_np": {
            "cumulative_scaled_rate_offset": 0.0,
            "gm_identity": "bc0fa7.fffe.003aa6",
            "gm_present": false,
            "gm_time_base_indicator": 0,
            "ingress_time": 0,
            "last_gm_phase_change": "0x0000'0000000000000000.0000",
            "master_offset": 0,
            "scaled_last_gm_phase_change": 0
        }
    },
    "sensor": {
        "multipurpose_io": {
            "mode": "OFF",
            "nmea": {
                "baud_rate": "BAUD_9600",
                "diagnostics": {
                    "decoding": {
                        "date_decoded_count": 0,
                        "last_read_message": "",
                        "not_valid_count": 0,
                        "utc_decoded_count": 0
                    },
                    "io_checks": {
                        "bit_count": 1,
                        "bit_count_unfiltered": 0,
                        "char_count": 0,
                        "start_char_count": 0
                    }
                },
                "ignore_valid_char": 0,
                "leap_seconds": 0,
                "locked": 0,
                "polarity": "ACTIVE_HIGH"
            },
            "sync_pulse_out": {
                "angle_deg": 360,
                "frequency_hz": 1,
                "polarity": "ACTIVE_HIGH",
                "pulse_width_ms": 10
            }
        },
        "sync_pulse_in": {
            "diagnostics": {
                "count": 1,
                "count_unfiltered": 0,
```

(continues on next page)

(continued from previous page)

```
            "last_period_nsec": 0
        },
        "locked": 0,
        "polarity": "ACTIVE_HIGH"
    },
    "timestamp": {
        "mode": "TIME_FROM_INTERNAL_OSC",
        "time": 311.36525506,
        "time_options": {
            "internal_osc": 311,
            "ptp_1588": 320,
            "sync_pulse_in": 1
        }
    }
},
"system": {
    "monotonic": 320.78890018,
    "realtime": 320.788918614,
    "tracking": {
        "frequency": 3.943,
        "last_offset": 0.0,
        "leap_status": "not synchronised",
        "ref_time_utc": 0.0,
        "reference_id": "00000000",
        "remote_host": "",
        "residual_frequency": 0.0,
        "rms_offset": 0.0,
        "root_delay": 1.0,
        "root_dispersion": 1.0,
        "skew": 0.0,
        "stratum": 0,
        "system_time_offset": -1e-09,
        "update_interval": 0.0
    }
 }
}
```

>json string: See sub objects for details.

statuscode: 200 No error

13.3.2 GET /api/v1/time/sensor

GET 169.254.198.184/api/v1/time/sensor
    Get the sensor time information

```
GET /api/v1/time/sensor HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-type: application/json
content-length: 773
```

(continues on next page)

(continued from previous page)

```
{
    "multipurpose_io": {
        "mode": "OFF",
        "nmea": {
            "baud_rate": "BAUD_9600",
            "diagnostics": {
                "decoding": {
                    "date_decoded_count": 0,
                    "last_read_message": "",
                    "not_valid_count": 0,
                    "utc_decoded_count": 0
                },
                "io_checks": {
                    "bit_count": 1,
                    "bit_count_unfiltered": 0,
                    "char_count": 0,
                    "start_char_count": 0
                }
            },
            "ignore_valid_char": 0,
            "leap_seconds": 0,
            "locked": 0,
            "polarity": "ACTIVE_HIGH"
        },
        "sync_pulse_out": {
            "angle_deg": 360,
            "frequency_hz": 1,
            "polarity": "ACTIVE_HIGH",
            "pulse_width_ms": 10
        }
    },
    "sync_pulse_in": {
        "diagnostics": {
            "count": 1,
            "count_unfiltered": 0,
            "last_period_nsec": 0
        },
        "locked": 0,
        "polarity": "ACTIVE_HIGH"
    },
    "timestamp": {
        "mode": "TIME_FROM_INTERNAL_OSC",
        "time": 27784.88125111,
        "time_options": {
            "internal_osc": 27784,
            "ptp_1588": 27795,
            "sync_pulse_in": 1
        }
    }
}
```

status code: 200 No error

Description: Returns JSON-formatted sensor timing configuration and status of udp `timestamp`, `sync_pulse_in`, and `multipurpose_io`. For more information on these parameters refer to the `get_time_info` TCP command.

13.3.3 GET /api/v1/time/system

`GET 192.0.2.123/api/v1/time/system`
Get the operating system time status. These values relate to the sensor operating system clocks, and not clocks related to hardware timestamp data from the lidar sensor.

```
GET /api/v1/time/system HTTP/1.1
Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8

{
  "monotonic": 345083.599570944,
  "realtime": 1551814510.730453,
  "tracking": {
    "frequency": -6.185,
    "last_offset": -3.315e-06,
    "leap_status": "normal",
    "ref_time_utc": 1551814508.1982567,
    "reference_id": "70747000",
    "remote_host": "ptp",
    "residual_frequency": -0.019,
    "rms_offset": 4.133e-06,
    "root_delay": 1e-09,
    "root_dispersion": 0.000128737,
    "skew": 1.14,
    "stratum": 1,
    "system_time_offset": 4.976e-06,
    "update_interval": 2
  }
}
```

>json float monotonic: Monotonic time of operating system. This timestamp never counts backwards and is the time since boot in seconds.

>json float realtime: Time in seconds since the Unix epoch, should match wall time if synchronized with external time source.

>json object tracking: Operating system time synchronization tracking status. See chronyc tracking documentation for more information.

statuscode: 200 No error

System `tracking` fields of interest:

rms_offset: Long-term average of the offset value.

system_time_offset: Time delta (in seconds) between the estimate of the operating system time and the current true time.

last_offset: Estimated local offset on the last clock update.

ref_time_utc: UTC Time at which the last measurement from the reference source was processed.

remote_host: This is either `ptp` if the system is synchronizing to a *PTP* time source or the address of a remote NTP server the system has selected if the sensor is connected to the Internet.

13.3.4 GET /api/v1/time/ptp

`GET 192.0.2.123/api/v1/time/ptp`

Get the status of the *PTP* time synchronization daemon.

Note: See the IEEE 1588-2008 standard for more details on the standard management messages.

```
GET /api/v1/time/ptp HTTP/1.1
Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8

{
  "current_data_set": {
    "mean_path_delay": 30110,
    "offset_from_master": 224159,
    "steps_removed": 1
  },
  "parent_data_set": {
    "gm_clock_accuracy": 33,
    "gm_clock_class": 6,
    "gm_offset_scaled_log_variance": 20061,
    "grandmaster_identity": "001747.fffe.700038",
    "grandmaster_priority1": 128,
    "grandmaster_priority2": 128,
    "observed_parent_clock_phase_change_rate": 2147483647,
    "observed_parent_offset_scaled_log_variance": 65535,
    "parent_port_identity": "001747.fffe.700038-1",
    "parent_stats": 0
  },
  "port_data_set": {
    "announce_receipt_timeout": 3,
    "delay_mechanism": 1,
    "log_announce_interval": 1,
    "log_min_delay_req_interval": 0,
    "log_min_pdelay_req_interval": 0,
    "log_sync_interval": 0,
    "peer_mean_path_delay": 0,
    "port_identity": "bc0fa7.fffe.00012c-1",
    "port_state": "SLAVE",
    "version_number": 2
  },
  "time_properties_data_set": {
```

(continues on next page)

(continued from previous page)

```
    "current_utc_offset": 37,
    "current_utc_offset_valid": 1,
    "frequency_traceable": 1,
    "leap59": 0,
    "leap61": 0,
    "ptp_timescale": 1,
    "time_source": 32,
    "time_traceable": 1
  },
  "time_status_np": {
    "cumulative_scaled_rate_offset": 0,
    "gm_identity": "001747.fffe.700038",
    "gm_present": true,
    "gm_time_base_indicator": 0,
    "ingress_time": 1551814546772493800,
    "last_gm_phase_change": "0x0000'0000000000000000.0000",
    "master_offset": 224159,
    "scaled_last_gm_phase_change": 0
  }
}
```

>json object current_data_set Result of the PMC `GET CURRENT_DATA_SET` command.

>json object parent_data_set Result of the PMC `GET PARENT_DATA_SET` command.

>json object port_data_set Result of the PMC `GET PORT_DATA_SET` command.

>json object time_properties_data_set Result of the PMC `GET TIME_PROPERTIES_DATA_SET` command.

>json object time_status_np Result of the PMC `GET TIME_STATUS_NP` command. This is a linuxptp non-portable command.

statuscode: 200 No error

Fields of interest:

current_data_set.offset_from_master Offset from master time source in nanoseconds as calculated during the last update from master.

parent_data_set.grandmaster_identity This should match the local grandmaster clock. If this displays the sensor's clock identity (derived from Ethernet hardware address) then this indicates the sensor is not properly synchronized to a grandmaster.

parent_data_set Various information about the selected master clock.

port_data_set.port_state This value will be `SLAVE` when a remote master clock is selected. See `parent_data_set` for selected master clock.

port_data_set Local sensor *PTP* configuration values. Grandmaster clock needs to match these for proper time synchronization.

time_properties_data_set *PTP* properties as given by master clock.

time_status_np.gm_identity Selected grandmaster clock identity.

time_status_np.gm_present True when grandmaster has been detected. This may stay true even if grandmaster goes off-line. Use `port_data_set.port_state` to determine up-to-date synchronization status. When this is false then the local clock is selected.

time_status_np.ingress_time Indicates when the last *PTP* message was received. Units are in nanoseconds.

time_status_np Linux *PTP* specific diagnostic values. The Red Hat manual provides some more information on these fields

13.3.5 GET /api/v1/time/ptp/profile

`GET 192.0.2.123/api/v1/time/ptp/profile`
Get the active PTP profile of the Ouster sensor

```
GET /api/v1/time/ptp/profile HTTP/1.1
   Content-Type: application/json
   Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
content-length: 9
content-type: application/json; charset=UTF-8

"gptp"
```

>json string: Active PTP profile.

statuscode: 200 No error

13.3.6 PUT /api/v1/time/ptp/profile

`PUT 192.0.2.123/api/v1/time/ptp/profile`
Change the PTP profile of the Ouster sensor

```
PUT /api/v1/time/ptp/profile HTTP/1.1
   Content-Type: application/json
   Host: 192.0.2.123

   "gptp"
```

```
HTTP/1.1 200 OK
content-length: 9
content-type: application/json; charset=UTF-8

"gptp"
```

<json string: PTP profile to be activated, valid options are `"default"`, `"gptp"`, and `"automotive-slave"`

>json string: Active PTP profile.

statuscode: 200 No error

13.4 Alerts, Diagnostics and Telemetry

13.4.1 GET /api/v1/sensor/alerts

Example 1: Using HTTPie command GET /api/v1/sensor/alerts

`GET 169.254.198.184/api/v1/sensor/alerts`
    Get the sensor lidar intrinsics

```
GET /api/v1/sensor/alerts HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-length: 3941
content-type: application/json

  {
      "active": [
          {
              "active": true,
              "category": "UDP_TRANSMISSION",
              "cursor": 8,
              "id": "0x01000018",
              "level": "WARNING",
              "msg": "Client machine announced it is not reachable on the provided not reachable on
                     IMU data port; check that udp_dest and udp_port_imu configured on the sensor
                     matches client IP and port.",
              "msg_verbose": "Failed to send imu UDP data to destination host 169.254.28.205:7503",
              "realtime": "1309574110356"
          }
      ],
      "log": [
          {
              "active": true,
              "category": "UDP_TRANSMISSION",
              "cursor": 0,
              "id": "0x01000018",
              "level": "WARNING",
              "msg": "Client machine announced it is not reachable on the provided not reachable on
                     IMU data port; check that udp_dest and udp_port_imu configured on the sensor
                     matches client IP and port.",
              "msg_verbose": "Failed to send imu UDP data to destination host 169.254.28.205:7503",
              "realtime": "87914851559"
          },
          {
              "active": true,
              "category": "UDP_TRANSMISSION",
              "cursor": 1,
              "id": "0x01000015",
              "level": "WARNING",
              "msg": "Client machine announced it is not reachable on the provided lidar data port;
                      check that udp_dest and udp_port_lidar configured on the sensor matches client
                      IP and port.",
              "msg_verbose": "Failed to send lidar UDP data to destination host 169.254.28.205:7502",
              "realtime": "88906828916"
```

(continues on next page)

(continued from previous page)

```
        },
        {
            "active": false,
            "category": "UDP_TRANSMISSION",
            "cursor": 2,
            "id": "0x01000015",
            "level": "WARNING",
            "msg": "Client machine announced it is not reachable on the provided lidar data port;
                   check that udp_dest and udp_port_lidar configured on the sensor matches client
                   IP and port.",
            "msg_verbose": "Cleared by reinitialization.",
            "realtime": "171640501848"
        },
        {
            "active": false,
            "category": "UDP_TRANSMISSION",
            "cursor": 3,
            "id": "0x01000018",
            "level": "WARNING",
            "msg": "Client machine announced it is not reachable on the provided not reachable on
                   IMU data port; check that udp_dest and udp_port_imu configured on the sensor matches
                   client IP and port.",
            "msg_verbose": "Cleared by reinitialization.",
            "realtime": "171640962692"
        },
        {
            "active": true,
            "category": "UDP_TRANSMISSION",
            "cursor": 4,
            "id": "0x01000018",
            "level": "WARNING",
            "msg": "Client machine announced it is not reachable on the provided not reachable on
                   IMU data port; check that udp_dest and udp_port_imu configured on the sensor
                   matches client IP and port.",
            "msg_verbose": "Failed to send imu UDP data to destination host 169.254.28.205:7503",
            "realtime": "188178225997"
        },
        {
            "active": true,
            "category": "UDP_TRANSMISSION",
            "cursor": 5,
            "id": "0x01000015",
            "level": "WARNING",
            "msg": "Client machine announced it is not reachable on the provided lidar data port;
                   check that udp_dest and udp_port_lidar configured on the sensor matches client
                   IP and port.",
            "msg_verbose": "Failed to send lidar UDP data to destination host 169.254.28.205:7502",
            "realtime": "189169539737"
        },
        {
            "active": false,
            "category": "UDP_TRANSMISSION",
            "cursor": 6,
            "id": "0x01000015",
```

(continues on next page)

(continued from previous page)

```
            "level": "WARNING",
            "msg": "Client machine announced it is not reachable on the provided lidar data port;
                   check that udp_dest and udp_port_lidar configured on the sensor matches client
                   IP and port.",
            "msg_verbose": "Cleared by reinitialization.",
            "realtime": "1293004145514"
        },
        {
            "active": false,
            "category": "UDP_TRANSMISSION",
            "cursor": 7,
            "id": "0x01000018",
            "level": "WARNING",
            "msg": "Client machine announced it is not reachable on the provided not reachable on
                   IMU data port; check that udp_dest and udp_port_imu configured on the sensor
                   matches client IP and port.",
            "msg_verbose": "Cleared by reinitialization.",
            "realtime": "1293004551057"
        },
        {
            "active": true,
            "category": "UDP_TRANSMISSION",
            "cursor": 8,
            "id": "0x01000018",
            "level": "WARNING",
            "msg": "Client machine announced it is not reachable on the provided not reachable on
                   IMU data port; check that udp_dest and udp_port_imu configured on the sensor
                   matches client IP and port.",
            "msg_verbose": "Failed to send imu UDP data to destination host 169.254.28.205:7503",
            "realtime": "1309574110356"
        }
    ],
    "next_cursor": 9
}
```

status code: 200 No error

Description: Returns JSON-formatted sensor diagnostic information.

Two lists will be returned, an **active** list and a **log** list. The **active** list will contain alert-trigger events for alerts that are currently active. An alert-trigger event will by-definition always have its active attribute set to true. There is no limit on the number of alert-trigger events that are displayed in the **active** event list. All currently active alert-trigger events will be displayed in the **active** event list.

The **log** list will contain all current and past alert-trigger and alert-clear events. An alert-clear event will by-definition have the exact same attributes and attribute values as its corresponding trigger event, with the exception of the realtime and cursor attributes which should have higher values, since an alert-clear event will always be received after an alert-trigger event. The **log** list has a length limit of 32 events with the oldest events automatically removed from the log list once a new event needs to be added to the **log** list and the **log** list length limit is reached, essentially acting as a FIFO (First In First Out) queue.

In addition to the **active** and **log** lists, `get_alerts` also returns a **next_cursor** field. Every alert event has a cursor attribute, which increments for every alert event logged. This can be used to track the alert activity that has been viewed and reduce message bandwidth. To do this, users are recommended to save the **next_cursor** field when calling `get_alerts` and then use that value as the `START_CURSOR` argument on the next `get_alerts` call to fetch only new **log** entries.

A valid value for **MODE** is either `summary` or `default`.

Note: Valid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=1

  `get_alerts 1`

- Example: Calling get_alerts with START_CURSOR=2 and MODE=summary

  `get_alerts 2 summary`

Invalid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=summary and MODE=2

  `get_alerts summary 2`

Note: The order in which the START_CURSOR and MODE arguments are specified for the get_alerts TCP command must be followed when providing both arguments, with START_CURSOR preceding the MODE. The arguments can be specified in any order when calling the equivalent /api/v1/sensor/alerts HTTP endpoint.

Example2: Using cURL command GET /api/v1/sensor/alerts

```
$ curl --request GET --url http://169.254.216.107/api/v1/sensor/alerts

 {
     "next_cursor": 10,
     "log":
     [
      {
         "active": true,
         "msg_verbose": "Failed to send imu UDP data to destination
                         host 169.254.117.239:7503",
         "cursor": 0,
         "id": "0x01000018",
         "category": "UDP_TRANSMISSION",
         "level": "WARNING",
         "msg": "Client machine announced it is not reachable on the
                 provided IMU data port; check that udp_dest and
                 udp_port_imu configured on the sensor matches client IP and port.
                 This Alert may occur on sensor startup if the client is not
                 listening at that time.",
         "realtime": "36290112071"
     },
     {
```

(continues on next page)

(continued from previous page)

```
        "active": true,
        "msg_verbose": "Failed to send lidar UDP data to destination
                        host 169.254.117.239:7502",
        "cursor": 1,
        "id": "0x01000015",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided
                lidar data port; check that udp_dest and udp_port_lidar configured
                on the sensor matches client IP and port.
                This Alert may occur on sensor startup if the client is not
                listening at that time.",
        "realtime": "37280855516"
    },
    {
        "active": false,
        "msg_verbose": "Cleared by reinitialization.",
        "cursor": 2,
        "id": "0x01000015",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided
                lidar data port; check that udp_dest and udp_port_lidar configured
                on the sensor matches client IP and port.
                This Alert may occur on sensor startup if the client is not listening
                at that time.",
        "realtime": "1324837582611"
    },
    {
        "active": false,
        "msg_verbose": "Cleared by reinitialization.",
        "cursor": 3,
        "id": "0x01000018",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided
                IMU data port; check that udp_dest and udp_port_imu configured
                on the sensor matches client IP and port.
                This Alert may occur on sensor startup if the client is not
                listening at that time.",
        "realtime": "1324837951308"
    },
    {
        "active": true,
        "msg_verbose": "Failed to send imu UDP data to destination
                        host 169.254.117.239:7503",
        "cursor": 4,
        "id": "0x01000018",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided IMU data port;
                check that udp_dest and udp_port_imu configured on the sensor matches client
                IP and port. This Alert may occur on sensor startup if the client is not
                listening at that time.",
```

(continues on next page)

(continued from previous page)

```
        "realtime": "1337796738411"
    },
    {
        "active": true,
        "msg_verbose": "Failed to send lidar UDP data to destination
                host 169.254.117.239:7502",
        "cursor": 5,
        "id": "0x01000015",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided lidar data port;
                check that udp_dest and udp_port_lidar configured on the sensor matches
                client IP and port. This Alert may occur on sensor startup if the client
                is not listening at that time.",
        "realtime": "1342789095053"
    },
    {
        "active": false,
        "msg_verbose": "Cleared by reinitialization.",
        "cursor": 6,
        "id": "0x01000015",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided lidar data port;
                check that udp_dest and udp_port_lidar configured on the sensor matches client
                IP and port. This Alert may occur on sensor startup if the client is not
                listening at that time.",
        "realtime": "2376756278143"
    },
    {
        "active": false,
        "msg_verbose": "Cleared by reinitialization.",
        "cursor": 7,
        "id": "0x01000018",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided IMU data port;
                check that udp_dest and udp_port_imu configured on the sensor matches
                client IP and port. This Alert may occur on sensor startup if the client
                is not listening at that time.",
        "realtime": "2376756665275"
    },
    {
        "active": true,
        "msg_verbose": "Failed to send imu UDP data to destination
                host 169.254.117.239:7503",
        "cursor": 8,
        "id": "0x01000018",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided IMU data port;
                check that udp_dest and udp_port_imu configured on the sensor matches
                client IP and port. This Alert may occur on sensor startup if the client
                is not listening at that time.",
```

(continues on next page)

(continued from previous page)

```
        "realtime": "2389805137860"
    },
    {
        "active": true,
        "msg_verbose": "Failed to send lidar UDP data to destination
                        host 169.254.117.239:7502",
        "cursor": 9,
        "id": "0x01000015",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided lidar
                data port; check that udp_dest and udp_port_lidar configured on the sensor
                matches client IP and port. This Alert may occur on sensor startup if the
                client is not listening at that time.",
        "realtime": "2390795752261"
    }
   ],
        "active":
   [
    {
        "active": true,
        "msg_verbose": "Failed to send lidar UDP data to destination
                        host 169.254.117.239:7502",
        "cursor": 9,
        "id": "0x01000015",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided lidar data port;
                check that udp_dest and udp_port_lidar configured on the sensor matches
                client IP and port. This Alert may occur on sensor startup if the client is
                not listening at that time.",
        "realtime": "2390795752261"
    },
    {
        "active": true,
        "msg_verbose": "Failed to send imu UDP data to destination
                        host 169.254.117.239:7503",
        "cursor": 8,
        "id": "0x01000018",
        "category": "UDP_TRANSMISSION",
        "level": "WARNING",
        "msg": "Client machine announced it is not reachable on the provided IMU data port;
                check that udp_dest and udp_port_imu configured on the sensor matches client
                IP and port. This Alert may occur on sensor startup if the client is not
                listening at that time.",
        "realtime": "2389805137860"
    }
   ]
}
```

13.4.2 GET /api/v1/diagnostics/dump

`GET 192.0.2.123/api/v1/diagnostics/dump`
Get the diagnostics files of the sensor

```
GET /api/v1/diagnostics/dump HTTP/1.1
Host: 192.0.2.123
```

```
HTTP/1.1 200 OK
content-disposition: attachment; filename="192.0.2.123_diagnostics-dump_29811b9e-2afc-11eb-ae01-
                                           bc0fa700190c.bin"
content-type: application/octet-stream

{binary data}
```

statuscode: 200 No error

13.4.3 GET /api/v1/sensor/telemetry

`GET 169.254.198.184/api/v1/sensor/telemetry`
Get the sensor telemetry information

```
GET /api/v1/sensor/telemetry HTTP/1.1
Host: 169.254.198.184

HTTP/1.1 200 OK
content-length: 150
content-type: application/json

 {
     "input_current_ma": 758,
     "input_voltage_mv": 23606,
     "internal_temperature_deg_c": 45,
     "phase_lock_status": "DISABLED",
     "timestamp_ns": 2962666299310
 }
```

status code: 200 No error

Description: Returns JSON-formatted response that provides sensor system state information. This includes the FPGA **Timestamp** in `ns` (Nanoseconds) at which the information was collected from the FPGA, **Lidar Input Voltage** in `mv` (Millivolt), **Lidar Input Current** in `ma` (Milliamp), **Internal Temperature** of the sensor in `°C` (Degree Celsius) and **Phase Lock status** namely `LOCKED`, `LOST`, `DISABLED`..

Note:

- **Internal temperature** can only be measured with Rev 06 and above sensors.
- **Phase lock** output will not indicate loss of lock if the PTP source is lost.

---

14 API Changelog

Version **v2.4.0**

Date 2022-08-31

Description

"Added"

- New HTTP Commands to help configure and read sensor config parameters (Refer to *Sensor Metadata* for more information).
  - GET /api/v1/sensor/config
  - POST /api/v1/sensor/config
- New optional argument for both TCP/HTTP commands <MODE> = "summary" (Refer to *get_alerts* for more information).

"Removed"

- `auto_start_flag` has been deprecated.
- `udp_ip` has been deprecated.
- `base_pn`, `base_sn` and `proto_rev` have been deprecated.

"Fixed"

- Pixel shift by row has been updated. Please refer to *Sensor Status and Calibration*.
- Bug in keep-alive behavior for HTTP 1.1.

Version **v2.3.0**

Date 2022-04-15

Description

"Added"

- Add additional options for config parameter udp_profile_lidar, refer to *Description- Configurable Parameters*.
- Add new TCP command get_telemetry, refer to *Sensor Status and Calibration*.
- Add the following GET HTTP Commands (Refer to *Sensor Metadata* for more information):
  - /api/v1/sensor/metadata/sensor_info
  - /api/v1/sensor/metadata/lidar_data_format
  - /api/v1/sensor/metadata/beam_intrinsics
  - /api/v1/sensor/metadata/imu_intrinsics
  - /api/v1/sensor/metadata/lidar_intrinsics

- /api/v1/sensor/metadata/calibration_status
- /api/v1/sensor/metadata
- /api/v1/sensor/telemetry
- /api/v1/time/sensor
- /api/v1/sensor/alerts

Version v2.2.0

Date 2021-12-18

Description

"Added"

- Add config parameter udp_profile_lidar, udp_profile_imu and their documentation
- Add initialization_id to get_sensor_info TCP command
- Add columns_per_packet to get_config_param TCP command

"Changed"

- The fields base_pn, base_sn, and proto_rev in get_sensor_info TCP command have been cleared

Version v2.1.3

Date 2021-10-22

Description

"Added"

- Added PN support

Version v2.1.2

Date 2021-07-16

Description

"Added"

- Add support for minor hardware revisions

Version v2.1.1

Date 2021-06-21

Description

"Added"

- Add configuration parameter `signal_multiplier` and its documentation

"Removed"

- Remove deprecated TCP command `set_data_dst_ip`
- Remove deprecated TCP command `get_data_dst_ip`
- Remove deprecated TCP command `set_udp_port_lidar`
- Remove deprecated TCP command `set_udp_port_imu`
- Remove deprecated TCP command `get_lidar_mode`
- Remove deprecated TCP command `set_lidar_mode`
- Remove deprecated TCP command `get_config_file_path`
- Remove deprecated TCP command `set_auto_start_flag`
- Remove deprecated TCP command `get_auto_start_flag`
- Remove deprecated TCP command `get_watchdog_status`

"Changed"

- Fixed azimuth_window parameter logic behavior. Notable changes:
- [0,0] now outputs only a single column instead of all columns.
- [1,2] results in sensor startup failure and an alert because there are no valid output columns.

Version **v2.0.0**

Date 2020-11-20

Description

"Added"

- Add TCP command `get_lidar_data_format`.
- Add in `azimuth_window` documentation.
- Add in commands `phase_lock_enable` and `phase_lock_offset` and their documentation.
- Add in verbose responses to parameter validation for TCP commands.
- Add in command `save_config_params` which supersedes the deprecated command `write_config_txt`, which will be deleted in future firmware.
- Add in command `get_config_param active` in favor of the deprecated command `get_config_txt`, which will be deleted in future firmware.
- Add in new STANDBY and WARMUP statuses.
- Add in parameter `operating_mode` in favor of the deprecated parameter `auto_start_flag`, which will be deleted in future firmware.
- Add in parameter `udp_dest` in favor of the deprecated parameter `udp_ip`, which will be deleted in future firmware. This is to be consistent with the `set_udp_dest_auto` parameter and to reflect that valid values can be hostnames in addition to ip addresses.
- Add in HTTP GET `api/v1/diagnostic/dump` endpoint.

"Removed"

- Remove deprecated TCP command `set_udp_ip`.

"Changed"

- TCP command `get_beam_intrinsics` now returns: 1) `lidar_origin_to_beam_origin_mm`, distance between the lidar origin and the beam origin in millimeters; and 2) beam altitude and azimuth angle arrays with padded zeros removed.
- `azimuth_window` parameter now in terms of millidegrees and implemented CCW.
- Deprecate `api/v1/system/time/` HTTP API and its sub-APIs and replace with `api/v1/time/`

Version **v1.13.0**

Date

Description

"Added"

- Add TCP command `set_udp_dest_auto`
- TCP command `get_alerts`, includes more descriptive errors for troubleshooting

"Changes"

- Packet Status now called Azimuth Data Block Status and is calculated differently
- Packets with bad CRC are now dropped upstream and replaced with `0` padded packets to ensure all packets are sent for each frame.
- Return format of TCP command `get_time_info` updated

"Removed"

- Removed reference to window_rejection_enable

Version **v1.12.0**

Date

Description

"Changes"

- Corrected IMU axis directions to match Sensor Coordinate Frame.
- Sensor Coordinate Frame section of sensor user manual for details on sensor coordinate frame. This change inverts IMU X, Y, and Z axis relative to v1.11.0.

Version **v1.11.0**

Date 2019-03-25

Description

- Add section on HTTP API commands.

- TCP Port now hard-coded to 7501; port is no longer configurable.
- Update to SYNC_PULSE_IN and MULTIPURPOSE_IO interface and configuration parameters (see details below).

Configuration parameters name changes:

- `pps_in_polarity` changed to `sync_pulse_in_polarity`
- `pps_out_mode` changed to `multipurpose_io_mode`
- `pps_out_polarity` changed to `sync_pulse_out_polarity`
- `pps_rate` changed to `sync_pulse_out_frequency`
- `pps_angle` changed to `sync_pulse_out_angle`
- `pps_pulse_width` changed to `sync_pulse_out_pulse_width`

New configuration parameters:

- `nmea_in_polarity`
- `nmea_ignore_valid_char`
- `nmea_baud_rate`
- `nmea_leap_seconds`

Configuration parameters option changes:

- timestamp_mode - `TIME_FROM_PPS` changed to `TIME_FROM_SYNC_PULSE_IN`
- multipurpose_io_mode (formerly pps_out_mode) - `OUTPUT_PPS_OFF` changed to `OFF` - `OUTPUT_FROM_PPS_IN_SYNCED` changed to `OUTPUT_FROM_SYNC_PULSE_IN` - Removed `OUTPUT_FROM_PPS_DEFINED_RATE` - Added `INPUT_NMEA_UART`

TCP command changes:

- **Added commands:**
  - `get_time_info`
- **Changed commands:**
  - `get_config_txt` (returned dictionary keys match parameter changes)
- **Removed commands:**
  - `set_pps_in_polarity`
  - `get_pps_out_mode`
  - `set_pps_out_mode`
  - `get_timestamp_mode`
  - `set_timestamp_mode`

Polarity changes: * `sync_pulse_in_polarity` was corrected to match parameter naming. * `sync_pulse_out_polarity` was corrected to match parameter naming.

Version **v1.10.0**

Date 2018-12-11

Description

"Added"

- Add `get_alerts`, `pps_rate` and `pps_angle` usage commands and expected output.

"Removed"

- Remove all references of `pulse_mode`.
- Remove TCP commands prior to v1.5.1.

Version **v1.9.0**

Date 2018-10-24

Description

"Changes"

- No TCP command change.

Version **v1.8.0**

Date 2018-10-11

Description

- `get_sensor_info` command gives `INITIALIZING`, `UPDATING`, `RUNNING`, `ERROR` and `UNCONFIGURED` status.

Version **v1.7.0**

Date 2018-09-05

Description

"Changes"

- No TCP command change.

Version **v1.6.0**

Date 2018-08-16

Description

"Added"

- Add `get_sensor_info` command gives `prod_line` info.

15 Troubleshooting

15.1 Sensor Homepage and HTTP Server

The sensor HTTP server page http://os-991900123456.local/ has information about the sensor system information, sensor status, firmware and configuration. To learn more about Web UI and it's use to troubleshoot the sensor, Please see the *Web Interface* portion of this user manual

Note: Please contact our Field Application Team and we can answer your questions and provide guidance for achieving proper operations.

15.2 Networking

Many initial problems with the sensor are associated with it not properly being assigned an IP address by a network switch or DHCP server on a client computer. Check your networking settings, the steps in *Connecting to Sensor*, and that all wires are firmly connected if you suspect this problem. Note that if the sensor is not connected via gigabit Ethernet, it will stop sending data and will output an error code if it fails to achieve a 1000 Mb/s+ full duplex link. Please see the *Networking Guide* for detailed guidance on network setup.

15.3 Get Alerts

To check for hardware errors, use the `get_alerts` TCP command.

If the watchdog is triggered, an alert code will be appended to the end of the response of the TCP command `get_alerts`. The sensor has a limited-size buffer that will record the first few alerts detected by the sensor.

The full list of possible alerts and error messages can be found in *Alerts and Errors* in the Appendix.

The alerts reported have the following format:

```
{
   "category": "Category of the alert: e.g. OVERTEMP, UDP_TRANSMISSION",
   "level": "Level of alert: e.g. NOTICE, WARNING, ERROR",
   "realtime": "The timestamp of the alert in nanoseconds",
   "active": "Whether the alert is active or not: <true/false>",
   "msg": "A description of the alert",
   "cursor": "The sequential number of the alert, starting from 0 counting up",
   "id": "The hexadecimal identification code of the alert: e.g. 0x01000017",
   "msg_verbose": "Any additional verbose description that the alert may present"
}
```

Example showing active and logged forced temperature sensor failures occurring at timestamps 1569712873477772800, 1569712879991844096, 1569712884968876544 (nanoseconds). The first logged error then resolves itself at 1569713260229536000. The example has been JSON formatted:

```
{
   "active": [
      {
         "category": "OVERTEMP",
         "level": "ERROR",
         "realtime": "1569712879991844096",
         "active": true,
         "msg": "Unit internal temperature out of bounds; please ensure proper heat sinking.",
         "cursor": 1,
         "id": "0x01000001",
         "msg_verbose": ""
      },
      {
         "category": "OVERTEMP",
         "level": "ERROR",
         "realtime": "1569712884968876544",
         "active": true,
         "msg": "Unit internal temperature out of bounds; please ensure proper heat sinking.",
         "cursor": 2,
         "id": "0x01000002",
         "msg_verbose": ""
      }
   ],
   "next_cursor": 4,
   "log": [
      {
         "category": "OVERTEMP",
         "level": "ERROR",
         "realtime": "1569712873477772800",
         "active": true,
         "msg": "Unit internal temperature out of bounds; please ensure proper heat sinking.",
         "cursor": 0,
         "id": "0x01000000",
         "msg_verbose": ""
      },
      {
         "category": "OVERTEMP",
         "level": "ERROR",
         "realtime": "1569712879991844096",
         "active": true,
         "msg": "Unit internal temperature out of bounds; please ensure proper heat sinking.",
         "cursor": 1,
         "id": "0x01000001",
         "msg_verbose": ""
      },
      {
         "category": "OVERTEMP",
         "level": "ERROR",
         "realtime": "1569712884968876544",
         "active": true,
         "msg": "Unit internal temperature out of bounds; please ensure proper heat sinking.",
         "cursor":2 ,
         "id": "0x01000002",
         "msg_verbose": ""
      },
```

(continues on next page)

(continued from previous page)

```
      {
         "category": "OVERTEMP",
         "level": "ERROR",
         "realtime": "1569713260229536000",
         "active": false,
         "msg": "Unit internal temperature out of bounds; please ensure proper heat sinking.",
         "cursor": 3,
         "id": "0x01000000",
         "msg_verbose": ""
      }
   ]
}
```

Note: Please contact our Field Application Team and we can answer your questions and provide guidance for achieving proper operations.

15.4 Using Latest Firmware

Upgrading to the latest firmware can often resolve issues found in earlier firmware. The latest firmware is always found at Ouster Downloads. Our Support team is best suited to be able to help you if you are running the latest firmware. Please refer to the *Updating Firmware* section to learn more on how to update firmware.

16 Alerts and Errors

The sensor provides alerts and error messages that can be used to help diagnose the sensor. These Alerts are accessible through the diagnostics tab on the sensor homepage. Alternately, users can query the sensor via the `get_alerts` *TCP command* and the *HTTP endpoint* **GET /api/v1/sensor/alerts**.

An alert would get triggered when it's trigger condition is met and would get cleared when the respective trigger condition no longer exists within a certain level of heuristics, or when a specific alert clearing condition is met.

`get_alerts` returns two lists, an **active** list and a **log** list. The **active** list will contain alert-trigger events for alerts that are currently active. An alert-trigger event will by-definition always have its active attribute set to true. There is no limit on the number of alert-trigger events that are displayed in the **active** event list. All currently active alert-trigger events will be displayed in the **active** event list.

The **log** list will contain all current and past alert-trigger and alert-clear events. An alert-clear event will by-definition have the exact same attributes and attribute values as its corresponding trigger event, with the exception of the realtime and cursor attributes which should have higher values, since an alert-clear event will always be received after an alert-trigger event. The **log** list has a length limit of 32 events with the oldest events automatically removed from the log list once a new event needs to be added to the **log** list and the **log** list length limit is reached, essentially acting as a FIFO (First In First Out) queue.

In addition to the **active** and **log** lists, `get_alerts` also returns a **next_cursor** field. Every alert event has a cursor attribute, which increments for every alert event logged. This can be used to track the alert activity that has been viewed and reduce message bandwidth. To do this, users are recommended to save the **next_cursor** field when calling `get_alerts` and then use that value as the `START_CURSOR` argument on the next `get_alerts` call to fetch only new **log** entries.

A valid value for **MODE** is either `summary` or `default`.

Note: Valid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=1

  ```
  get_alerts 1
  ```

- Example: Calling get_alerts with START_CURSOR=2 and MODE=summary

  ```
  get_alerts 2 summary
  ```

Invalid uses of get_alerts:

- Example: Calling get_alerts with START_CURSOR=summary and MODE=2

  ```
  get_alerts summary 2
  ```

Note: The order in which the START_CURSOR and MODE arguments are specified for the get_alerts TCP command must be followed when providing both arguments, with START_CURSOR preceding the

MODE. The arguments can be specified in any order when calling the equivalent /api/v1/sensor/alerts HTTP endpoint.

16.1 Table of All Alerts and Errors

Possible alerts and errors that the sensor can provide are listed below. Where appropriate, the message from the sensor aims to help the user diagnose and fix the issue themselves.

Note: Please note that if the recommended action does not clear the ALERT and the issue persists, Users are encouraged to update to the latest FW version. If that does not mitigate the issue, please collect the diagnostics file from the sensor Web UI and contact Ouster Support.

Table16.1: Alerts and Errors

ID	Category	Level	Alert Message	Sensor Action
0x01000000	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000001	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000002	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000003	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000004	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000005	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000006	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000007	UNDERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000008	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x01000009	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x0100000A	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x0100000B	OVERTEMP	Error	Unit internal temperature too high; Unit is shutting down. Please refer to Thermal integration guide for heat sinking requirements.	SHUTDOWN
0x0100000C	INTERNAL_COMM	Warning	Unit has experienced an internal COMM warning. If the issue persists, update to the latest FW. Contact Ouster support if the above steps do not resolve the alert with Diagnostic file.	No Action

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x0100000D	INTERNAL_COMM	Warning	Unit has experienced an internal COMM warning. If the issue persists, update to the latest FW. Contact Ouster support if the above steps do not resolve the alert with Diagnostic file.	No Action
0x0100000E	SHOT_LIMITING	Notice	Temperature is high enough where shot limiting may be engaged; Please refer to Thermal integration guide for heat sinking requirements.	No Action
0x0100000F	SHOT_LIMITING	Warning	Shot limiting mode is active. Laser power is partially attenuated; Please refer to Thermal integration guide for heat sinking requirements.	No Action
0x01000010	INTERNAL_FW	Error	Unit has experienced an internal error; If the issue persists, update to the latest FW. Contact Ouster support with Diagnostic file, if the above steps do not resolve the alert.	No Action
0x01000011	ETHERNET_LINK_BAD	Warning	Sensor has detected an issue with the connected ethernet link. Please check the network setup including the network switch and harnessing can support 1 Gbps Ethernet. If you experience no issues with this Alert active, this alert can be ignored.	No Action
0x01000012	INTERNAL_COMM	Warning	Unit has experienced an internal COMM warning: some measurements may have been skipped. No action is required from the user to clear the Alert. If the issue persists on FW update, contact Ouster support with diagnostic file.	No Action

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000013	INTERNAL_COMM	Warning	Unit has experienced an internal COMM warning: some measurements may have been skipped. No action is required from the user to clear the Alert. If the issue persists on FW update, contact Ouster support with diagnostic file.	No Action
0x01000014	INTERNAL_COMM	Warning	Unit has experienced an internal COMM warning: some measurements may have been skipped. No action is required from the user to clear the Alert. If the issue persists on FW update, contact Ouster support with diagnostic file.	No Action
0x01000015	UDP_TRANSMISSION	Warning	Client machine announced it is not reachable on the provided lidar data port; check that udp_dest and udp_port_lidar configured on the sensor matches client IP and port.This Alert may occur on sensor startup, if the client is not listening at that time. If the issue persists, contact Ouster support.	No Action
0x01000016	UDP_TRANSMISSION	Warning	Could not send lidar data UDP packet to host; check that network is up and the destination is reachable.	No Action
0x01000017	UDP_TRANSMISSION	Warning	Received an unknown error when trying to send lidar data UDP packet; closing socket.	No Action
0x01000018	UDP_TRANSMISSION	Warning	Client machine announced it is not reachable on the provided IMU data port; check that udp_dest and udp_port_imu configured on the sensor matches client IP and port.	No Action

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000019	UDP_TRANSMISSION	Warning	Could not send IMU UDP packet to host; check that network is up and the destination is reachable.	No Action
0x0100001A	UDP_TRANSMISSION	Warning	Received an unknown error when trying to send IMU UDP packet; closing socket.	No Action
0x0100001B	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x0100001C	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x0100001D	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x0100001E	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x0100001F	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x01000020	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000021	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x01000022	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x01000023	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x01000024	STARTUP	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x01000025	INTERNAL_COMM	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x01000026	INTERNAL_COMM	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN
0x01000027	INTERNAL_COMM	Error	Unit has experienced a startup error; Unit is shutting down. Update the Firmware to the latest version, if the issue persists please contact Ouster support.	SHUTDOWN

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000028	STARTUP	Warning	Unit has experienced an internal warning during startup and is restarting.	RESTART
0x01000029	STARTUP	Warning	Unit has experienced an internal warning during startup and is restarting.	RESTART
0x0100002A	STARTUP	Warning	Unit has experienced an internal warning during startup and is restarting.	RESTART
0x0100002B	STARTUP	Warning	Unit has experienced an internal warning during startup and is restarting.	RESTART
0x0100002C	STARTUP	Warning	Unit has experienced an internal warning during startup and is restarting.	RESTART
0x0100002D	STARTUP	Warning	Unit has experienced an internal warning during startup and is restarting.	RESTART
0x0100002E	INPUT_VOLTAGE	Warning	Input voltage is close to being too low. Consult the hardware user manual for voltage requirements. Raise voltage immediately.	No Action
0x0100002F	INPUT_VOLTAGE	Error	Input voltage is too low. Unit may shut down if the voltage drops farther. Consult the hardware user manual for voltage requirements.	SHUTDOWN
0x01000030	INPUT_VOLTAGE	Warning	Input voltage is close to being too high. Consult the hardware user manual for voltage requirements. Lower voltage immediately.	No Action
0x01000031	INPUT_VOLTAGE	Error	Input voltage is too high. Unit may shut down if the voltage increases farther. Consult the hardware user manual for voltage requirements.	SHUTDOWN
0x01000032	UDP_CONNECT	Warning	Couldn't open lidar UDP socket; please contact Ouster support.	No Action

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000033	UDP_CONNECT	Warning	Couldn't resolve hostname using DNS for lidar data; check network, DNS server, and udp_dest. If using static IP override, try setting udp_dest to an IP address or via auto-setting.	No Action
0x01000034	UDP_CONNECT	Warning	Invalid UDP port number; check network and udp_port_lidar.	No Action
0x01000035	UDP_CONNECT	Warning	Couldn't reach destination client for lidar data; verify cabling, network address configuration, and subnet mask if using static IP override	No Action
0x01000036	UDP_CONNECT	Warning	Couldn't open imu UDP socket; please contact Ouster support.	No Action
0x01000037	UDP_CONNECT	Warning	Couldn't resolve hostname using DNS for IMU data; check network, DNS server, and udp_dest. If using static IP override, try setting udp_dest to an IP address or via auto-setting.	No Action
0x01000038	UDP_CONNECT	Warning	Invalid UDP port number; check network and udp_port_imu.	No Action
0x01000039	UDP_CONNECT	Warning	Couldn't reach destination client for IMU data; verify cabling, network address configuration, and subnet mask if using static IP override	No Action
0x0100003A	SHOT_LIMITING	Warning	Shot limiting mode at maximum. Sensor shutdown imminent.	No Action

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x0100003B	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is in Error Stopped (Shutdown) state. Please contact Ouster support.	SHUTDOWN
0x0100003C	INTERNAL_FAULT	Error	Internal fault detected; Unit will restart to attempt recovery. If the issue persists, update Firmware to the latest version. Contact Ouster support if the above steps don't resolve the Alert.	RESTART
0x0100003D	INTERNAL_FAULT	Error	Internal fault detected; unit will restart to attempt recovery. If the issue persists, update Firmware to the latest version. Contact Ouster support if the above steps don't resolve the Alert.	RESTART
0x0100003E	INTERNAL_FAULT	Error	Internal fault detected; unit will restart to attempt recovery. If the issue persists, update Firmware to the latest version. Contact Ouster support if the above steps don't resolve the Alert.	RESTART
0x0100003F	INTERNAL_COMM	Error	Unit has experienced an internal COMM error; Unit is in Error Stopped(Shutdown) state. Please contact Ouster support.	SHUTDOWN
0x01000040	INTERNAL_FAULT	Error	Unit has experienced an internal COMM error; Unit is in Error Stopped(Shutdown) state. Please contact Ouster support.	SHUTDOWN
0x01000041	INTERNAL_COMM	Warning	Unit has experienced an internal COMM warning: some measurements may have been skipped.	No Action
0x01000042	INTERNAL_COMM	Error	Unit has experienced an internal COMM error; please contact Ouster support.	No Action

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000043	INTERNAL_FW	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x01000044	INTERNAL_FW	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x01000045	INTERNAL_FW	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x01000046	INTERNAL_FW	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x01000047	INTERNAL_FW	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x01000048	INTERNAL_FW	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x01000049	INTERNAL_FW	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x0100004A	STARTUP	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x0100004B	STARTUP	Error	Unit has experienced a startup error; please contact Ouster support.	SHUTDOWN
0x0100004C	INTERNAL_FAULT	Error	Internal fault detected; unit going to error stop state.	SHUTDOWN
0x0100004D	INTERNAL_FAULT	Error	Internal fault detected; unit going to error stop state.	SHUTDOWN
0x0100004E	WARMUP_ISSUE	Warning	Sensor warmup process is taking longer than expected; please ensure sensor is thermally constrained per requirements.	RESTART

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x0100004F	WARMUP_ISSUE	Warning	Sensor warmup process is taking longer than expected; please ensure sensor is thermally constrained per requirements.	RESTART
0x01000050	MOTOR_CONTROL	Warning	The phase lock offset error has exceeded the threshold. Check that the time source is accurate and the reduce the movement of the sensor including mechanical movement, shock, or vibration.	No Action
0x01000051	MOTOR_CONTROL	Error	The phase lock control failed to achieve a lock multiple times; Check that the time source is accurate.	No Action
0x01000052	CONFIG_INVALID	Error	Configuration value is invalid or out of bounds. Unit is shutting down. Try resetting the sensor configuration.	SHUTDOWN
0x01000053	WARMUP_ISSUE	Error	Sensor warmup process has failed. Unit is shutting down. Check the sensor operating conditions are within operating bounds.	SHUTDOWN
0x01000054	INTERNAL_FAULT	Notice	Unexpected hardware configuration detected. Please contact Ouster support.	No Action
0x01000055	UDP_TRANSMISSION	Warning	Unit has experienced a packet drop rate above normal threshold. Please check that the network has at least 1000 Mbps connection. Common causes of this notice may be 100 or 10 Mbps network connections.	No Action
0x01000056	INTERNAL_FAULT	Error	Internal fault detected; unit will restart to attempt recovery.	RESTART
0x01000057	OVERTEMP	Warning	Sensor temperature is too high. Sensor could have degraded range performance.	No Action

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000058	OVERTEMP	Error	Sensor temperature is too high; unit going to error stop state (Shutdown).	SHUTDOWN
0x01000059	INTERNAL_FAULT	Warning	Internal fault detected; unit will restart to attempt recovery.	RESTART
0x0100005A	INTERNAL_FAULT	Warning	Unit has experienced an internal COMM warning: some measurements may have been skipped. Check sensor operating conditions.	No Action
0x0100005B	INTERNAL_FAULT	Warning	Unit has experienced an internal COMM warning: some measurements may have been skipped. Check sensor operating conditions.	No Action
0x0100005C	INTERNAL_FAULT	Warning	Unit has experienced an internal COMM warning: some measurements may have been skipped. Check sensor operating conditions.	No Action
0x0100005D	INTERNAL_FAULT	Warning	Internal fault detected; unit going to error stop state.	SHUTDOWN
0x0100005E	INTERNAL_FAULT	Warning	Unit has experienced an overcurrent event; unit will restart to attempt recovery.	RESTART
0x0100005F	IO_CONNECTION	Warning	Unit has stopped receiving SYNC_PULSE_IN signals and is configured to expect them. Check electrical inputs to sensor.	No Action
0x01000060	IO_CONNECTION	Warning	Unit has stopped receiving NMEA messages at the MULTIPURPOSE_IO port and is configured to expect them. Check electrical inputs to sensor.	No Action
0x01000061	INTERNAL_COMM	Error	Unit has experienced an internal COMM error Unit will restart to attempt recovery.	RESTART

continues on next page

Table 16.1 - continued from previous page

ID	Category	Level	Alert Message	Sensor Action
0x01000062	INTERNAL_FAULT	Error	Unit has experienced a internal error; Unit is in Error stopped state. Please contact Ouster support.	No Action
0x01000063	MOTOR_CONTROL	Warning	Unit is spinning outside of tolerant range; Check sensor operating conditions.	No Action
0x01000064	MOTOR_CONTROL	Error	Unit failed to maintain target spin rate; please contact Ouster support.	No Action
0x01000065	MOTOR_CONTROL	Error	Unit has experienced a internal error; Unit is in Error stopped state. Please contact Ouster support.	No Action
0x01000066	MOTOR_CONTROL	Error	Unit has experienced a startup error; Unit is in Error stopped state. Please contact Ouster support.	SHUTDOWN
0x01000067	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is in Error stopped state. Please contact Ouster support.	SHUTDOWN
0x01000068	INTERNAL_FW	Error	Unit has experienced a startup error; Unit is in Error stopped state. Please contact Ouster support.	SHUTDOWN

17 Networking Guide

This guide will help you understand how to quickly get connected to your sensor to start doing great things with it. When trying to connect to the sensor for the first time there are some basics that need to be achieved for successful communication between the host machine and the sensor.

We need to ensure that the sensor receives an IP address from the host machine so that we can talk to it. This can be achieved with a few different methods such as DHCP, link-local, static IP. We also need to ensure that the sensor and the host machine are talking on the same subnet.

Once the sensor receives an IP address and is on the correct subnet we can talk to it using its hostname, `os-991234567890.local`, where `991234567890` is the sensor serial number. The sensor serial number can be found on a sticker affixed to the top of the sensor.

Based on the platform being used the user can refer to the following:

- *Windows*
- *macOS*
- *Linux*

17.1 Networking Terminology

If some of this terminology is new to you don't fret, we have defined some of it for you. Here is some basic terminology that will help you digest the steps and be more familiar with networking in general.

IPv4 Address This is the address that can be used to communicate with devices on a network. The format of an IPv4 address is a set of four octets, `xxx.xxx.xxx.xxx` with `xxx` being in the range `0-255`. For example, your host machine Ethernet port may have an address of `192.0.2.1` and your sensor may have an address of `192.0.2.130`.

DHCP (Dynamic Host Configuration Protocol) Server This is a server that may run on your host machine, switch, or router which will serve an IPv4 address to a device that is connected to it. It will ensure that each device connected will have a unique IPv4 address on the network.

Link-local IPv4 Address These are the addresses that are self-assigned between the host machine and a device connected to it in the absence of a DHCP server. They are only valid within the network segment that the host is connected to. The addresses lie within the block `169.254.0.0/16 (169.254.0.0 - 169.254.255.255)`.

Subnet Mask This defines which bits of the IPv4 address are the network prefix and which are the host identifiers. See the table below for an example.

	Binary Form	**Decimal-dot notation**
IP address	`11000000.00000000.00000010.10000010`	`192.0.2.130`
Subnet mask	`11111111.11111111.11111111.00000000`	`255.255.255.0`
Network prefix	`11000000.00000000.00000010.00000000`	`192.0.2.0`
Host identifier	`00000000.00000000.00000000.10000010`	`0.0.0.130`

Note: Subnet mask can be abbreviated with the number of bits that apply to the network prefix. E.g. /24 for 255.255.255.0 or /16 for 255.255.0.0.

Static IPv4 Address This is when you specify the addresses for the host machine and/or connected device rather than letting the host machine self-assign or using a DHCP server. For example, you may want to specify the host machine IPv4 address to be `192.0.2.100/24` and the sensor to be `192.0.2.200`.

Hostname This is the more human readable name that comes with your sensor. The sensor's hostname is `os-991234567890.local`, where `991234567890` is the sensor serial number.

Note: The `.local` portion of the hostname denotes the local domain used in combination with multicast DNS (mDNS). It is employed when using the sensor in a local network environment with supporting operating system services. This means when the sensor is directly connected to the host machine or if the host machine and sensor are on the same network connected through a router or switch. If you are trying to connect to the sensor on another domain with a supporting DHCP and DNS server configuration you should replace the .local with the domain the sensor is on. For example, if the sensor is connected to a network with domain `ouster-domain.com` the sensor will be reachable on `os-991234567890.ouster-domain.com`.

17.2 Windows

The following steps have been tested on Windows 10. The sensor's hostname is `os-991234567890.local`, where `991234567890` is the sensor serial number. The sensor serial number can be found on a sticker affixed to the top of the sensor.

17.2.1 Connecting the Sensor

1. Connect the sensor to an available Ethernet port on your host machine or router.
2. The sensor will automatically obtain an IP address either through link-local or DHCP (if preconfigured) depending on your network configuration.

Note: It can take up to 60 seconds to obtain an IP address from the initial power-up of the sensor.

17.2.2 The Sensor Homepage

1. Type `os-991234567890.local`/ in the address bar of your browser to view the sensor homepage

Note: If you are unable to load the sensor homepage, follow the steps in *Determining the IPv4 Address of the Sensor* to verify your sensor is on the network and has a valid IPv4 address.

17.2.3 Determining the IPv4 Address of the Sensor

1. Open a command prompt on the host machine by pressing **Win+X** and then **A**
2. Use the ping command to determine the IPv4 address of the sensor

Command

```
ping -4 [sensor_hostname]
```

Example

```
C:\\WINDOWS\\system32>ping -4 |os-sn|
```

Note: If this command hangs you may need to go back and configure your interface to link-local in the section *Connecting the Sensor*

Response

```
Pinging |os-sn| [|sensor-ip|] with 32 bytes of data:
Reply from |sensor-ip|: bytes=32 time<1ms TTL=64
Reply from |sensor-ip|: bytes=32 time<1ms TTL=64
Reply from |sensor-ip|: bytes=32 time<1ms TTL=64
Reply from |sensor-ip|: bytes=32 time<1ms TTL=64

Ping statistics for |sensor-ip|:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`. If your sensor IPv4 address is of the form `169.254.x.x` it is connected via link-local.

3. You can also browse for the sensor IPv4 address using dns-sd and the sensor hostname. Learn more about this in *Finding a Sensor with mDNS Service Discovery*

Command

```
dns-sd -G v4 [sensor_hostname]
```

Example

```
C:\\WINDOW\\system32>dns-sd -G v4 |os-sn|
```

Response

```
Timestamp     A/R  Flags   if   Hostname                Address        TTL
14:22:46.897  Add  2       6    |os-sn|  |sensor-ip|   120
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`. If your sensor IPv4 address is of the form `169.254.x.x` it is connected via link-local.

17.2.4 Determining the IPv4 Address of the Interface

1. Open a command prompt by pressing **Win+X** and then **A**
2. View the IPv4 address of your interfaces

Command

```
netsh interface ip show config
```

Example

```
C:\\WINDOWS\\system32>netsh interface ip show config
```

Response

```
Configuration for interface "Local Area Connection"
    DHCP enabled:                         Yes
    IP Address:                           |interface-ip|
    Subnet Prefix:                        169.254.0.0/16 (mask 255.255.0.0)
    InterfaceMetric:                      25
    DNS servers configured through DHCP:  None
    Register with which suffix:           Primary only
    WINS servers configured through DHCP: None

Configuration for interface "Loopback Pseudo-Interface 1"
    DHCP enabled:                         No
    IP Address:                           127.0.0.1
    Subnet Prefix:                        127.0.0.0/8 (mask 255.0.0.0)
    InterfaceMetric:                      75
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None
```

- In this example, your sensor is plugged into interface "Local Area Connection"
- Your host IPv4 address will be on the line that starts with IP Address: In this case it is `169.254.0.1`

Note: If your interface IPv4 address is of the form `169.254.x.x`, it is connected via link-local to the sensor. This means that Windows self-assigned an IP address in the absence of a DHCP server.

17.2.5 Setting the Host Interface to DHCP

Use this to set your interface to automatically obtain an IP address via DHCP. This is useful for architectures that need to be more plug and play.

Set your interface to DHCP.

Command

```
netsh interface ip set address ["Network Interface Name"] dhcp
```

Example

with interface name `"Local Area Connection"`

```
C:\\WINDOWS\\system32>netsh interface ip set address "Local Area Connection" dhcp
```

Response

```
blank
```

17.2.6 Setting the Host Interface to Static IP

Use this to set your interface to be assigned a static IPv4 address. This is useful for controlling the IP address that the sensor will be sending data to.

Set your interface to static.

Command

```
netsh interface ip set address name="Network Interface Name" static [IP address] [Subnet Mask]
[Gateway]
```

Example

with interface name "Local Area Connection" and IPv4 address 192.0.2.1/24.

```
C:\\WINDOWS\\system32>netsh interface ip set address name="Local Area Connection"
 static 192.0.2.1/24
```

Note: The `/24` is shorthand for Subnet Mask = `255.255.255.0`

Response

```
blank
```

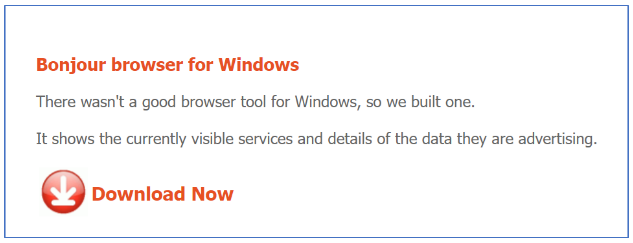
17.2.7 Finding a Sensor with mDNS Service Discovery

The sensor announces its presence on the network using Multicast Domain Name Service (mDNS) with a service type named `_roger._tcp`. You can use service discovery tools such as Bonjour browser (Windows) to find all sensors connected to the network.

Note: Click Bonjour to install Bonjour Browser.

Example using Bonjour Browser:

Step 1: User can download the Bonjour Browser

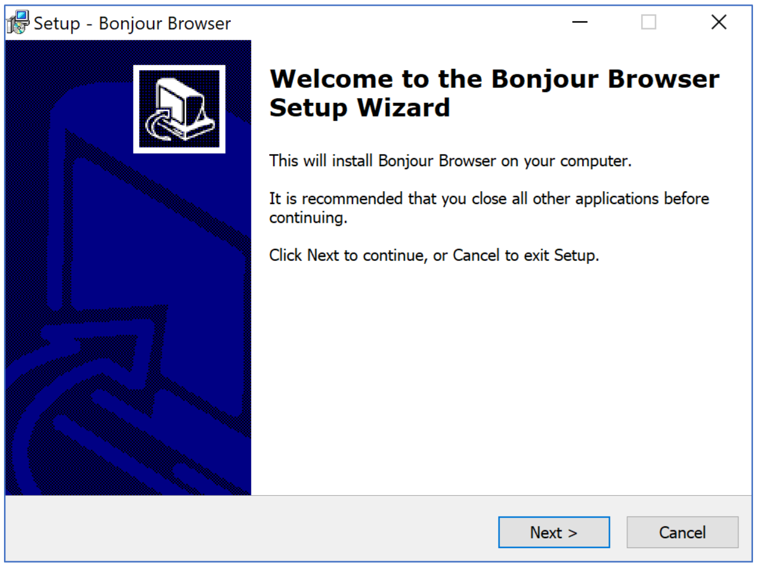


Bonjour browser for Windows

There wasn't a good browser tool for Windows, so we built one.

It shows the currently visible services and details of the data they are advertising.

Figure 17.1: Downloading Application



Welcome to the Bonjour Browser Setup Wizard

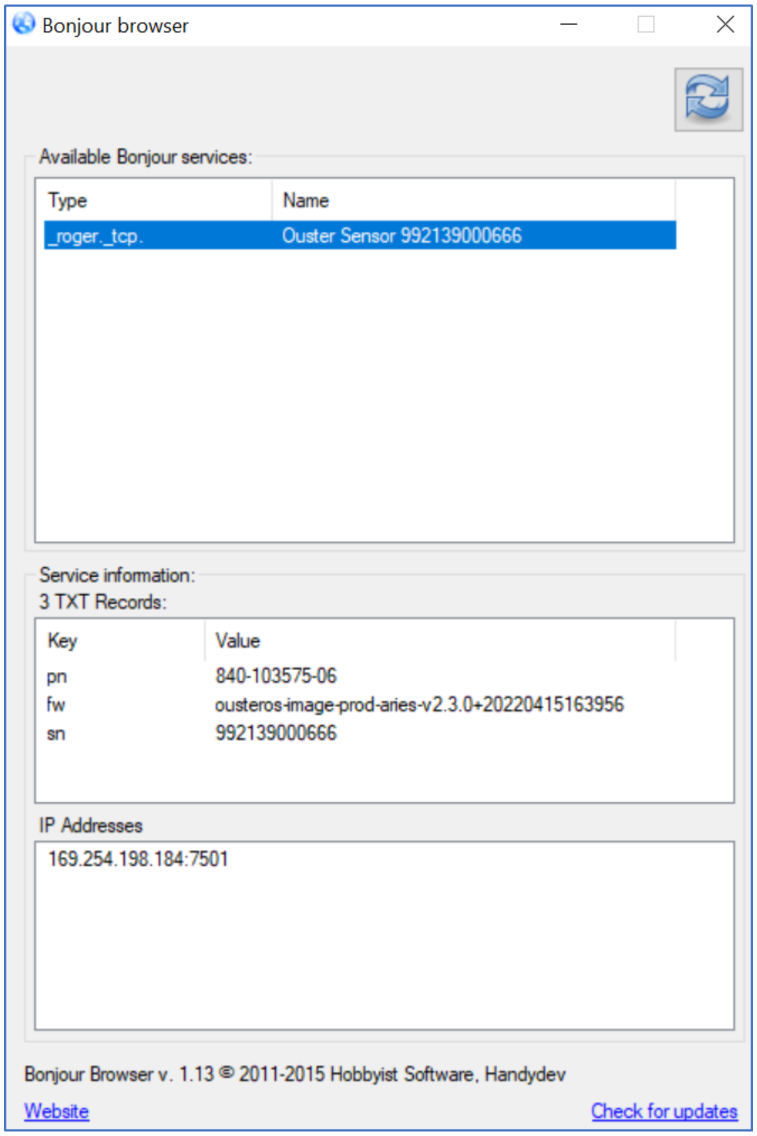
This will install Bonjour Browser on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Figure 17.2: Software Setup and Installation

Step 2: Sensor announces its presence on the network using Multicast Domain Name Service (mDNS) with a service type named `_roger._tcp`. Click on this to get all the information required.



Available Bonjour services:

Type	Name
_roger._tcp.	Ouster Sensor 992139000666

Service information:
3 TXT Records:

Key	Value
pn	840-103575-06
fw	ousteros-image-prod-aries-v2.3.0+20220415163956
sn	992139000666

IP Addresses

169.254.198.184:7501

Bonjour Browser v. 1.13 © 2011-2015 Hobbyist Software, Handydev

Figure 17.3: `_roger._tcp`

17.3 macOS

The following steps have been tested on macOS 10.15.4. The sensor's hostname is `os-991234567890.local`, where `991234567890` is the sensor serial number. The sensor serial number can be found on a sticker affixed to the top of the sensor.

17.3.1 Connecting the Sensor

1. Connect the sensor to an available Ethernet port on your host machine or router.
2. The sensor will automatically obtain an IP address either through link-local or DHCP (if preconfigured) depending on your network configuration.

Note: It can take up to 60 seconds to obtain an IP address from the initial power-up of the sensor.

17.3.2 The Sensor Homepage

1. Type `os-991234567890.local` in the address bar of your browser to view the sensor homepage

Note: If you are unable to load the sensor homepage, follow the steps in *Determining the IPv4 Address of the Sensor* to verify your sensor is on the network and has a valid IPv4 address.

17.3.3 Determining the IPv4 Address of the Sensor

1. Open a Terminal window on the host machine by pressing **CMD+SPACE** and typing **Terminal** in the search bar, then press enter.
2. Use the ping command to determine the IPv4 address of the sensor

Command

```
ping -c3 [sensor_hostname]
```

Example

```
Mac-Computer:~ username$ ping -c3 |os-sn|
```

Note: If this command hangs you may need to go back and configure your interface to link-local in the section *Connecting the Sensor*

Response

```
PING |os-sn| (|sensor-ip|): 56 data bytes
64 bytes from |sensor-ip|: icmp_seq=0 ttl=64 time=0.644 ms
64 bytes from |sensor-ip|: icmp_seq=1 ttl=64 time=0.617 ms
```

(continues on next page)

(continued from previous page)

```
64 bytes from |sensor-ip|: icmp_seq=2 ttl=64 time=0.299 ms

--- |os-sn| ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.299/0.520/0.644/0.157 ms
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`. If your sensor IPv4 address is of the form `169.254.x.x` it is connected via link-local.

3. You can also browse for the sensor IPv4 address using dns-sd and the sensor hostname. Learn more about this in *Finding a Sensor*

Command

```
dns-sd -G v4 [sensor_hostname]
```

Example

```
Mac-Computer:~ username$ dns-sd -G v4 |os-sn|
```

Response

```
DATE: ---Tue 28 Apr 2020---
11:40:43.228  ...STARTING...
Timestamp     A/R  Flags  if  Hostname                Address          TTL
11:40:43.414  Add  2      18  |os-sn|.  |sensor-ip|     120
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`. If your sensor IPv4 address is of the form `169.254.x.x` it is connected via link-local.

17.3.4 Determining the IPv4 Address of the Interface

This will help you find the IPv4 address of the interface that you have plugged the sensor into. It is helpful to know which interface you have plugged into, e.g. `en1` in the example below.

1. Open a Terminal window on the host machine by pressing **CMD+SPACE** and typing **Terminal** in the search bar, then press enter.
2. View the IPv4 address of your interfaces

Command

```
ifconfig
```

Example

```
Mac-Computer:~ username$ ifconfig
```

Response

```
   lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
      options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
      inet 127.0.0.1 netmask 0xff000000
      inet6 ::1 prefixlen 128
      inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
      nd6 options=201<PERFORMNUD,DAD>
   en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      options=400<CHANNEL_IO>
      ether 38:f9:d3:d6:33:8a
      inet6 fe80::1c30:1246:93a2:9f68%en0 prefixlen 64 secured scopeid 0x7
      inet 192.0.2.7 netmask 0xffffff00 broadcast 192.0.2.255
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect
      status: active
   en1: flags=8963<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      options=400<CHANNEL_IO>
      ether 48:65:ee:1d:22:35
      inet6 fe80::c27:1917:47ed:bcfe%en1 prefixlen 64 secured scopeid 0x12
      inet |interface-ip| netmask 0xffff0000 broadcast 169.254.255.255
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect (1000baseT <full-duplex>)
      status: active


* In this example, your sensor is plugged into interface ``en1``
* Your host IPv4 address will be on the line that starts with ``inet``: In this case it is |interface-ip|
```

Note: If your interface IPv4 address is of the form `169.254.x.x`, it is connected via link-local to the sensor. This means that the macOS self-assigned an IP address in the absence of a DHCP server.

17.3.5 Setting the Host Interface to DHCP

Use this to set your interface to automatically obtain an IP address via DHCP. This is useful for architectures that need to be more plug and play.

Set your interface to DHCP

Command

```
sudo ipconfig set [interface_name] DHCP
```

Example

with interface name en1

```
Mac-Computer:~ username$ sudo ipconfig set en1 DHCP
```

Response

```
blank

Note: However you can verify the change has been made with the ``ifconfig`` command.
      The ``inet`` line will be blank if nothing is plugged in or shows the DHCP or
      link-local self-assigned IPv4 address. E.g. |interface-ip|


en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500

options=6407<RXCSUM,TXCSUM,VLAN_MTU,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
   ether 48:65:ee:1d:22:35
   inet6 fe80::1c24:5e0a:2ea8:12e9%en1 prefixlen 64 secured scopeid 0x7
   inet |interface-ip| netmask 0xffff0000 broadcast 169.254.255.255
   nd6 options=201<PERFORMNUD,DAD>
   media: autoselect (1000baseT <full-duplex>)
   status: active
```

17.3.6 Setting the Host Interface to Static IP

Use this to set your interface to be assigned a static IPv4 address. This is useful for controlling the IP address that the sensor will be sending data to.

Set your interface to static

Command

```
sudo ipconfig set [interface_name] MANUAL [ip_address] [subnet_mask]
```

Example

with interface name en1 and IPv4 address 192.0.2.1 and subnet mask 255.255.255.0.

```
Mac-Computer:~ username$ sudo ipconfig set en1 MANUAL 192.0.2.1 255.255.255.0
```

Note: The /24 is shorthand for Subnet Mask = 255.255.255.0

Response

```
blank

Note: However you can verify the change has been made with the ``ifconfig`` command.
      The ``inet`` line will show the static IPv4 address. e.g. ``192.0.2.1``.
```

(continues on next page)

(continued from previous page)

```
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500

options=6407<RXCSUM,TXCSUM,VLAN_MTU,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
 ether 48:65:ee:1d:22:35
 inet6 fe80::1c24:5e0a:2ea8:12e9%en1 prefixlen 64 secured scopeid 0x7
 inet 192.0.2.1 netmask 0xffffff00 broadcast 192.0.2.255
 nd6 options=201<PERFORMNUD,DAD>
 media: autoselect (1000baseT <full-duplex>)
 status: active
```

17.3.7 Finding a Sensor

With mDNS Service Discovery:

The sensor announces its presence on the network using Multicast Domain Name Service (mDNS) with a service type named `_roger._tcp`. You can use service discovery tools such as dns-sd (Windows/macOS) to find all sensors connected to the network.

1. Find all sensors and their associated service text on a network.

Command

```
dns-sd -Z [service type]
```

Example

```
Mac-Computer:~ username$ dns-sd -Z _roger._tcp
```

Response

```
Browsing for _roger._tcp
DATE: ---Thu 30 Apr 2020---
17:27:52.242  ...STARTING...

; To direct clients to browse a different domain, substitute that domain in
 place of '@'
lb._dns-sd._udp                                 PTR     @

; In the list of services below, the SRV records will typically reference dot-local
 Multicast DNS names.
; When transferring this zone file data to your unicast DNS server, you'll need to
 replace those dot-local
; names with the correct fully-qualified (unicast) domain name of the target host
 offering the service.

_roger._tcp                                   PTR
Ouster Sensor |sn|._roger._tcp
Ouster Sensor |sn|._roger._tcp    SRV     0 0 7501 |os-sn|. ;
Replace with unicast FQDN of target host
Ouster Sensor |sn|._roger._tcp    TXT     "pn=840-102145-B" "sn= |sn|"
"fw=ousteros-image-prod-aries-v2.0.0-20200417193957" "sn= |sn|"
```

2. Browse for the sensor IPv4 address using dns-sd and the sensor hostname.

Command

```
dns-sd -G v4 [sensor_hostname]
```

Example

```
Mac-Computer:~ username$ dns-sd -G v4 |os-sn|
```

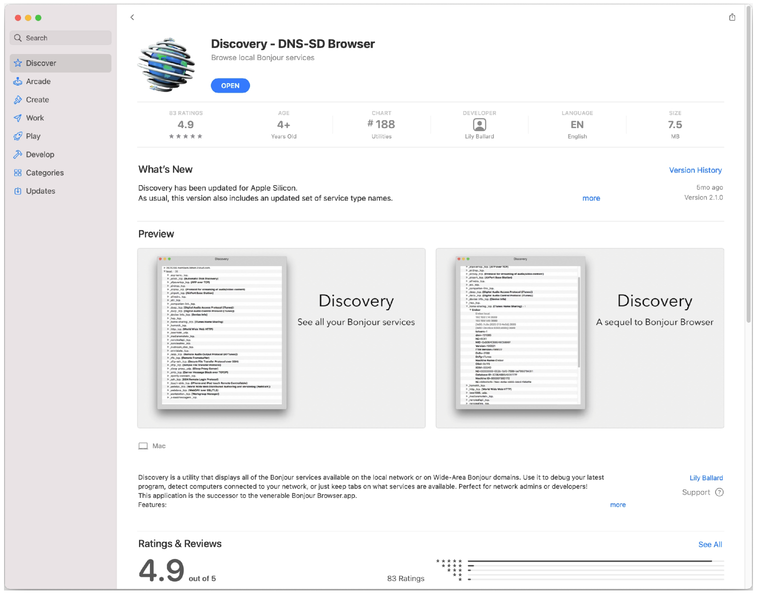
Response

```
DATE: ---Thu 30 Apr 2020---
17:37:33.155  ...STARTING...
Timestamp     A/R  Flags  if   Hostname                Address          TTL
17:37:33.379  Add  2       7   |os-sn|.  |sensor-ip|     120
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`

With Discovery App:

Step 1: User can download the Discovery DNS-SD



Discovery - DNS-SD Browser

Browse local Bonjour services

What's New

Discovery has been updated for Apple Silicon.
As usual, this version also includes an updated set of service type names.

5mo ago
Version 2.1.0

Preview

Discovery is a utility that displays all of the Bonjour services available on the local network or on Wide-Area Bonjour domains. Use it to debug your latest program, detect computers connected to your network, or just keep tabs on what services are available. Perfect for network admins or developers!
This application is the successor to the venerable Bonjour Browser.app.
Features:

Ratings & Reviews

Figure 17.4: Downloading Application

Step 2: Using finder, the user can search for *Discovery*

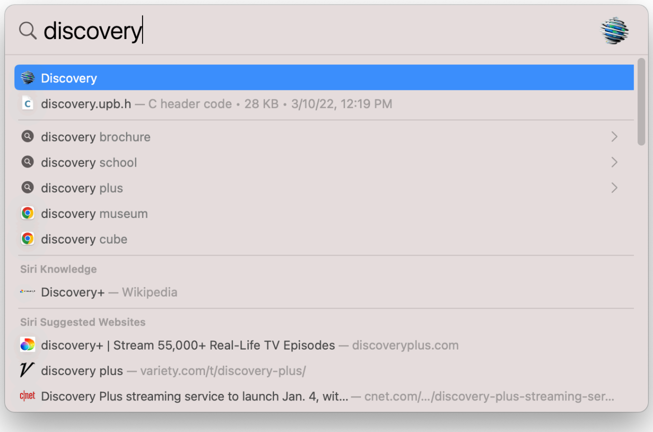

discovery
Discovery
discovery.upb.h — C header code • 28 KB • 3/10/22, 12:19 PM
discovery brochure
discovery school
discovery plus
discovery museum
discovery cube
Siri Knowledge
Discovery+ — Wikipedia
Siri Suggested Websites
discovery+ | Stream 55,000+ Real-Life TV Episodes — discoveryplus.com
discovery plus — variety.com/t/discovery-plus/
Discovery Plus streaming service to launch Jan. 4, wit... — cnet.com/.../discovery-plus-streaming-ser...

Figure 17.5: Finding the Application

Step 3: Sensor announces its presence on the network using Multicast Domain Name Service (mDNS) with a service type named `_roger._tcp`. Click on this to get all the information required.

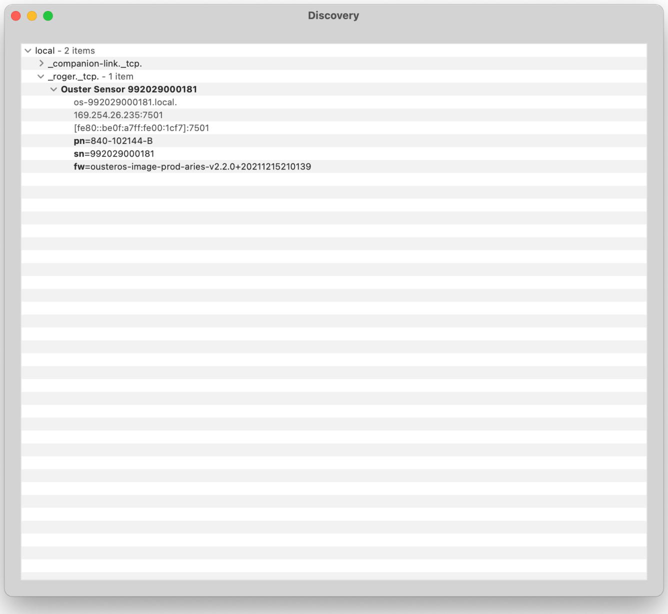
17.4 Linux

The following steps have been tested on Ubuntu 18.04 & 20.04.4 LTS. The sensor's hostname is `os-991234567890.local`, where `991234567890` is the sensor serial number. The sensor serial number can be found on a sticker affixed to the top of the sensor.

17.4.1 Connecting the Sensor

1. Connect the sensor to an available Ethernet port on your host machine or router.
2. The sensor will automatically obtain an IP address either through link-local or DHCP (if preconfigured) depending on your network configuration.
3. If directly connecting to the host machine you may need to set your Ethernet interface to `Link-Local Only` mode. This can be done via the command line or GUI. See instructions in *Setting the Interface to Link-Local Only*

Note: It can take up to 60 seconds to obtain an IP address from the initial power-up of the sensor.



Discovery

```
local - 2 items
  _companion-link._tcp.
  _roger._tcp. - 1 item
    Ouster Sensor 992029000181
      os-992029000181.local.
      169.254.26.235:7501
      [fe80::be0f:a7ff:fe00:1cf7]:7501
      pn=840-102144-B
      sn=992029000181
      fw=ousteros-image-prod-aries-v2.2.0+20211215210139
```

Figure 17.6: `_roger._tcp`

17.4.2 Setting the Interface to Link-Local Only

Via Command Line

Command

```
nmcli con modify [interface_name] ipv4.method link-local ipv4.addresses ""
```

Note: To identify the name of your connection, please use the command: `nmcli connection show`.

Example

with interface name `eth0` and IPv4 address `""`.

```
username@ubuntu:~$ nmcli con modify eth0 ipv4.method link-local ipv4.addresses ""
```

Response

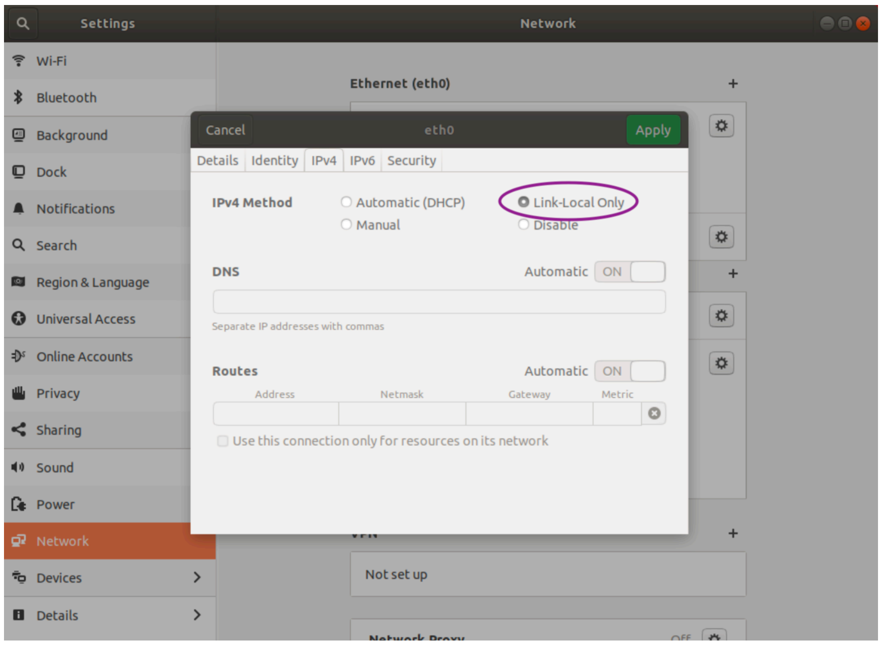
```
blank

Note: However you can verify the change has been made with the ``ip addr`` command.
      The ``inet`` line for the interface ``eth0`` will show the link-local IPv4 address automatically
      negotiated once the sensor is reconnected to the interface. e.g. |interface-ip|.

```

(continues on next page)

(continued from previous page)

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
       default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
       default qlen 1000
   link/ether 00:0c:29:2b:cc:48 brd ff:ff:ff:ff:ff:ff
   inet |interface-ip|/16 brd 169.254.255.255 scope link noprefixroute eth0
      valid_lft forever preferred_lft forever
   inet6 fe80::be9f:d2a4:4451:3dfe/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
   default qlen 1000
   link/ether 00:50:56:28:7a:8a brd ff:ff:ff:ff:ff:ff
   inet 172.16.79.232/24 brd 172.16.79.255 scope global wlan0
      valid_lft forever preferred_lft forever
   inet6 fe80::250:56ff:fe28:7a8a/64 scope link
      valid_lft forever preferred_lft forever
```

Via GUI: The image below illustrates how to set the interface to `Link-Local Only` mode using the graphical user interface.



IPv4 Method: Automatic (DHCP) [✓ Link-Local Only] Manual Disable

DNS Automatic ON

Separate IP addresses with commas

Routes Automatic ON

Address Netmask Gateway Metric

Use this connection only for resources on its network

Note: It can take up to 60 seconds to obtain an IP address from the initial power-up of the sensor.

17.4.3 The Sensor Homepage

1. Type `os-991234567890.local`/ in the address bar of your browser to view the sensor homepage

---

Note: If you are unable to load the sensor homepage, follow the steps in *Determining the IPv4 Address of the Sensor* to verify your sensor is on the network and has a valid IPv4 address.

17.4.4 Determining the IPv4 Address of the Sensor

1. Open a Terminal window on the host machine by pressing **Ctrl+Alt+T**.
2. Use the `ping` command to determine the IPv4 address of the sensor

Command

```
ping -4 -c3 [sensor_hostname]
```

Example

```
username@ubuntu:~$ ping -4 -c3 |os-sn|
```

Note: If this command hangs you may need to go back and configure your interface to link-local in the section *Setting the Interface to Link-Local Only*

Response

```
PING |os-sn| (|sensor-ip|) 56(84) bytes of data.
64 bytes from |os-sn| (|sensor-ip|): icmp_seq=1 ttl=64 time=1.56 ms
64 bytes from |os-sn| (|sensor-ip|): icmp_seq=2 ttl=64 time=0.893 ms
64 bytes from |os-sn| (|sensor-ip|): icmp_seq=3 ttl=64
time=0.568 ms

--- |os-sn| ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2025ms
rtt min/avg/max/mdev = 0.568/1.008/1.565/0.416 ms
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`. If your sensor IPv4 address is of the form `169.254.x.x` it is connected via link-local.

---

3. You can also browse for the sensor IPv4 address using `avahi-browse` and the sensor service type, which is `_roger._tcp`. Learn more about this in *Finding a Sensor with mDNS Service Discovery*

Command

```
avahi-browse -lrt [service type]
```

Example

```
username@ubuntu:~$ avahi-browse -lrt _roger._tcp
```

Response

```
+   eth0 IPv6 Ouster Sensor |sn|                    _roger._tcp          local
+   eth0 IPv4 Ouster Sensor |sn|                    _roger._tcp          local
=   eth0 IPv6 Ouster Sensor |sn|                    _roger._tcp          local
   hostname = [|os-sn|]
   address = [fe80::be0f:a7ff:fe00:1852]
   port = [7501]
   txt = ["fw=ousteros-image-prod-aries-v2.0.0-20200417193957" "sn=99201000067
8" "pn=840-102145-B"]
=   eth0 IPv4 Ouster Sensor |sn|                    _roger._tcp          local
   hostname = [|os-sn|]
   address = [|sensor-ip|]
   port = [7501]
   txt = ["fw=ousteros-image-prod-aries-v2.0.0-20200417193957" "sn= |sn|"
          "pn=840-102145-B"]
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`. If your sensor IPv4 address is of the form `169.254.x.x` it is connected via link-local.

17.4.5 Determining the IPv4 Address of the Interface

This will help you find the IPv4 address of the interface that you have plugged the sensor into. It is helpful to know which interface you have plugged into, e.g. `eth0` in the example below.

1. Open a Terminal window on the host machine by pressing **Ctrl+Alt+T**.
2. View the IPv4 address of your interfaces

Command

```
ip addr
```

Example

```
username@ubuntu:~$ ip addr
```

Response

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
       default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
```

(continues on next page)

(continued from previous page)

```
      default qlen 1000
    link/ether 00:0c:29:2b:cc:48 brd ff:ff:ff:ff:ff:ff
    inet |interface-ip|/16 brd 169.254.255.255 scope link noprefixroute eth0
      valid_lft forever preferred_lft forever
    inet6 fe80::be9f:d2a4:4451:3dfe/64 scope link noprefixroute
      valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
      default qlen 1000
    link/ether 00:50:56:28:7a:8a brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.232/24 brd 192.0.2.255 scope global wlan0
      valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe28:7a8a/64 scope link
      valid_lft forever preferred_lft forever
4: gpd0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group
      default qlen 500
    link/none
```

- In this example, your sensor is plugged into interface `eth0`.
- Your host IPv4 address will be on the line that starts with `inet`: In this case it is `169.254.0.1`.

Note: If your interface IPv4 address is of the form `169.254.x.x`, it is connected via link-local to the sensor. This means that the Linux self-assigned an IP address in the absence of a DHCP server.

17.4.6 Setting the Host Interface to DHCP

Use this to set your interface to automatically obtain an IP address via DHCP. This is useful for architectures that need to be more plug and play.

Note: It is recommended that you unplug the cable from the interface prior to making changes to the interface.

Via Command Line

Command

```
nmcli con modify [interface_name] ipv4.method auto ipv4.addresses ""
```

Example

with interface name `eth0`

```
username@ubuntu:~$ nmcli con modify eth0 ipv4.method auto ipv4.addresses ""
```

Response

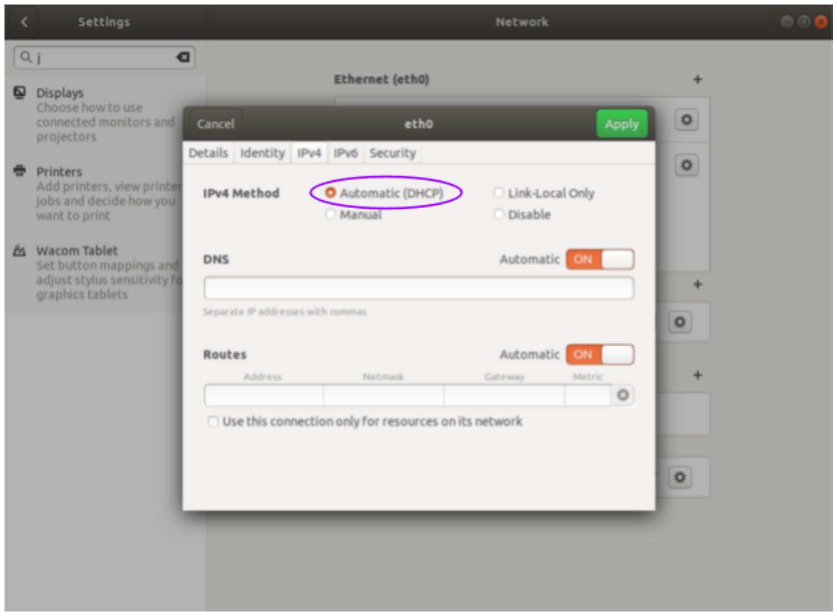
```
 blank
```

(continues on next page)

(continued from previous page)

```
 Note: However you can verify the change has been made with the ``ip addr`` command.
       There will be no ``inet`` line for the interface ``eth0`` until you plug in a cable
       to a device that has a DHCP server to provide an IPv4 address the interface

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
   default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
   valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
   default qlen 1000
   link/ether 00:0c:29:2b:cc:48 brd ff:ff:ff:ff:ff:ff
   inet6 fe80::be9f:d2a4:4451:3dfe/64 scope link noprefixroute
   valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
   default qlen 1000
   link/ether 00:50:56:28:7a:8a brd ff:ff:ff:ff:ff:ff
   inet 172.16.79.232/24 brd 172.16.79.255 scope global wlan0
   valid_lft forever preferred_lft forever
   inet6 fe80::250:56ff:fe28:7a8a/64 scope link
   valid_lft forever preferred_lft forever
```

Via GUI The image below illustrates how to set the interface to `Automatic (DHCP)` mode using the graphical user interface.


Settings
Network
Displays
Choose how to use connected monitors and projectors
Printers
Add printers, view printer jobs and decide how you want to print
Wacom Tablet
Set button mappings and adjust stylus sensitivity for graphics tablets
Ethernet (eth0)
Cancel
eth0
Apply
Details
Identity
IPv4
IPv6
Security
Address
Netmask
Gateway
Metric
Separate IP addresses with commas

IPv4 Method: [✓ Automatic (DHCP)] Manual Link-Local Only Disable

DNS: Automatic ON

Routes: Automatic ON

Use this connection only for resources on its network

17.4.7 Setting the Host Interface to Static IP

Use this to set your interface to be assigned a static IPv4 address. This is useful for controlling the IP address that the sensor will be sending data to.

Note: It is recommended that you unplug the cable from the interface prior to making changes to the interface.

Via Command Line

Command

```
nmcli con modify [interface_name] ipv4.method manual ipv4.addresses [ip_address]
```

Example

with interface name `eth0` and IPv4 address `192.0.2.1/24`.

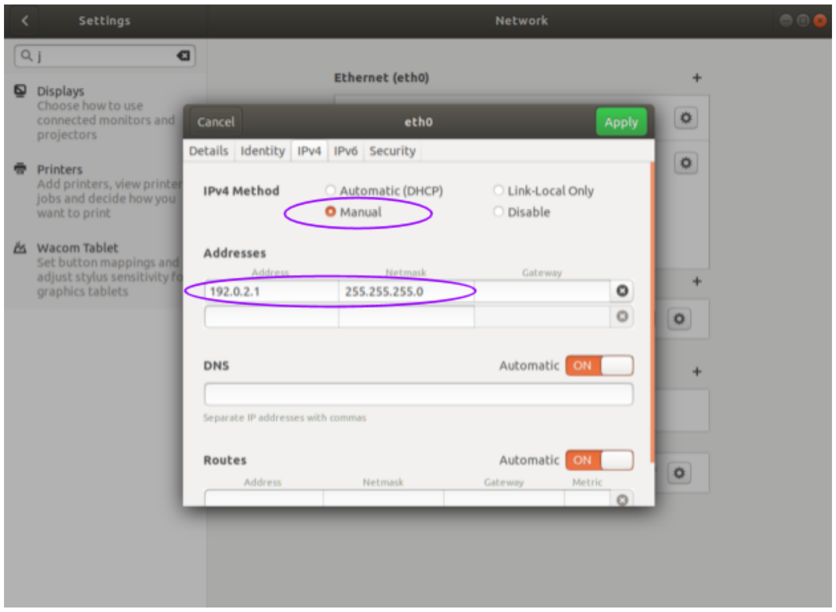
```
username@ubuntu:~$ nmcli con modify eth0 ipv4.method manual ipv4.addresses 192.0.2.1/24
```

Note: The `/24` is shorthand for Subnet Mask = `255.255.255.0`

Response

```
blank

Note: However you can verify the change has been made with the ``ip addr`` command.
      The ``inet`` line for the interface ``eth0`` will show the static IPv4 address. e.g. ``192.0.2.1``

   1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
          default qlen 1000
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
       inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever
   2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
          default qlen 1000
       link/ether 00:0c:29:2b:cc:48 brd ff:ff:ff:ff:ff:ff
       inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute eth0
          valid_lft forever preferred_lft forever
       inet6 fe80::be9f:d2a4:4451:3dfe/64 scope link noprefixroute
          valid_lft forever preferred_lft forever
   3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
          default qlen 1000
       link/ether 00:50:56:28:7a:8a brd ff:ff:ff:ff:ff:ff
       inet 172.16.79.232/24 brd 172.16.79.255 scope global wlan0
          valid_lft forever preferred_lft forever
       inet6 fe80::250:56ff:fe28:7a8a/64 scope link
          valid_lft forever preferred_lft forever
```

Via GUI The image below illustrates how to set the interface to `Manual (static)` mode using the graphical user interface.


Settings
Network
Displays
Choose how to use connected monitors and projectors
Printers
Add printers, view printer jobs and decide how you want to print
Wacom Tablet
Set button mappings and adjust stylus sensitivity for graphics tablets
Ethernet (eth0)
Cancel
eth0
Apply
Details
Identity
IPv4
IPv6
Security
Automatic (DHCP)
Link-Local Only
Disable
Addresses
Address
Netmask
Gateway
Separate IP addresses with commas
Metric

IPv4 Method: Manual

Address: 192.0.2.1 Netmask: 255.255.255.0

DNS Automatic ON

Routes Automatic ON

17.4.8 Finding a Sensor with mDNS Service Discovery

The sensor announces its presence on the network using Multicast Domain Name Service (mDNS) with a service type named `_roger._tcp`. You can use service discovery tools such as `avahi-browse` (Linux) to find all sensors connected to the network.

1. Find all sensors and their associated service text which includes the sensor IPv4 address using `avahi-browse` and the sensor service type `_roger._tcp`.

Command

```
avahi-browse -lrt [service type]
```

Example

```
username@ubuntu:~$ avahi-browse -lrt _roger._tcp
```

Response

```
+   eth0 IPv6 Ouster Sensor |sn|                    _roger._tcp          local
+   eth0 IPv4 Ouster Sensor |sn|                    _roger._tcp         local
=   eth0 IPv6 Ouster Sensor |sn|                    _roger._tcp          local
   hostname = [|os-sn|]
   address = [fe80::be0f:a7ff:fe00:1852]
   port = [7501]
   txt = ["fw=ousteros-image-prod-aries-v2.0.0-20200417193957" "sn=99201000067
8" "pn=840-102145-B"]
=   eth0 IPv4 Ouster Sensor |sn|                    _roger._tcp          local
   hostname = [|os-sn|]
```

(continues on next page)

(continued from previous page)

```
   address = []
   port = [7501]
   txt = ["fw=ousteros-image-prod-aries-v2.0.0-20200417193957" "sn= |sn|"
          "pn=840-102145-B"]
```

Note: In this example, your sensor IPv4 address is determined to be `169.254.0.123`.

18 Firmware Changelog

Version v2.4.0

Date 2022-08-31

Description

"Fixed"

- Bug Fix in High Input Voltage Alert behavior.
- Range aliasing improvement.
- Bug in keep-alive behavior for HTTP 1.1.
- **Critical bug fix:** Heavy vibration can trigger the sensor to reset without any alerts, causing an interruption to the data stream. The data stream will resume in approximately 60s. Note: Only affects FW version 2.3 and later.

"Changed"

- pixel_shift_by_row value updated.
- *Alerts and Errors* table has been updated to provide more information on each individual alert.

"Added"

- Add the following HTTP Commands to help configure and read the sensor config parameters (Refer to *Sensor Metadata* for more information): 1. GET /api/v1/sensor/config. 2. POST /api/v1/sensor/config.
- New optional argument for both TCP/HTTP commands "summary" for `get_alerts` command. **Note:** Valid uses of get_alerts:

  - get_alerts 1 (1 is the value of the cursor).
  - get_alerts summary.
  - get_alerts 2 summary (2 is the value of the cursor).

  Invalid uses of get_alerts:

  - get_alerts summary 2.

"Removed"

- `auto_start_flag` has been deprecated.
- `udp_ip` has been deprecated.
- `base_pn`, `base_sn` and `proto_rev` have been deprecated.

Version v2.3.1

Date 2022-06-03

Description

"Fixed"

- Bug Fix in RNG19_RFL8_SIG16_NIR16 mode.

"Added"

- Add support for new PN's.

Version v2.3.0

Date 2022-04-15

Description

"Added"

- Additional Channel Data Profiles (*Single Return Profile*, *Low Data Rate Profile*) were added as part of the Configurable Data Packet Format, refer to *Configurable Data Packet Format*.
- Additional options for config parameter udp_profile_lidar, refer to *Description- Configurable Parameters*.
- New TCP command get_telemetry, refer to *Sensor Status and Calibration*.

"Changed"

- Reflectivity and Signal value is now from 1 to 255. 0 correspond to an invalid reflectivity and Signal value (similar as Range).

"Fixed"

- Fixed startup packet drop behavior (5b alert).

Version v2.2.1

Date 2022-02-17

Description

"Fixed"

- Fixed point cloud detection on a subset of sensors

Version v2.2.0

Date 2021-12-18

Description

"Added"

- Add support for Configurable Data Packet Format
- Add support for Dual Returns mode

"Changed"

- Update Sensor Web UI
- Update TCP command get_lidar_data_format
- Update TCP command get_sensor_info

Version v2.1.3

Date 2021-10-22

Description

"Added"

- Added PN support

Version v2.1.2

Date 2021-07-16

Description

"Added"

- Add support for minor hardware revisions

Version v2.1.1

Date 2021-06-21

Description

"Added"

- Add support for Calibrated Reflectivity
- Add Config UI to sensor web page (Beta)
- Add signal multiplier modes to increase signal strength in the enabled azimuth window for gen2 sensors only
- Add alerts for motor speed
- Add alerts for unexpected sensor state transition
- Improve OS2 cold start to -20°C
- Improve OS2 signal strength by 16%

"Removed"

- Delete TCP command set_data_dst_ip
- Delete TCP command get_data_dst_ip
- Delete TCP command set_udp_port_lidarAdded PN support for new turntable boards
- Delete TCP command set_udp_port_imu

- Delete TCP command get_lidar_mode
- Delete TCP command set_lidar_mode
- Delete TCP command get_config_file_path
- Delete TCP command set_auto_start_flag
- Delete TCP command get_auto_start_flag
- Delete TCP command get_watchdog_status

"Changed"

- Change the Reflectivity values in the packets from 16-bit to 8-bit

"Fixed"

- Fixed phase locked motor control to handle out-of-bounds motor velocity
- Slow time sync on initial boot with PTP
- Fixed azimuth_window parameter logic behavior

19 Appendix

19.1 Features / Releases Support Table

Table 19.1: Features / Releases Support Table

Features	Supported FW Versions	Supported HW Revisions
Signal multiplier	2.1.0 and higher	Rev C (PN: 840-102XXX-C) and higher
Azimuth window masking	2.1.0 and higher	Rev C (PN: 840-102XXX-C) and higher
Calibrated reflectivity	2.1.0 and higher	OS0 & OS1 Rev C (PN: 840-102XXXC) and higher
Dual Returns	2.2.0 and higher	Rev 06 (PN:840-102xxx-06) and higher
Channel Data Profiles (Single Return, Low Data Rate) & *Sensor Telemetry*	2.3.0 and higher	All Hardware revisions

Note: *Channel Data Profiles* (Single Return, Low Data Rate configurations along with Dual returns) were added as part of the Configurable Data Packet Format in FW 2.3. Please note in order to enable dual returns the user needs to have both a **Rev 06** sensor or later and FW version 2.2 or later.

Note: Internal Temperature using *Sensor Telemetry* can be measured only on **Rev 06** and later sensors.

19.2 Lidar Packet Format Update

Starting in firmware v2.0.0, all sensors with the same number of channels have the same data structure and same maximum data rate. Prior to v2.0.0, all sensors, regardless of their number of channels, had the same data rate.

If you have either a Gen 1 OS1-16 or Gen 1 OS1-32, upon upgrading to firmware v2.0.0, you will see a drop in data rate. Please refer to the diagram below for a visualization of lidar packet structure.

Prior to v2.0.0, all sensors, regardless of number of channels, had a fixed number of data blocks in their lidar packets. In v2.0.0, the number of data blocks in a sensor's Measurement Block is equal to the number of channels it has. Customers with Gen 1 OS1-16 or Gen 1 OS1-32 will see a 75% and 50% respective drop in data rate due to unused data blocks being omitted from the sensor output.

These customers will also see a change in the output of the TCP command `get_beam_intrinsics`. Previously, the `beam_azimuth_angles` and `beam_altitude_angles` output array was padded with zeros so that they were always of length 64, regardless of the number of channels in that sensor. Now, the padded zeros are dropped so that the lengths of both arrays are equal to number of channel in the sensor e.g. all 32-channel sensors will have `beam_azimuth_angles` and `beam_altitude_angles` output arrays of length 32 on v2.0.0 and beyond.

The TCP command `get_lidar_data_format` can also be useful in interpreting the lidar data format structure:

- `columns_per_frame`: Number of data columns per frame. This can be 512, 1024, or 2048, depending upon the set lidar mode.
- `columns_per_packet`: Number of Measurement Blocks contained in a single lidar packet. In v2.0.0 and earlier, this is 16.
- `pixel_shift_by_row`: Offset in terms of pixel count. Can be used to destagger image. Varies by lidar mode. Length of this array is equal to the number of channels of the sensor.
- `pixels_per_column`: Number of channels of the sensor.
- `column_window`: Indices of active columns of data in the sensor. These bounds will change when a custom azimuth window is used.

Note: Please refer to *Packet Size Calculation (Configurable)* section to compare max data rates and the *Packet Size Calculation (LEGACY)* table to compare lidar packet sizes of all sensors on firmware v2.0.0.

19.3 Lidar format update appearing in v2.2.0

A new data packet format has been implemented in addition to the LEGACY one (still available). Please refer to *Configurable Data Packet Format* section for more details.

19.4 PTP Profiles Guide

This guide provides instructions on setting the Precision Time Protocol (PTP) profile of the Ouster sensor. The profile of the Ouster sensor and your master clock must match for time synchronization to be possible.

19.4.1 PTP Profiles

There are several PTP profiles that are commonly used. The supported profiles on the Ouster sensor are listed below:

- `default` - The IEEE 1588 Default PTP profile addresses many common applications. Most PTP capable devices support the Default profile.
- `gptp` - Generalized PTP (gPTP) is the common name for the IEEE standard 802.1AS-2011 which improves the interoperability of PTP by simplifying the supported options. The gPTP profile is useful when using the Ouster sensor with gPTP compatible hardware such as an Audio Visual Bridge (AVB), e.g. the MOTU AVB.
- `automotive-slave` - The Automotive Slave PTP profile is commonly used in automotive applications. The primary differences from other profiles are that the Best Master Clock Algorithm (BMCA) is disabled, the slave device inhibits announce messages, and the time convergence controller is approximately 8 times faster than the Default profile.

19.4.2 PTP HTTP API

The PTP profile of the sensor is changed using an HTTP PUT request. This can be done using several different tools such as HTTPie, curl, Advanced REST Client, etc. The full API is available in *GET /api/v1/time/ptp/profile*.

- The request URL is: http://<sensor_hostname>/api/v1/time/ptp/profile/
- Valid values are (“”, are included):
  - `“default”`
  - `“gptp”`
  - `“automotive-slave”`

Note: Changing the PTP profile does not require reinitialization or writing the configuration text file to be persistent. It is persistent as soon a valid PUT request is executed and a valid response is received.

19.4.3 Enabling the PTP profiles

Below are some examples using popular command-line tools.

19.4.4 Example using cURL

In this example we are setting the PTP profile of the Ouster sensor to `"gptp"` using the cURL command line tool.

- Command

```
curl -X PUT -H "Content-Type: application/json" -d '"gptp"' http://<sensor_hostname>/api/v1/time/ptp
/profile/
```

- Response

```
"gptp"%
```

19.4.5 Example using HTTPie

In this example we are setting the PTP profile of the Ouster sensor to `"default"` using the HTTPie command line tool.

- Command

```
http PUT http://<sensor_hostname>/api/v1/time/ptp/profile <<< '"default"'
```

- Response

```
"default"%
```

19.4.6 Sync Verification

Please see the *Sensor PTP Sync Verification* section for details on how to verify the sensor is synchronized.

19.5 PTP Quickstart Guide

There are many configurations for a PTP network, this quick start guide aims to cover the basics by using Ubuntu 18.04 as an example. It provides configuration settings for a commercial PTP grandmaster clock and also provides directions on setting up a Linux computer (Ubuntu 18.04) to function as a PTP grandmaster.

The linuxptp project provides a suite of PTP tools that can be used to serve as a PTP master clock for a local network of sensors.

19.5.1 Assumptions

- Command line Linux knowledge (e.g., package management, command line familiarity, etc.).
- Ethernet interfaces that support hardware timestamping.
- Ubuntu 18.04 is assumed for this tutorial, but any modern distribution should suffice.
- Knowledge of systemd service configuration and management.
- Familiarity with Linux permissions.

19.5.2 Physical Network Setup

Ensure the Ouster sensor is connected to the PTP master clock with at most one network switch. Ideally the sensor should be connected directly to the PTP grandmaster. Alternatively, a simple layer-2 gigabit Ethernet switch will suffice. Multiple switches are not recommended and will add unnecessary jitter.

19.5.3 Third Party Grandmaster Clock

A dedicated grandmaster clock should be used for the highest absolute accuracy often with a GPS receiver.

It must be configured with the following parameters which match the *linuxptp* client defaults:

- Transport: `UDP IPv4`
- Delay Mechanism: `E2E`
- Sync Mode: `Two-Step`
- Announce Interval: `1` - sent every 2 seconds
- Sync Interval: `0` - sent every 1 second
- Delay Request Interval: `0` - sent every 1 second

For more settings, review the `port_data_set` field returned from the sensor's `HTTP /time/ptp` interface.

19.5.4 Linux PTP Grandmaster Clock

An alternative to an external grandmaster PTP clock is to run a local Linux PTP master clock if accuracy allows. This is often implemented on a vehicle computer that interfaces directly with the lidar sensors.

This section outlines how to configure a master clock.

19.5.5 Example Network Setup

This section assumes the following network setup as it has elements of a local master clock and the option for an upstream PTP time source.

```
+-------------------------------------+
| Ubuntu 18.04 System                 |
| * 2x Intel i210 Ethernet Interfaces |
| * Linux PTP service                 |
|                                     |
|      eno1                 eno2      |
+-------+---------------------+-------+
        |                     |
+-------+-------+    +--------+------+
| Trimble GM100 |    |              + +
|  GPS -> PTP   |    |  Ouster OS1   | |
|  grandmaster  |    |               | |
|  (optional)   |    |               | |
+---------------+    +---------------- |
                      +--------------- +
```

The focus is on configuring the Linux PTP service to serve a common clock to all the downstream Ouster OS1 sensors using the Linux system time from the Ubuntu host machine.

Optionally, a grandmaster clock can be added to discipline the system time of the Linux host.

19.5.6 Installing Necessary Packages

Several packages are needed for PTP functionality and verification:

- `linuxptp` - Linux PTP package with the following components:
  - `ptp4l` daemon to manage hardware and participate as a PTP node
  - `phc2sys` to synchronize the Ethernet controller's hardware clock to the Linux system clock or shared memory region
  - `pmc` to query the PTP nodes on the network.
- `chrony` - A NTP and PTP time synchronization daemon. It can be configured to listen to both NTP time sources via the Internet and a PTP master clock such as one provided by a GPS with PTP support. This will validate the time configuration makes sense given multiple time sources.
- `ethtool` - A tool to query the hardware and driver capabilities of a given Ethernet interface.

```
$ sudo apt update
...
```

(continues on next page)

(continued from previous page)

```
Reading package lists... Done
Building dependency tree
Reading state information... Done

$ sudo apt install linuxptp chrony ethtool
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  chrony ethtool linuxptp
0 upgraded, 3 newly installed, 0 to remove and 29 not upgraded.
Need to get 430 kB of archives.
After this operation, 1,319 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 ethtool amd64 1:4.15-0ubuntu1 [114 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 linuxptp amd64 1.8-1 [112 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 chrony amd64 3.2-4ubuntu4.2 [203 kB]
Fetched 430 kB in 1s (495 kB/s)
Selecting previously unselected package ethtool.
(Reading database ... 117835 files and directories currently installed.)
Preparing to unpack .../ethtool_1%3a4.15-0ubuntu1_amd64.deb ...
Unpacking ethtool (1:4.15-0ubuntu1) ...
Selecting previously unselected package linuxptp.
Preparing to unpack .../linuxptp_1.8-1_amd64.deb ...
Unpacking linuxptp (1.8-1) ...
Selecting previously unselected package chrony.
Preparing to unpack .../chrony_3.2-4ubuntu4.2_amd64.deb ...
Unpacking chrony (3.2-4ubuntu4.2) ...
Setting up linuxptp (1.8-1) ...
Processing triggers for ureadahead (0.100.0-20) ...
ureadahead will be reprofiled on next reboot
Setting up chrony (3.2-4ubuntu4.2) ...
Processing triggers for systemd (237-3ubuntu10.13) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up ethtool (1:4.15-0ubuntu1) ...
```

19.5.7 Ethernet Hardware Timestamp Verification

Identify the Ethernet interface to be used on the client (Linux) machine, e.g., eno1. Run the ethtool utility and query this network interface for supported capabilities.

Output of `ethtool -T` for a functioning Intel i210 Ethernet interface:

```
$ sudo ethtool -T eno1
Time stamping parameters for eno1:
Capabilities:
        hardware-transmit     (SOF_TIMESTAMPING_TX_HARDWARE)
        software-transmit     (SOF_TIMESTAMPING_TX_SOFTWARE)
        hardware-receive      (SOF_TIMESTAMPING_RX_HARDWARE)
        software-receive      (SOF_TIMESTAMPING_RX_SOFTWARE)
        software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
        hardware-raw-clock    (SOF_TIMESTAMPING_RAW_HARDWARE)
PTP Hardware Clock: 0
```

(continues on next page)

(continued from previous page)

```
Hardware Transmit Timestamp Modes:
        off                   (HWTSTAMP_TX_OFF)
        on                    (HWTSTAMP_TX_ON)
Hardware Receive Filter Modes:
        none                  (HWTSTAMP_FILTER_NONE)
        all                   (HWTSTAMP_FILTER_ALL)
```

19.5.8 Configurations

Configuring ``ptp4l`` for Multiple Ports

On a Linux system with multiple Ethernet ports (i.e. Intel i210) `ptp4l` needs to be configured to support all of them.

```
boundary_clock_jbod 1
[eno1]
[eno2]
```

Note: Add the above required modification at the end of the existing file. Deleting or editing the default settings section of the ptp4l.conf file will result in an error.

The default systemd service file for Ubuntu 18.04 attempts to use the eth0 address on the command line. Override systemd service file so that the configuration file is used instead of hard coded in the service file.

Create a systemd drop-in directory to override the system service file:

```
$ sudo mkdir -p /etc/systemd/system/ptp4l.service.d
```

Create a file at `/etc/systemd/system/ptp4l.service.d/override.conf` with the following contents:

```
[Service]
ExecStart=
ExecStart=/usr/sbin/ptp4l -f /etc/linuxptp/ptp4l.conf
```

Restart the `ptp4l` service so the change takes effect:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart ptp4l
$ sudo systemctl status ptp4l
* ptp4l.service - Precision Time Protocol (PTP) service
   Loaded: loaded (/lib/systemd/system/ptp4l.service; enabled; vendor preset: enabled)
  Drop-In: /etc/systemd/system/ptp4l.service.d
           └─override.conf
   Active: active (running) since Wed 2019-03-13 14:38:57 PDT; 3s ago
     Docs: man:ptp4l
 Main PID: 25783 (ptp4l)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/ptp4l.service
```

(continues on next page)

(continued from previous page)

```
         └─25783 /usr/sbin/ptp4l -f /etc/linuxptp/ptp4l.conf

Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.756] port 1: INITIALIZING to LISTENING on INITIALIZE
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.756] driver changed our HWTSTAMP options
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.756] tx_type   1 not 1
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.756] rx_filter 1 not 12
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.756] port 2: INITIALIZING to LISTENING on INITIALIZE
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.757] port 0: INITIALIZING to LISTENING on INITIALIZE
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.757] port 1: link up
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.757] port 2: link down
Mar 13 14:38:57 leadlizard ptp4l[25783]: [590188.757] port 2: LISTENING to FAULTY on
FAULT_DETECTED (FT_UNSPECIFIED)
Mar 13 14:38:58 leadlizard ptp4l[25783]: [590189.360] port 1: new foreign master 001747.fffe.700038-1
```

The above `systemctl status ptp4l` console output shows systemd correctly reading the override file created earlier before starting several seconds after the restart command.

The log output shows that a grandmaster clock has been discovered on port 1 (`eno1`) and port 2 (`eno2`) is currently disconnected and in the faulty state as expected. In the test network a Trimble Thunderbolt PTP GM100 Grandmaster Clock is attached on `eno1`.

Logs can be monitored (i.e. followed) like so:

```
$ journalctl -f -u ptp4l
-- Logs begin at Fri 2018-11-30 06:40:50 PST. --
Mar 13 14:51:37 leadlizard ptp4l[25783]: [590948.224] master offset   -17 s2 freq  -25963 path delay 14183
Mar 13 14:51:38 leadlizard ptp4l[25783]: [590949.224] master offset   -13 s2 freq  -25964 path delay 14183
Mar 13 14:51:39 leadlizard ptp4l[25783]: [590950.225] master offset    35 s2 freq  -25920 path delay 14192
Mar 13 14:51:40 leadlizard ptp4l[25783]: [590951.225] master offset   -59 s2 freq  -26003 path delay 14201
Mar 13 14:51:41 leadlizard ptp4l[25783]: [590952.225] master offset   -24 s2 freq  -25986 path delay 14201
Mar 13 14:51:42 leadlizard ptp4l[25783]: [590953.225] master offset   -39 s2 freq  -26008 path delay 14201
Mar 13 14:51:43 leadlizard ptp4l[25783]: [590954.225] master offset    53 s2 freq  -25928 path delay 14201
Mar 13 14:51:44 leadlizard ptp4l[25783]: [590955.226] master offset   -85 s2 freq  -26050 path delay 14207
Mar 13 14:51:45 leadlizard ptp4l[25783]: [590956.226] master offset   127 s2 freq  -25863 path delay 14207
Mar 13 14:51:46 leadlizard ptp4l[25783]: [590957.226] master offset     9 s2 freq  -25943 path delay 14208
Mar 13 14:51:47 leadlizard ptp4l[25783]: [590958.226] master offset   -23 s2 freq  -25973 path delay 14208
Mar 13 14:51:48 leadlizard ptp4l[25783]: [590959.226] master offset   -61 s2 freq  -26018 path delay 14190
Mar 13 14:51:49 leadlizard ptp4l[25783]: [590960.226] master offset    69 s2 freq  -25906 path delay 14190
Mar 13 14:51:50 leadlizard ptp4l[25783]: [590961.226] master offset   -73 s2 freq  -26027 path delay 14202
Mar 13 14:51:51 leadlizard ptp4l[25783]: [590962.226] master offset    19 s2 freq  -25957 path delay 14202
Mar 13 14:51:52 leadlizard ptp4l[25783]: [590963.226] master offset   147 s2 freq  -25823 path delay 14202
...
```

Configuring `ptp4l` as a Local Master Clock

The IEEE-1588 Best Master Clock Algorithm (*BMCA*) will select a grandmaster clock based on a number of masters. In most networks there should be only a single master. In the example network the Ubuntu machine will be configured with a non-default *clockClass* so its operation qualifies it to win the BMCA.

Replace the default value with a lower clock class (higher priority) and restart linuxptp. Edit `/etc/linuxptp/ptp4l.conf` and comment out the default `clockClass` value and insert a line setting it 128.

```
#clockClass     248
clockClass      128
```

Restart ptp4l so the configuration change takes effect.

```
$ sudo systemctl restart ptp4l
```

This will configure `ptp4l` to advertise a master clock on eno2 as a clock that will win the BMCA for an Ouster OS1 sensor.

However, the `ptp4l` service is only advertising the Ethernet controller's PTP hardware clock, not the Linux system time as is often expected.

Configuring ``phc2sys`` to Synchronize the System Time to the PTP Clock

To synchronize the Linux system time to the PTP hardware clock the `phc2sys` utility needs to be run. The following configuration will tell `phc2sys` to take the Linux `CLOCK_REALTIME` and write that time to the PTP hardware clock in the Ethernet controller for `eno2`. These interfaces are then connected to PTP slaves such as Ouster OS1 sensors.

Create a systemd drop-in directory to override the system service file:

```
$ sudo mkdir -p /etc/systemd/system/phc2sys.service.d
```

Create a file at `/etc/systemd/system/phc2sys.service.d/override.conf` with the following contents:

```
[Service]
ExecStart=
ExecStart=/usr/sbin/phc2sys -w -s CLOCK_REALTIME -c eno2
```

Note: If multiple interfaces need to be synchronized from `CLOCK_REALTIME` then multiple instances of the `phc2sys` service need to be run as it only accepts a single slave (i.e. `-c`) argument.

Restart the `phc2sys` service so the change takes effect:

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart phc2sys
$ sudo systemctl status phc2sys
```

Configuring Chrony to Set System Clock Using PTP

An upstream PTP grandmaster clock (e.g., a GPS disciplined PTP clock) can be used to set the system time if precise absolute time is needed for sensor data.

Chrony is a Linux time service that can read from NTP and PTP and set the Linux system time using the most accurate source available. With a properly functioning PTP grandmaster the PTP time source will be selected and the error from the public time servers can be reviewed.

The following phc2shm service will synchronize the time from `eno1` (where the external grandmaster is attached) to the system clock.

Create a file named `/etc/systemd/system/phc2shm.service` with the following contents:

```
# /etc/systemd/system/phc2shm.service
[Unit]
Description=Synchronize PTP hardware clock (PHC) to NTP SHM
Documentation=man:phc2sys
After=ntpdate.service
Requires=ptp4l.service
After=ptp4l.service

[Service]
Type=simple
ExecStart=/usr/sbin/phc2sys -s eno1 -E ntpshm -w

[Install]
WantedBy=multi-user.target
```

Then start the newly created service and check that it started.

```
$ sudo systemctl start phc2shm
$ sudo systemctl status phc2shm
```

Add the PTP time source to the chrony configuration which will read the shared memory region managed by the `phc2shm` service created above.

Append the following to the `/etc/chrony/chrony.conf` file:

```
refclock SHM 0 poll 1 refid ptp
```

Restart chrony so the updated configuration file takes effect:

```
$ sudo systemctl restart chrony
```

After waiting a minute for the clock to synchronize, review the chrony client timing accuracy:

```
$ chronyc tracking
Reference ID    : 70747000 (ptp)
Stratum         : 1
Ref time (UTC)  : Thu Mar 14 02:22:58 2019
System time     : 0.000000298 seconds slow of NTP time
Last offset     : -0.000000579 seconds
RMS offset      : 0.001319735 seconds
Frequency       : 0.502 ppm slow
Residual freq   : -0.028 ppm
Skew            : 0.577 ppm
Root delay      : 0.000000001 seconds
Root dispersion : 0.000003448 seconds
Update interval : 2.0 seconds
Leap status     : Normal

$ chronyc sources -v
210 Number of sources = 9

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
```

(continues on next page)

(continued from previous page)

```
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#* ptp                           0   1   377     1     +27ns[  +34ns] +/-  932ns
^- chilipepper.canonical.com     2   6   377    61   -482us[ -482us] +/-   99ms
^- pugot.canonical.com           2   6   377    62   -498us[ -498us] +/-  112ms
^- golem.canonical.com           2   6   337    59   -467us[ -468us] +/-   95ms
^- alphyn.canonical.com          2   6   377    58   +957us[ +957us] +/-   95ms
^- legacy13.chi1.ntfo.org        3   6   377    62    -10ms[  -10ms] +/-  178ms
^- tesla.selinc.com              2   6   377   128   +429us[ +514us] +/-   42ms
^- io.crash-override.org         2   6   377    59   +441us[ +441us] +/-   58ms
^- hadb2.smatwebdesign.com       3   6   377    58  +1364us[+1364us] +/-   99ms
```

Note that the `Reference ID` matches the `ptp` reference ID from the chrony.conf file and that the sources output shows the `ptp` reference ID as selected (signified by the `*` state in the second column). Additionally, the NTP time sources show a small relative error to the high accuracy PTP time source.

In this case the PTP grandmaster is properly functioning.

If this error is large, chrony will select the NTP time sources and mark the PTP time source as invalid. This typically signifies that something is mis-configured with the PTP grandmaster upstream of this device or the linuxptp configuration.

19.5.9 Verifying Operation

If the PTP grandmaster was just set up and configured, it's recommended to power cycle the sensor. The sensor will then jump to the correct time instead of slowly easing in the time adjustment which will take time if the grandmaster initially set the sensor to the wrong time.

19.6 Sensor PTP Sync Verification

The sensor can be queried for the state of its local PTP service through the *GET /api/v1/time/ptp* request.

JSON response fields to check:

- `parent_data_set.grandmaster_identity` should list the identity of the local grandmaster
- `port_data_set.port_state` should be `SLAVE`
- `time_status_np.gm_present` should be `true`
- `time_status_np.master_offset` which is given in nanoseconds, should be less than `250000`. This equates to 250 microseconds.

PTP Example JSON Response

```
{
  "profile": "default",
  "parent_data_set":
  {
    "grandmaster_identity": "001747.fffe.700038",
    "parent_port_identity": "ac1f6b.fffe.1db84e-2",
    "parent_stats": 0,
    "gm_clock_class": 6,
    "observed_parent_clock_phase_change_rate": 2147483647,
    "gm_clock_accuracy": 33,
    "gm_offset_scaled_log_variance": 65535,
    "grandmaster_priority1": 128,
    "grandmaster_priority2": 128,
    "observed_parent_offset_scaled_log_variance": 65535
  },
  "current_data_set":
  {
    "steps_removed": 1,
    "offset_from_master": 61355,
    "mean_path_delay": 117977.0
  },
  "port_data_set":
  {
    "port_state": "SLAVE",
    "peer_mean_path_delay": 0,
    "log_min_delay_req_interval": 0,
    "port_identity": "bc0fa7.fffe.c48254-1",
    "log_sync_interval": 0,
    "log_announce_interval": 1,
    "delay_mechanism": 1,
    "log_min_pdelay_req_interval": 0,
    "announce_receipt_timeout": 3,
    "version_number": 2
  },
  "time_status_np":
  {
    "gm_time_base_indicator": 0,
    "gm_identity": "001747.fffe.700038",
    "cumulative_scaled_rate_offset": 0,
    "scaled_last_gm_phase_change": 0,
    "ingress_time": 0,
    "master_offset": 61355,
    "last_gm_phase_change": "0x0000'0000000000000000.0000",
    "gm_present": true
  },
  "time_properties_data_set":
  {
    "frequency_traceable": 0,
    "leap61": 0,
    "time_traceable": 0,
    "current_utc_offset": 37,
    "leap59": 0,
    "current_utc_offset_valid": 0,
```

(continues on next page)

(continued from previous page)

```
    "time_source": 160,
    "ptp_timescale": 1
  }
}
```

19.6.1 LinuxPTP PMC Tool

The sensor will respond to PTP management messages. The linuxptp `pmc` (see `man pmc`) utility can be used to query all PTP devices on the local network.

On the Linux host for the `pmc` utility to communicate with then run the following command:

```
$ sudo pmc 'get PARENT_DATA_SET' 'get CURRENT_DATA_SET' 'get PORT_DATA_SET' 'get TIME_STATUS_NP' -i eno2
sending: GET PARENT_DATA_SET
sending: GET CURRENT_DATA_SET
sending: GET PORT_DATA_SET
sending: GET TIME_STATUS_NP
        bc0fa7.fffe.c48254-1 seq 0 RESPONSE MANAGEMENT PARENT_DATA_SET
                parentPortIdentity                    ac1f6b.fffe.1db84e-2
                parentStats                           0
                observedParentOffsetScaledLogVariance 0xffff
                observedParentClockPhaseChangeRate    0x7fffffff
                grandmasterPriority1                  128
                gm.ClockClass                         6
                gm.ClockAccuracy                      0x21
                gm.OffsetScaledLogVariance            0x4e5d
                grandmasterPriority2                  128
                grandmasterIdentity                   001747.fffe.700038
        bc0fa7.fffe.c48254-1 seq 1 RESPONSE MANAGEMENT CURRENT_DATA_SET
                stepsRemoved     2
                offsetFromMaster 61355.0
                meanPathDelay    117977.0
        bc0fa7.fffe.c48254-1 seq 2 RESPONSE MANAGEMENT PORT_DATA_SET
                portIdentity            bc0fa7.fffe.c48254-1
                portState               SLAVE
                logMinDelayReqInterval  0
                peerMeanPathDelay       0
                logAnnounceInterval     1
                announceReceiptTimeout  3
                logSyncInterval         0
                delayMechanism          1
                logMinPdelayReqInterval 0
                versionNumber           2
        bc0fa7.fffe.c48254-1 seq 3 RESPONSE MANAGEMENT TIME_STATUS_NP
                master_offset              61355
                ingress_time               0
                cumulativeScaledRateOffset +0.000000000
                scaledLastGmPhaseChange    0
                gmTimeBaseIndicator        0
                lastGmPhaseChange          0x0000'0000000000000000.0000
                gmPresent                  true
                gmIdentity                 001747.fffe.700038
```

19.6.2 Tested Grandmaster Clocks

- **Trimble Thunderbolt PTP GM100 Grandmaster Clock**
    - Firmware version: `20161111-0.1.4.0, November 11 2016 15:58:25`
    - PTP configuration:

```
 > get ptp eth0
             Enabled : Yes
            Clock ID : 001747.fffe.700038-1
             Profile : 1588
       Domain number : 0
  Transport protocol : IPV4
             IP Mode : Multicast
     Delay Mechanism : E2E
           Sync Mode : Two-Step
         Clock Class : 6
          Priority 1 : 128
          Priority 2 : 128
       Multicast TTL : 0
       Sync interval : 0
    Del Req interval : 0
       Ann. interval : 1
Ann. receipt timeout : 3
```

- **Ubuntu 18.04 + Linux PTP as a master clock**
    - Intel i210 Ethernet interface
    - PCI hardware identifiers: `8086:1533 (rev 03)`
- Ubuntu 18.04 kernel package: `linux-image-4.18.0-16-generic`
- Ubuntu 18.04 linuxptp package: `linuxptp-1.8-1`

19.7 Analyzing Linux Networking Issues

Note: Users are recommended to follow this section only in the case of intermittent packet drops or packet reordering. Please make sure to double check `udp_dest` settings at the beginning of this section, as the information provided is not useful if users are getting zero data.

In case the users are getting zero data and are unable to resolve the issue please contact our Field Application Team.

This section captures tools and procedures to troubleshoot networking issues for a system consisting of a PC/Workstation L2 Switch and one or more Ouster Sensors. Though examples use the Linux Operating System as a model, the material is equally relevant to debugging issues in the Windows environment. Where possible Windows command-line and UI analogs will be discussed in passing.

Debugging the Workstation Data Path

The workstation maintains a set of statistics associated with each layer in the network stack that can be used to diagnose packet loss. The correct way to approach a network stack problem is to start with the lowest layer in the stack first, examine the statistics for errors, and work your way up to the highest layer. The reason that we start with the lowest layer is that issues in the lowest layer can cause issues in other parts of the data-path.

19.7.1 Link Layer Statistics and Configuration

ethtool

In Linux, ethtool is used to query the NIC for statistics as well as view and change the NIC configuration. Linux also offers more generic mechanisms to do this by writing/reading keys in the kernel file-system. Ethtool is often the tool that is widely use to debug system, and is generally the most complete system for configuration and debug. Ethtool is a double edged-sword, because ethtool is vendor-centric the output of its commands and range of configuration options will be slightly different depending on which NIC is used.

Line Interface Statistics

The most useful starting point when debugging the link-layer is to examine the line-interface statics, these are queried with ethtool -S <ethX> where ethX is the identifier of the NIC as listed by ifconfig, if the device has multiple NICs and you are uncertain which NIC is receiving the traffic, run some traffic and monitor the stats reported by ifconfig.

Note: The output of ethtool -S <ethX> is 100% NIC vendor specific and will be quite different depending on NIC vendor used in your system.

Example: Output of ethtool -S:

```
NIC statistics:
    rx_packets: 0
    tx_packets: 0
    rx_bytes: 0
```

(continues on next page)

(continued from previous page)

```
    tx_bytes: 0
    rx_broadcast: 0
    tx_broadcast: 0
    rx_multicast: 0
    tx_multicast: 0
    rx_errors: 0
    tx_errors: 0
    tx_dropped: 0
    multicast: 0
    collisions: 0
    rx_length_errors: 0
    rx_over_errors: 0
    rx_crc_errors: 0
    rx_frame_errors: 0
    rx_no_buffer_count: 0
    rx_missed_errors: 0
    tx_aborted_errors: 0
    tx_carrier_errors: 0
    tx_fifo_errors: 0
    tx_heartbeat_errors: 0
    tx_window_errors: 0
    tx_abort_late_coll: 0
    tx_deferred_ok: 0
    tx_single_coll_ok: 0
    tx_multi_coll_ok: 0
    tx_timeout_count: 52
    tx_restart_queue: 0
    rx_long_length_errors: 0
    rx_short_length_errors: 0
    rx_align_errors: 0
    tx_tcp_seg_good: 0
    tx_tcp_seg_failed: 0
    rx_flow_control_xon: 0
    rx_flow_control_xoff: 0
    tx_flow_control_xon: 0
    tx_flow_control_xoff: 0
    rx_csum_offload_good: 0
    rx_csum_offload_errors: 0
    rx_header_split: 0
    alloc_rx_buff_failed: 0
    tx_smbus: 0
    rx_smbus: 0
    dropped_smbus: 0
    rx_dma_failed: 0
    tx_dma_failed: 0
    rx_hwtstamp_cleared: 0
    uncorr_ecc_errors: 0
    corr_ecc_errors: 0
    tx_hwtstamp_timeouts: 0
    tx_hwtstamp_skipped: 0
```

MAC Errors

Users are mainly interested in the path where the sensor is transmitting to the workstation, focusing on the "rx" (receive) statistics. Generally, anything that is labeled as rx.*error on this NIC constitutes a stats that might be helpful in diagnosing the problem.

Based on the NIC, these "error" statistics are primarily associated with problems identified by the MAC. Such problems are generally indicative of an L1 problem (though they could also indicate a problem with the link-partner's MAC), such as a loose connector, faulty transceiver, or an out-of-spec cable.

Internal System Errors

User might come across stats like rx_dma_failed and rx_no_buffer_count that do not have an "error" postfix but constitute very real errors. These are indicative of failures in the hand-off between the NIC driver.

Solving MAC Errors

If users encounter MAC errors this most likely points to a cabling issue, so the first step would be to replace the cable. If the errors persist, the next step would be to try to test against a different node. One can use the "iPerf" or "iPerf3" utility (discussed below) to validate that the workstation against another workstation computer. A final step would be to swap out the sensor.

Solving Internal System Errors

These errors are often the most difficult to understand. It can be quite surprising that the MAC is receiving everything and traffic is still being dropped. The root cause is generally that the processor cannot handle the peak rate. Though the average load may be only a few hundred megabits, the real situation is that all traffic received by the NIC arrives at line rate - for a 10G NIC this means that many frames may be received back-to-back at the line rate of the NIC.

Just how many frames arrive depends on the behavior of the sensors. Ouster sensor attempts to transmit the entire LIDAR frame all at once. Assuming a 40K (on the wire) LiDAR frame and 10 sensors, the worst case load will be 40K x 10 = 400K at 10G (since the peak transmit rate of each sensor is 1G x 10 = 10G.) 400K is a lot of 10G data to process all at once, and without hardware buffering things will certainly fail.

The NIC maintains a hardware ring-buffer or on advanced hardware, potentially multiple ring-buffers. The entries in the ring-buffer are pointers into kernel packet-buffer structures. This mechanism enables the NIC to efficiently deliver packets to the kernel at line rate. For our specific use-case the default size of this ring-buffer may be too small.

To update this value user can use ethtool:

- ethtool -g <ethX> will display the current setting and device limits
- ethtool -G <ethX> rx <value> is used to update the setting

Example: Using a laptop/sytem, ring-buffer has enough buffer for 256 entries by default:

```
ethtool -g enp0s31f6
Ring parameters for enp0s31f6:
Pre-set maximums:
RX: 4096
```

(continues on next page)

(continued from previous page)

```
RX Mini: 0
RX Jumbo: 0
TX: 4096
Current hardware settings:
RX: 256
RX Mini: 0
RX Jumbo: 0
TX: 256
```

To find out how much buffer is sufficient we can apply the burst-tolerance equation:

```
fill_rate = NIC_line_speed - max_measured_throughput
fill_time = rx_buffer_size * 1518 * 8 / fill_rate
MBS = fill_time * NIC_line_speed
```

Note: It is not always easy to obtain max_measured_throughput, and in a busy workstation it can be subject to variable delay.

As a rule-of-thumb we need to at least accommodate one max-burst (one LiDAR packet) from the sensor. Assuming a 40KB LiDAR packet that's 40KB/1518=27 frames. So 256 should be more than adequate.

However, even with the default buffer of 256, user can observe packet loss due to DMA errors. This is because the work-station is not a real-time system and the delay can be quite variable. Linux uses a technique called interrupt coalescence that determines how often it will service the driver, when it gets very busy.

Interrupt coalescence is controlled by the kernel filesystem key:

`/proc/sys/net/core/netdev_budget_usecs and by default it's 8000us!`

On a 10G interface like Bane that's .008 * 10G / (1518 * 8) = 6588 frames

If the problem is not resolved by increasing the buffer size, it's possible to reduce netdev_budget_usecs in order to favor moving data over other activities that the system could be doing. It's also possible to increase the maximum number of frames the OS is willing to process when the line interface does get serviced which is controlled by:

`/proc/sys/net/core/netdev_budget`

Note: On some systems the user need to make the rx-ring-buffer quite large or disable interrupt coalescence all together.

In addition to the "soft" interrupt coalescence that is found under /proc/sys/net/core the NIC itself will delay the hardware interrupt. User can find the settings with ethtool in the usual way. Here is an example that shows the ACQ107's default settings:

```
ethtool -c enp4s0
Coalesce parameters for enp4s0:
Adaptive RX: off
TX: off
stats-block-usecs: 0
sample-interval: 0
pkt-rate-low: 0
pkt-rate-high: 0
rx-usecs: 112
rx-frames: 0
rx-usecs-irq: 0
rx-frames-irq: 0
tx-usecs: 510
tx-frames: 0
tx-usecs-irq: 0
tx-frames-irq: 0
rx-usecs-low: 0
rx-frames-low: 0
tx-usecs-low: 0
tx-frames-low: 0
rx-usecs-high: 0
rx-frames-high: 0
tx-usecs-high: 0
tx-frames-high: 0
```

Another useful parameter is the /proc/sys/net/core/netdev_max_backlog. The backlog queue, is a FIFO on the other side of the NIC ring-buffer. Increasing the backlog buffer is one more way to add capacity earlier in the data-path. It's difficult to determine when to increase netdev_max_backlog vs increasing the rx ring-buffer. Certainly the ring-buffer is the only place where we can add capacity that can absorb traffic bursts at line rate.

Troubleshooting Advanced NICs

Advanced hardware interfaces have multiple ring-buffers that are typically mapped to different CPU cores (a technique known as RSS.) Each NIC has its own proprietary scheme for mapping input traffic flows to ring-buffers, and sometimes a NIC will incorrectly split a traffic flow into multiple FIFOs. If you see this behavior it means that the NIC itself will cause frames to be reordered in a way that will horribly disrupt the IP stack above it. The ACQ107 is one such NIC. The problem can be identified by looking at ethtool -S <ethX>. The NIC will list stats for each FIFO, and by sending a single large traffic flow we can see that device errantly split the flow into all of the different FIFOs. Below you can see that this NIC has stats labeled Queue[0] ... Queue[7].

Example:

```
ethtool -S enp4s0
NIC statistics:
InPackets: 350287807
InUCast: 350048688
InMCast: 231724
InBCast: 7395
InErrors: 0
OutPackets: 363162007
OutUCast: 363160208
```

(continues on next page)

(continued from previous page)

```
OutMCast: 1306
OutBCast: 493
InUCastOctets: 525223100117
OutUCastOctets: 545214487081
InMCastOctets: 16440320
OutMCastOctets: 206101
InBCastOctets: 1316312
OutBCastOctets: 58497
InOctets: 525240856749
OutOctets: 545214751679
InPacketsDma: 23207849
OutPacketsDma: 22064728
InOctetsDma: 34568308793
OutOctetsDma: 33164524696
InDroppedDma: 2002075
Queue[0] InPackets: 23087183
Queue[0] InJumboPackets: 0
Queue[0] InLroPackets: 0
Queue[0] InErrors: 0
Queue[0] AllocFails: 0
Queue[0] SkbAllocFails: 0
Queue[0] Polls: 7373190
Queue[0] OutPackets: 649028
Queue[0] Restarts: 0
Queue[1] InPackets: 80
Queue[1] InJumboPackets: 0
Queue[1] InLroPackets: 0
Queue[1] InErrors: 0
Queue[1] AllocFails: 0
Queue[1] SkbAllocFails: 0
Queue[1] Polls: 14672
Queue[1] OutPackets: 1651541
Queue[1] Restarts: 0
Queue[2] InPackets: 103
Queue[2] InJumboPackets: 0
Queue[2] InLroPackets: 0
Queue[2] InErrors: 0
Queue[2] AllocFails: 0
Queue[2] SkbAllocFails: 0
Queue[2] Polls: 215484
Queue[2] OutPackets: 3815296
Queue[2] Restarts: 0
Queue[3] InPackets: 269
Queue[3] InJumboPackets: 0
Queue[3] InLroPackets: 0
Queue[3] InErrors: 0
Queue[3] AllocFails: 0
Queue[3] SkbAllocFails: 0
Queue[3] Polls: 14469
Queue[3] OutPackets: 1580307
Queue[3] Restarts: 0
Queue[4] InPackets: 119681
Queue[4] InJumboPackets: 0
Queue[4] InLroPackets: 0
```

(continues on next page)

(continued from previous page)

```
Queue[4] InErrors: 0
Queue[4] AllocFails: 0
Queue[4] SkbAllocFails: 0
Queue[4] Polls: 157920
Queue[4] OutPackets: 3670607
Queue[4] Restarts: 0
Queue[5] InPackets: 83
Queue[5] InJumboPackets: 0
Queue[5] InLroPackets: 0
Queue[5] InErrors: 0
Queue[5] AllocFails: 0
Queue[5] SkbAllocFails: 0
Queue[5] Polls: 9006
Queue[5] OutPackets: 931971
Queue[5] Restarts: 0
Queue[6] InPackets: 407
Queue[6] InJumboPackets: 0
Queue[6] InLroPackets: 0
Queue[6] InErrors: 0
Queue[6] AllocFails: 0
Queue[6] SkbAllocFails: 0
Queue[6] Polls: 15387
Queue[6] OutPackets: 1636793
Queue[6] Restarts: 0
Queue[7] InPackets: 43
Queue[7] InJumboPackets: 0
Queue[7] InLroPackets: 0
Queue[7] InErrors: 0
Queue[7] AllocFails: 0
Queue[7] SkbAllocFails: 0
Queue[7] Polls: 11584
Queue[7] OutPackets: 343508
Queue[7] Restarts: 0
PTP Queue[16] InPackets: 0
PTP Queue[16] InJumboPackets: 0
PTP Queue[16] InLroPackets: 0
PTP Queue[16] InErrors: 0
PTP Queue[16] AllocFails: 0
PTP Queue[16] SkbAllocFails: 0
PTP Queue[16] Polls: 0
PTP Queue[16] OutPackets: 0
PTP Queue[16] Restarts: 0
PTP Queue[31] InPackets: 0
PTP Queue[31] InJumboPackets: 0
PTP Queue[31] InLroPackets: 0
PTP Queue[31] InErrors: 0
PTP Queue[31] AllocFails: 0
PTP Queue[31] SkbAllocFails: 0
PTP Queue[31] Polls: 0
MACSec InCtlPackets: 0
MACSec InTaggedMissPackets: 0
MACSec InUntaggedMissPackets: 23252064
MACSec InNotagPackets: 23252064
MACSec InUntaggedPackets: 0
```

(continues on next page)

(continued from previous page)

```
MACSec InBadTagPackets: 0
MACSec InNoSciPackets: 0
MACSec InUnknownSciPackets: 0
MACSec InCtrlPortPassPackets: 0
MACSec InUnctrlPortPassPackets: 23252064
MACSec InCtrlPortFailPackets: 0
MACSec InUnctrlPortFailPackets: 0
MACSec InTooLongPackets: 0
MACSec InIgpocCtlPackets: 0
MACSec InEccErrorPackets: 0
MACSec InUnctrlHitDropRedir: 0
MACSec OutCtlPackets: 1
MACSec OutUnknownSaPackets: 22064727
MACSec OutUntaggedPackets: 0
MACSec OutTooLong: 0
MACSec OutEccErrorPackets: 0
MACSec OutUnctrlHitDropRedir: 0
```

The vendor provided a workaround in their README.

Note: RSS for UDP

Currently, NIC does not support RSS for fragmented IP packets, which leads to an incorrect handling of RSS for fragmented UDP traffic. To disable RSS for UDP one can use the following RX Flow L3/L4 rule: ethtool -N eth0 flow-type udp4 action 0 loc 32

When Stats Fail

Sometimes a NIC will drop frames without any error stats incrementing. When this happens, the issue can be detected by inserting a managed L2 switch in between the sensor and the workstation. The managed switch will report receive and transmit stats, which can be correlated against the rx stats of the NIC to determine that the NIC has dropped frames without incrementing any stat.

19.7.2 IP Statistics

After the link layer the next layer up is IP. IP errors can be identified with the netstat tool:

`netstat -s`

This tool will output a lot of information, but in this document we will focus on only the IP section.

In this report you can see that there are a few different error categories, and you have to review carefully through all of the text to find them:

Let's look at each class of error and consider it's implications:

- Packets received with invalid address means that they were sent to our MAC, but with an incorrect source IP.
- Packets dropped because of missing route indicates that the packet was sent to the correct IP address but no client program was listening on the destination port.

- Fragments dropped after timeout means that we received some data but subsequent data didn't arrive in time to be processed.
- Fragments reassemblies failed means that some data was missing due to an Ethernet frame being aborted by the stack or being lost in transit and the IP layer was not able to reassemble a complete datagram.

Debugging a Layer 3 Issue

The best way to debug issues in the IP layer is to find them in the link layer, because generally speaking layer-2 issues are caused by layer-3 bugs, but this is not always the case.

For instance, packets received with invalid address are probably indicative of stale ARP table entries or some other external network bug or temporal state that will most likely clear up on its own. This sort of problem is probably not worth debugging unless its persistent. Packets dropped because of missing route is more indicative of an issue at the application layer (the client or server simply wasn't listening when the packets arrived).

If a problem is detectable by L3 and not by L2, then its most likely a problem in the NIC itself, and if the NIC isn't providing a FIFO or DMA stat that explains it. One possibility is packet reordering by the NIC. This can be detected by modifying

```
/proc/sys/net/ipv4/ipfrag_max_dist
```

This kernel attribute determines the systems tolerance to receiving out-of-order IPv4 frames. Nominally L2 networks do not reorder packets, so you should be able to configure a value of 1 and not observe a change in behavior. However, if setting a low threshold exacerbates the issue, or setting a high value makes the problem less severe then the NIC is most likely to blame.

19.7.3 Useful network debugging tools

iPerf

`iPerf` is a useful tool when debugging the performance of a network. It can be used to quickly validate whether or not a system can handle a given throughput. It can be configured to output a stream of data in a variety of formats to mimic the expected load on the system during use. For more information refer to iPerf documentation.

How to use iPerf to debug sensor network issues

`iPerf` can be used to rule out sensor failures, and quickly reproduce errors that occur when the network is under a high-traffic load. `iPerf` must be used from two machines:

- Server (receiving data)
- Client (sending data)

Both the server and client will measure the number of packets sent/received, and report a percentage of packets lost.

Example usage of `iPerf` to test sender can send 300Mbps of UDP packets of 20KB to receiver:

Receiver arguments

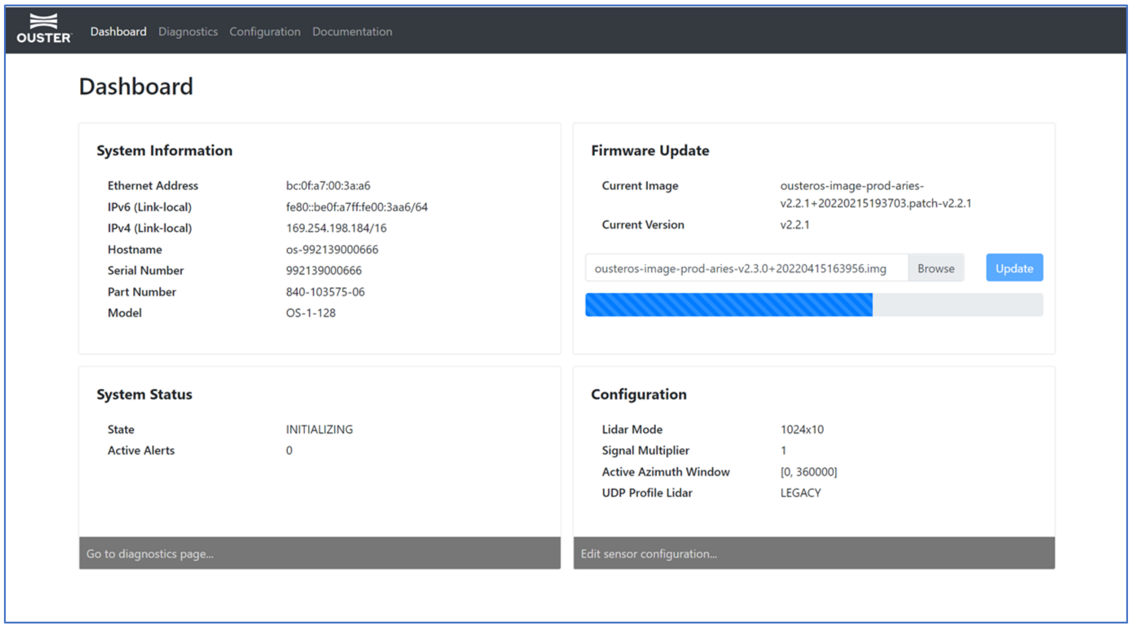
- `--server` : Required to indicate that this is the machine that will be RECEIVING data.
- `--port 5300` : Specify the port at which to listen for incoming data. Useful if testing with multiple sources simultaneously.

Sender arguments

- `--client` 192.168.88.248 : The IP address to send data to. Must be the IP address or hostname of the receiver.
- `--port 5300` : The port to send data to. This must match the -port argument provided by the receiver.
- `--udp` : Indicates that UDP traffic will be sent. If not supplied, TCP data will be sent.
- `--bitrate 300M` : The rate in (in bits per second) to send data to the receiver. This can be used to simulate different amounts of network load.This supports a suffix such as K , M , or G to indicate Kbps, Mbps, or Gbps instead of bps.
- `--length` 20K

19.8 Updating Firmware

Sensor firmware can be updated with an Ouster-provided firmware file from Ouster FW (or directly from the deployment engineering team) by accessing the sensor over http - e.g., http://os991900123456.local/ and uploading the file as prompted.



Dashboard

System Information

Ethernet Address	bc:0f:a7:00:3a:a6
IPv6 (Link-local)	fe80::be0f:a7ff:fe00:3aa6/64
IPv4 (Link-local)	169.254.198.184/16
Hostname	os-992139000666
Serial Number	992139000666
Part Number	840-103575-06
Model	OS-1-128

Firmware Update

Current Image	ousteros-image-prod-aries-v2.2.1+20220215193703.patch-v2.2.1
Current Version	v2.2.1

ousteros-image-prod-aries-v2.3.0+20220415163956.img

System Status

State	INITIALIZING
Active Alerts	0

Configuration

Lidar Mode	1024x10
Signal Multiplier	1
Active Azimuth Window	[0, 360000]
UDP Profile Lidar	LEGACY

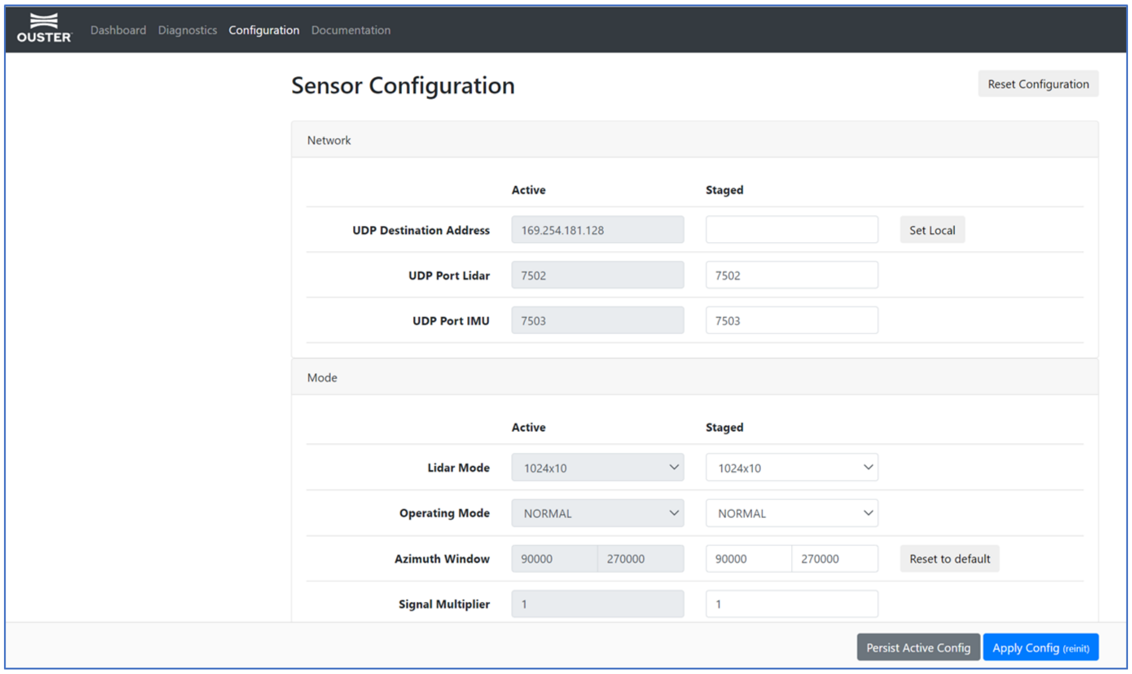
Figure 19.1: Uploading a new firmware image onto the sensor

Always check your firmware version before attempting an update. Only update to an equal or higher version number.

After the web UI confirms that the update is complete, please allow the sensor to reboot (about 2 minutes) and refresh your webpage to get access to the updated Web UI.

19.9 Downgrading Firmware

Do not roll back firmware to lower numbered versions without having been instructed to do so by Ouster. If you do, your sensor may experience issues. If your sensor is experiencing startup issues upon downgrading from v2.3.x, reset the on-sensor configuration by using the **Reset Configuration** button on Sensor Homepage.



Sensor Configuration

Network

	Active	Staged
UDP Destination Address	169.254.181.128	
UDP Port Lidar	7502	7502
UDP Port IMU	7503	7503

Mode

	Active	Staged
Lidar Mode	1024x10	1024x10
Operating Mode	NORMAL	NORMAL
Azimuth Window	90000 270000	90000 270000
Signal Multiplier	1	1

Figure 19.2: Reset Configuration