Ouster Sensor API Guide

Release v1.14.0-beta.11

Ouster Sensor TCP and HTTP API Guide

Jun 17, 2020

TCP and HTTP APIs

This on-sensor reference document is a quick guide to the TCP and HTTP APIs. For the most up-to-date and comprehensive information about your sensor, please see the sensor user manuals found at www.ouster.com

1 TCP API

1.1 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 that "os-xxx" refers to the sensor serial number and can look like "os-xxx" or "os1-xxx".

```
$ nc os-991900123456 7501
get_sensor_info
{"status": "RUNNING", "image_rev": "ousteros-image-prod-aries-v2.0.0-20200429230129",
 "base_pn": "000-101323-03", "prod_sn": "991900123456", "proto_rev": "v1.1.1",
 "base_sn": "101837000752", "prod_line": "OS-1-128", "build_rev": "v2.0.0",
 "prod_pn": "840-101855-02", "build_date": "2020-01-28T22:58:18Z"}
```

A sensor may have one of the following statuses:

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.
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.

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

The following commands will return sensor configuration and calibration information:

Table1: Sensor Configuration and Calibration

Command	Description	Response Example

get_config_txt	Returns JSON-formatted sensor configuration.	{
		"timestamp_mode": "TIME_FROM_INTERNAL_OSC",
		"multipurpose_io_mode": "OFF",
		"nmea_leap_seconds": 0,
		"lidar_mode": "1024x10",
		"sync_pulse_in_polarity": "ACTIVE_HIGH",
		"nmea_in_polarity": "ACTIVE_HIGH",
		"sync_pulse_out_polarity": "ACTIVE_HIGH",
		"udp_ip": "192.0.2.123",
		"nmea_ignore_valid_char": 0,
		"auto_start_flag": 1,
		"sync_pulse_out_pulse_width": 10,
		"nmea_baud_rate": "BAUD_9600",
		"sync_pulse_out_angle": 360,
		"sync_pulse_out_frequency": 1,
		"udp_port_imu": 7503,
		"udp_port_lidar": 7502,
		"azimuth_window": [0, 36000]
		}

continues on next page

Table 1 - continued from previous page

Command	Description	Response Example

get_sensor_info	Returns JSON-formatted sensor metadata: serial number, hardware and software revision, and sensor status.	{
		"status": "RUNNING",
		"image_rev": "ousteros-image-prod-aries-v2.0.
		↪0-20200429230129",
		"base_pn": "000-101323-03",
		"prod_sn": "991900123456",
		"proto_rev": "v1.1.1",
		"base_sn": "101837000752",
		"prod_line": "OS-1-128",
		"build_rev": "v2.0.0",
		"prod_pn": "840-101855-02",
		"build_date": "2020-01-28T22:58:18Z"
		}

continues on next page

Table 1 - continued from previous page

Command	Description	Response Example

get_time_info	Returns JSON-formatted sensor timing configuration and status of udp timestamp, sync_pulse_in, and multipurpose_io.	{
		"timestamp": {
		"time": 302.96139565999999,
		"mode": "TIME_FROM_INTERNAL_OSC",
		"time_options": {
		"sync_pulse_in": 0,
		"internal_osc": 302,
		"ptp_1588": 309
		}
		},
		"sync_pulse_in": {
		"locked": 0,
		"diagnostics": {
		"last_period_nsec": 0,
		"count_unfiltered": 1,
		"count": 0
		},
		"polarity": "ACTIVE_HIGH"
		},
		"multipurpose_io": {
		"mode": "OFF",
		"sync_pulse_out": {
		"pulse_width_ms": 10,
		"angle_deg": 360,
		"frequency_hz": 1,
		"polarity": "ACTIVE_HIGH"
		},
		"nmea": {
		"locked": 0,
		"baud_rate": "BAUD_9600",
		"diagnostics": {
		"io_checks": {
		"bit_count": 1,
		"bit_count_unfilterd": 0,
		"start_char_count": 0,
		"char_count": 0
		},
		"decoding": {
		"last_read_message": "",
		"date_decoded_count": 0,
		"not_valid_count": 0,
		"utc_decoded_count": 0
		}
		},
		"leap_seconds": 0,
		"ignore_valid_char": 0,
		"polarity": "ACTIVE_HIGH"
		}
		}
		}

continues on next page

Table 1 - continued from previous page

Command	Description	Response Example

get_beam_intrinsics	Returns JSON-formatted beam altitude and azimuth offsets, in degrees. Length of arrays is equal to the number of channels in sensor. Also returns distance between lidar origin and beam origin in mm, to be used for point cloud calculations.	{
		"lidar_origin_to_beam_origin_mm": 13.762,
		"beam_altitude_angles": [
		16.926,
		16.313,
		"...",
		-16.078,
		-16.689
		],
		"beam_azimuth_angles": [
		3.052,
		0.857,
		"...",
		-0.868,
		-3.051
		]
		}

get_imu_intrinsics	Returns JSON-formatted IMU transformation matrix needed to adjust 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
		]
		}

continues on next page

Table 1 - continued from previous page

Command	Description	Response Example

get_lidar_intrinsics	Returns JSON-formatted lidar transformation matrix needed to adjust 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
		]
		}

continues on next page

Table 1 - continued from previous page

Command	Description	Response Example

`get_alerts <START_CURSOR>`	Returns JSON-formatted sensor diagnostic information.
	The `log` list contains Alerts when they were activated or deactivated. An optional `START_CURSOR` argument specifies where the log should start.
	The `active` list contains all currently active alerts.
		{
		"next_cursor": 2,
		"active": [
		{
		"category": "UDP_TRANSMISSION",
		"msg": "Received an unknown error when trying to send lidar data UDP packet; closing socket.",
		"realtime": "1569631015375767040",
		"cursor": 0,
		"id": "0x01000017",
		"active": true,
		"msg_verbose": "192.0.2.123:7502",
		"level": "WARNING"
		},
		],
		"log": [
		{
		"category": "UDP_TRANSMISSION",
		"msg": "Received an unknown error when trying to send lidar data UDP packet; closing socket.",
		"realtime": "1569631015375767040",
		"cursor": 0,
		"id": "0x01000017",
		"active": true,
		"msg_verbose": "192.0.2.123:7502",
		"level": "WARNING"
		},
		{
		"category": "UDP_TRANSMISSION",
		"msg": "Received an unknown error when trying to send IMU UDP packet; closing socket.",
		"realtime": "1569631015883802368",
		"cursor": 1,
		"id": "0x0100001a",
		"active": false,
		"msg_verbose": "192.0.2.123:7503",
		"level": "WARNING"
		}
		]
		}

continues on next page

Table 1 - continued from previous page

Command	Description	Response Example

get_lidar_data_format	Returns JSON-formatted response that describes the structure of a lidar packet.
	columns_per_frame: Number of azimuth columns per frame. This can be 512, 1024, or 2048, depending upon the set lidar mode.
	columns_per_packet: Number of azimuth blocks contained in a single lidar packet. Currently in v1.14 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.
		<pre>{
		  "columns_per_frame": 1024,
		  "columns_per_packet": 16,
		  "pixel_shift_by_row": [
		  18,
		  12,
		  6,
		  0,
		  "...",
		  18,
		  12,
		  6,
		  0
		  ],
		  "pixels_per_column": 128
		}</pre>

1.2 Querying Active or Staged Parameters

Sensor configurations / operating modes can also be queried over TCP. Below is the latest 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 or after sensor reset.

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

```
$ nc os1-991900123456 7501
get_config_param active lidar_mode
1024x10
```

The following commands will return sensor active or staged configuration parameters:

Table2: Sensor Configurations

get_config_param	Command Description	Response
udp_ip	Returns the ip address to which the sensor sends UDP traffic.	"" (default)
udp_port_lidar	Returns the port number of lidar UDP data packets.	7502 (default)
udp_port_imu	Returns the port number of IMU UDP data packets.	7503 (default)
lidar_mode	Returns a string indicating the horizontal resolution and rotation frequency [Hz].	One of 512x10, 1024x10, 2048x10, 512x20, or 1024x20
timestamp_mode	Returns the method used to timestamp measurements.	One of TIME_FROM_INTERNAL_OSC, TIME_FROM_PTP_1588, TIME_FROM_SYNC_PULSE_IN
nmea_in_polarity	Returns the polarity of NMEA UART input $GPRMC messages. See time synchronization section in sensor user guides for NMEA use case. Use ACTIVE_HIGH if UART is active high, idle low, and start bit is after a falling edge.	One of ACTIVE_HIGH (default) or ACTIVE_LOW
nmea_ignore_valid_char	Returns 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.	0 (default) or 1
nmea_baud_rate	Returns BAUD_9600 (default) or BAUD_115200 for the expected baud rate the sensor is attempting to decode for NMEA UART input $GPRMC messages.	One of BAUD_9600, BAUD_115200
nmea_leap_seconds	Returns the 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.	0 (default)

continues on next page

Table 2 - continued from previous page

get_config_param	Command Description	Response
auto_start_flag	Returns the 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.	0 (default)

Table3: Sensor Modes

Command	Command Description	Response
multipurpose_io_mode	Returns the configured mode of the MULTIPURPOSE_IO pin. See Time Syncrhonization section in sensor user guide for a detailed description of each option.	One of OFF (default), INPUT_NMEA_UART, OUTPUT_FROM_INTERNAL_OSC, OUTPUT_FROM_SYNC_PULSE_IN, OUTPUT_FROM_PTP_1588, or OUTPUT_FROM_ENCODER_ANGLE
sync_pulse_out_polarity	Returns the polarity of SYNC_PULSE_OUT output, if sensor is using this for time synchronization.	One of ACTIVE_HIGH or ACTIVE_LOW (default)
sync_pulse_out_frequency	Returns the output SYNC_PULSE_OUT pulse rate in Hz.	1 (default)
sync_pulse_out_angle	Returns the output SYNC_PULSE_OUT pulse rate defined in rotation angles.	360 (default)
sync_pulse_out_pulse_width	Returns the output SYNC_PULSE_OUT pulse width in ms.	10 (ms, default)

1.3 Setting Configuration Parameters

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

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

`write_config_txt` 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 `write_config_txt` capture it to take effect on reset.

`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_ip <ip address>`.

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

```
$ nc os1-991900123456 7501
set_config_param lidar_mode 512x20
set_config_param
set_udp_dest_auto
set_udp_dest_auto
reinitialize
reinitialize
write_config_txt
write_config_txt
```

The following commands will set sensor configuration parameters:

Table4: Setting Config Params

set_config_param	Command Description	Response
`udp_ip <ip address>`	Set the <ip address> to which the sensor sends UDP traffic. On boot, the sensor will not output data until this is set. If the IP address is not known, this can also be accomplished with the `set_udp_dest_auto` command (details above). The sensor supports unicast, IPv4 broadcast (`255.255.255.255`), and IPv6 multicast (`ff02::01`) addresses.	`set_config_param` on success, `error:` otherwise
`udp_port_lidar <port>`	Set the <port> on udp_ip to which lidar data will be sent (`7502`, default).	`set_config_param` on success, `error:` otherwise
`udp_port_imu <port>`	Set the <port> on udp_ip to which IMU data will be sent (`7503`, default).	`set_config_param` on success, `error:` otherwise

continues on next page

Table 4 - continued from previous page

set_config_param	Command Description	Response
`lidar_mode <mode>`	Set the horizontal resolution and rotation rate of the sensor. Valid modes are `512x10`, `1024x10`, `2048x10 512x20`, `1024x20`. Each 50% the total number of points gathered is reduced (e.g., from 2048x10 to 1024x10) extends range by 15-20%.	`set_config_param` on success, `error:` otherwise
`timestamp_mode <mode>`	Set the method used to timestamp measurements. Valid modes are `TIME_FROM_INTERNAL_OSC`, `TIME_FROM_SYNC_PULSE_IN`, or `TIME_FROM_PTP_1588`.	`set_config_param` on success, `error:` otherwise
`sync_pulse_in_polarity <1/0>`	Set 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`.	`set_config_param` on success, `error:` otherwise
`nmea_in_polarity <1/0>`	Set the polarity of NMEA UART input $GPRMC messages. See Time Synchronization section in sensor user guide for NMEA use case. Use `ACTIVE_HIGH` if UART is active high, idle low, and start bit is after a falling edge.	`set_config_param` on success, `error:` otherwise
`nmea_ignore_valid_char <1/0>`	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.	`set_config_param` on success, `error:` otherwise
`nmea_baud_rate <rate in baud/s>`	Set `BAUD_9600` (default) or `BAUD_115200` for the expected baud rate the sensor is attempting to decode for NMEA UART input $GPRMC messages.	`set_config_param` on success, `error:` otherwise
`nmea_leap_seconds <s>`	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`.	`set_config_param` on success, `error:` otherwise
`multipurpose_io_mode <mode>`	Configure the mode of the MULTIPURPOSE_IO pin. Valid modes are `OFF`, `INPUT_NMEA_UART`, `OUTPUT_FROM_INTERNAL_OSC`, `OUTPUT_FROM_SYNC_PULSE_IN`, `OUTPUT_FROM_PTP_1588`, or `OUTPUT_FROM_ENCODER_ANGLE`.	`set_config_param` on success, `error:` otherwise

continues on next page

Table 4 - continued from previous page

set_config_param	Command Description	Response
`auto_start_flag <1/0>`	Set `1` to ensure that the *next* time the sensor starts up, it will stay in the dormant state until it is manually commanded to start spinning and firing lasers again. It will **not** put the sensor into the dormant state while it is already running. To use the `auto_start_state` configuration, sensor must be power cycled after setting this flag. When it comes up again, it will be in the `UNCONFIGURED` state and will not lase or spin. Running the following commands will bring it back into the `RUNNING` state: `set_config_param auto_start_flag 1`, `reinitialize`, `write_config_txt`. This mode is useful for power-sensitive applications, e.g. drone flights.	`0` (default)

Table5: Setting Sync

set_config_param	Command Description	Response
`sync_pulse_out_polarity <1/0>`	Set the polarity of SYNC_PULSE_OUT output, if sensor is set as the master sensor used for time synchronization.	`set_config_param` on success, `error:` otherwise
`sync_pulse_out_frequency <rate in Hz>`	Set output SYNC_PULSE_OUT rate. Valid inputs are integers > 0 Hz, but also limited by the criteria described in the Time Syncrhonization section of the sensor user manual.	`set_config_param` on success, `error:` otherwise
`sync_pulse_out_angle <angle in deg>`	Set output SYNC_PULSE_OUT rate defined by rotation angle. Valid inputs are integers up to 360 degrees but also limited by the criteria described in the Time Syncrhonization section of sensor user guide.	`set_config_param` on success, `error:` otherwise

continues on next page

Table 5 - continued from previous page

set_config_param	Command Description	Response
`sync_pulse_out_pulse_width <width in ms>`	Set output SYNC_PULSE_OUT pulse width in ms, in 1 ms increments. Valid inputs are integers greater than 0 ms, but also limited by the criteria described in the Time Syncrhonization section of sensor user guide.	`set_config_param` on success, `error:` otherwise

Table6: Reinitialize, Write Configuration, & Auto Destination UDP

Command	Command Description	Response
`reinitialize`	Restarts the sensor. Changes to lidar, multipurpose_io, and timestamp modes will only take effect after reinitialization.	`reinitialize` on success
`write_config_txt`	Make the current settings persist after reboot.	`write_config_txt` on success
`set_udp_dest_auto`	Set the destination of UDP traffic to the IP address that issued the command.	`set_udp_dest_auto` on success

2 HTTP API Reference

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

API Resources

- `system/firmware`
- `system/network`
- `system/time`

2.1 system/firmware

GET system/firmware

```
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"
}
```

→

Response JSON Object

- fw (string) - Running firmware image name and version.

Status Codes

- 200 OK - No error

2.2 system/network

GET 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": "os1-991900123456",
   "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
}
```

→

Response JSON Object

- `carrier` (`boolean`) - State of Ethernet link, `true` when physical layer is connected.
- `duplex` (`string`) - Duplex mode of Ethernet link, `half` or `full`.
- `ethaddr` (`string`) - Ethernet hardware (MAC) address.
- `hostname` (`string`) - Hostname of the sensor, also used when requesting *DHCP* lease.
- `ipv4` (`object`) - See *ipv4 object*
- `ipv6.link_local` (`string`) - Link-local IPv6 address.
- `speed` (`integer`) - Ethernet physical layer speed in Mbps, should be 1000 Mbps.

Status Codes

- 200 OK - No error

`GET 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
}
```

→

Response JSON Object

- `addr` (`string`) - Current global or private IPv4 address.
- `link_local` (`string`) - Link-local IPv4 address.
- `override` (`string`) - Static IP override value, this should match `addr`. This value will be `null` when unset and operating in *DHCP* mode.

Status Codes

- 200 OK - No error

`GET 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
```

→

Response JSON Object

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

Status Codes

- 200 OK - No error

`PUT 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 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.

Static IP override should only be used in special use cases. The dynamic *DHCP* 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"
```

→

Request JSON Object

- `string` - Static IP override value with subnet mask

Response JSON Object

- `string` - Static IP override value that system will set after a short delay.

Status Codes

- 200 OK - No error

`DELETE 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 *DHCP* lease is obtained from a network *DHCP* server.

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

→

Status Codes

- 204 No Content - No error, no content

2.3 system/time

GET system/time

Get the system time configuration for all timing components of the sensor.

```
GET /api/v1/system/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": 37950,
      "offset_from_master": -211488,
      "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": {
      "current_utc_offset": 37,
      "current_utc_offset_valid": 1,
      "frequency_traceable": 1,
      "leap59": 0,
      "leap61": 0,
      "ptp_timescale": 1,
      "time_source": 32,
```

(continues on next page)

(continued from previous page)


```
      "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": 1552413985821448000,
      "last_gm_phase_change": "0x0000'0000000000000000.0000",
      "master_offset": -211488,
      "scaled_last_gm_phase_change": 0
    }
  },
  "sensor": {
    "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_unfilterd": 0,
          "char_count": 0,
          "start_char_count": 0
        }
      },
      "ignore_valid_char": 0,
      "leap_seconds": 0,
      "locked": 0,
      "polarity": "ACTIVE_HIGH"
    },
    "sync_pulse_in": {
      "diagnostics": {
        "count": 1,
        "count_unfiltered": 0,
        "last_period_nsec": 0
      },
      "locked": 0,
      "polarity": "ACTIVE_HIGH"
    },
    "sync_pulse_out": {
      "angle_deg": 360,
      "frequency_hz": 1,
      "mode": "OFF",
      "polarity": "ACTIVE_HIGH",
      "pulse_width_ms": 10
    },
    "timestamp": {
      "mode": "TIME_FROM_INTERNAL_OSC",
      "time": 57178.44114677,
```


(continues on next page)

(continued from previous page)


```
      "time_options": {
        "internal_osc": 57178,
        "ptp_1588": 1552413986,
        "sync_pulse_in": 1
      }
    }
  },
  "system": {
    "monotonic": 57191.819600378,
    "realtime": 1552413949.3948405,
    "tracking": {
      "frequency": -7.036,
      "last_offset": 5.942e-06,
      "leap_status": "normal",
      "ref_time_utc": 1552413947.8259742,
      "reference_id": "70747000",
      "remote_host": "ptp",
      "residual_frequency": 0.006,
      "rms_offset": 5.358e-06,
      "root_delay": 1e-09,
      "root_dispersion": 0.000129677,
      "skew": 1.144,
      "stratum": 1,
      "system_time_offset": -2.291e-06,
      "update_interval": 2
    }
  }
}
```


→

Response JSON Object

- `string` - See sub objects for details.

Status Codes

- 200 OK - No error

`GET system/time/system`

Get the operating system time status. These values relate to the operating system clocks, and not clocks related to hardware timestamp data from the lidar sensor.

```
GET /api/v1/system/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,
```


(continues on next page)

(continued from previous page)


```
    "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
  }
}
```


→

Response JSON Object

- `monotonic` (`float`) - Monotonic time of operating system. This timestamp never counts backwards and is the time since boot in seconds.
- `realtime` (`float`) - Time in seconds since the Unix epoch, should match wall time if synchronized with external time source.
- `tracking` (`object`) - Operating system time synchronization tracking status. See chronyc tracking documentation for more information.

Status Codes

- 200 OK - No error

System `tracking` fields of interest:

→

Rms_offset Long-term average of the offset value.
System_time_offset Time delta (in seconds) between 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.

`GET system/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/system/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,
```


(continues on next page)

(continued from previous page)


```
    "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": {
    "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
  }
}
```


→

Response JSON Object

- `current_data_set` (`object`) - Result of the PMC `GET CURRENT_DATA_SET` command.
- `parent_data_set` (`object`) - Result of the PMC `GET PARENT_DATA_SET` command.
- `port_data_set` (`object`) - Result of the PMC `GET PORT_DATA_SET` command.
- `time_properties_data_set` (`object`) - Result of the PMC `GET TIME_PROPERTIES_DATA_SET` command.

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

Status Codes

- 200 OK - 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 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

`GET system/time/sensor`

Get the lidar sensor time status. These values relate to the hardware timestamping mechanism of the sensor.

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

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

{
  "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_unfilterd": 0,
        "char_count": 0,
        "start_char_count": 0
      }
    },
```

(continues on next page)

(continued from previous page)


```
    "ignore_valid_char": 0,
    "leap_seconds": 0,
    "locked": 0,
    "polarity": "ACTIVE_HIGH"
  },
  "sync_pulse_in": {
    "diagnostics": {
      "count": 1,
      "count_unfiltered": 0,
      "last_period_nsec": 0
    },
    "locked": 0,
    "polarity": "ACTIVE_HIGH"
  },
  "sync_pulse_out": {
    "angle_deg": 360,
    "frequency_hz": 1,
    "mode": "OFF",
    "polarity": "ACTIVE_HIGH",
    "pulse_width_ms": 10
  },
  "timestamp": {
    "mode": "TIME_FROM_INTERNAL_OSC",
    "time": 57178.44114677,
    "time_options": {
      "internal_osc": 57178,
      "ptp_1588": 1552413986,
      "sync_pulse_in": 1
    }
  }
}
```


For more information on these parameters refer to the `get_time_info` TCP command.

3 Change Log

→

Version v1.6.0

Date 2018-08-16

Description

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

→

Version v1.7.0

Date 2018-09-05

Description

- 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 v1.9.0

Date 2018-10-24

Description

- No TCP command change.

→

Version v1.10.0

Date 2018-12-11

Description

- Remove all references of `pulse_mode`.
- Add `get_alerts`, `pps_rate` and `pps_angle` usage commands and expected output.

- Remove TCP commands prior to v1.5.1.

→

Version v1.11.0

Date 2019-03-25

Description

- Add section on HTTP API commands.
- TCP Port now hardcoded to 7501; port is no longer configurable.
- Update to SYNC_PULSE_IN and MULTIPURPOSE_IO interface and configuration parameters (see details below).

Details on interface changes:

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 v1.12.0

Date

Description

- 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 v1.13.0

Date

Description

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

→

Version v1.14.0

Date 2018-02-16

Description

- Add TCP command `get_lidar_data_format`
- TCP command `get_beam_intrinsics` now returns the the altitude and azimuth angle arrays of only the active channels. Length of the two arrays is now equal to the number of channels in the sensor.
- TCP command `get_beam_intrinsics` also includes new parameter `lidar_origin_to_beam_origin_mm`
- Add in additional details to `auto_start_flag` parameter

HTTP Routing Table

/system