QORTEX™ DTC 2.3 User Guide

Notices

QUANERGY, the QUANERGY logo, QORTEX, QORTEX DTC and associated logos, QORTEX MXP, QORTEX PEOPLE COUNTER, QORTEX Aware, M1, M1 EDGE, M8, MQ, S3, Q-GUARD, Q-VIEW, QSPU, QPU-L7, QPU MINI, MULTI-LIDAR FUSION, SENSORFUSION, Quanergy Insights are all trademarks of Quanergy Systems, Inc. All other trademarks are the property of their respective owners.

See also:

- Acknowledgments of copyrighted material: https://downloads.quanergy.com/License.txt
- End User Software License Terms that apply to all platforms hosting the Q-View software: https://downloads.quanergy.com/quanergy_end_user_software_license_terms.pdf

ISO 9001:2015 Certified

Contact

Quanergy Systems, Inc.
433 Lakeside Drive
Sunnyvale, CA 94085
https://quanergy.com/

- For purchases made directly from Quanergy: contact support@quanergy.com
- For purchases from a third party such as value-added reseller/system integrator: contact them for support

Follow Us!

https://www.linkedin.com/company/quanergy/

https://twitter.com/quanergy/

https://www.facebook.com/quanergy/

https://www.youtube.com/c/QuanergySystems/

Revision History

Version	Date	What Changed	Change Location
A	08/09/18	Released to support QORTEX DTC 2.0 version Beta 1.	
B	03/28/19	Released to support QORTEX DTC 2.0 version Beta 2.	
		Changed title and software name of Q-Guard to QORTEX DTC 1.x.	*Notices* (2)
		ROS, RViz, Continuous Recording, and Terminating Processes.	Sections removed.
		Installation and licensing	*LiDAR Sensor Prerequisites* (31), *Environment Prerequisites* (30), *Install QORTEX DTC Client Software* (43), *Managing the License* (45), *Download QORTEX DTC Server Software and Documents* (37)
		Start, stop, and interface	*Getting Started* (23), *Starting and Stopping QORTEX DTC* (56), *QORTEX DTC Client Interface* (74)
		Using Qortex DTC	*Visualizing Data* (114), *View Recorded (PLAYBACK) Data* (156), *Recording Data* (153), and *Troubleshooting* (160)
		API	*API for Remote Procedure Calls* (172)
C	07/19/19	Released to support QORTEX DTC 2.0 version Beta 3.	
		QSPU mini	Content removed
		QPU	Content removed version N
		Zone import, Zone create	*Establishing Zones and* Counter Lines (105)
		gRPC API	*API for Remote Procedure Calls* (184)
D	02/20/20	Released to support QORTEX DTC 2.0, feature complete version.	
		Installation and licensing	Updated: *Contact* (2), *LiDAR Sensor Prerequisites* (31), *Object and Capacity Specifications* (26), *Installing QORTEX DTC Server and Client* (37), *QORTEX DTC Client Host Computer* (34), *Download QORTEX DTC Server Software and Documents* (37), *Managing the License* (45) New: *Table 2. QORTEX DTC Server Specifications*

Version	Date	What Changed	Change Location
		Start, stop, and interface, viewing controls	*Getting Started* (23), *Start the Server Daemon Service the First Time* (56), *QORTEX DTC Client Interface* (74)
		Using Qortex DTC, zones, zone widgets, sensor settings, playback, refresh tracks	*View Recorded (PLAYBACK) Data* (156), *Establishing Zones and* Counter Lines (105), Sensor Health Status Options (103), *Visibility Menu* (82), *Visualize Playback Data in Loopback Mode* (158), *Refresh Tracks* (158)
		Troubleshooting and status	*Troubleshooting* (160), *Status and Error Messages* (162), *State List* (179), *Settings File Error* (161)
		API, parameters, Cartesian reference	*API for Remote Procedure Calls* (172), and *Template Parameter Settings* (184), *Cartesian Coordinate System* (274)
E	08/20/20	Released to support QORTEX DTC 2.0 with MQ-8 sensors.	
		VMS plug-in, Essentials license, and primary sensor.	Content removed
		New feature reference	*QORTEX DTC Fast Track* (310), *Automated ID Handover* (294)
		API, parameters, Cartesian reference	*API for Remote Procedure Calls* (172), and *Template Parameter Settings* (184), *Cartesian Coordinate System* (274)
F	10/07/20	Released to support QORTEX DTC 2.0 with client on Linux.	
		Install on Ubuntu	*Install on Ubuntu* (44), *Install or Update QORTEX DTC with a Debian Package* (40)
		Start, Stop client	*Start the Client on an Ubuntu Host* (59), *Stop the Client* (61)
		Client grid cell, object list	*Grid Cell* (120), *Object List* (174)
G	12/29/20	Release to support QORTEX DTC 2.1 Beta.	
		Install, Ubuntu 20.04, license PTZ camera and rules, TCP publishing	*Installing QORTEX DTC Server and Client* (37). *Managing the License* (45), *TCP Publishing Format* (92), *Environment Prerequisites* (30), *QORTEX DTC Client Host Computer* (34)
		PTZ cameras	*Adding PTZ Cameras* (125)
		Rules recording, PTZ cameras, network actions	*Adding Rules* for Event Zones (142)
		Event zone sector options	*Add a Zone*/Line (107)
		Recording in PTZ camera video	*Recording Data* (153)

Version	Date	What Changed	Change Location
		gRPC APIs for PTZ camera	*API for Remote Procedure Calls* (184)
		Guide restructure	Merged Prerequisites with Preparing chapter. Moved visualization tasks from Client to Visualization chapter. Created chapters for Configuration, Sensors, PTZ cameras, Recording, and Rules. Applied new template.
H	03/19/21	Release QORTEX DTC 2.1.	
		Added to gRPC API descriptions for PTZ camera and rules.	*API for Remote Procedure Calls* (184)
		Added Docker server installation.	*Install QORTEX DTC in a Docker Container* (41)
		Added to supported cameras: AXIS Q6115-E PTZ Network Camera	*Supported PTZ Cameras* (125), *Prepare the PTZ Camera* (126)
		Updated sections.	*Calibrate a PTZ Camera* (129), *Set PTZ Camera Home Location* (135), *Sort Rule Priority* (151)
I	4/20/21	Rolling update release QORTEX DTC 2.1	
		Add camera support: Hikvision DS-2DF8336IV-AEL, Bosch AUTODOME IP starlight 5000i.	*Supported PTZ Cameras* (125)
		Add visualization button: Grid Cell, Grid Area.	*Visualizer Controls* (81), *Grid Cell* (120), *Grid Area* (120)
		Updated section.	*Start the Client on an Ubuntu Host* (59)
I	5/12/21	Rolling update release QORTEX DTC 2.1	
		Add camera support: Hikvision DS-2DE5225IW-AE, Bosch MIC IP starlight 7100i.	*Supported PTZ Cameras* (125)
		Add camera following distance.	*Add PTZ Camera Field of View* (133)
J	5/24/21	Release QORTEX DTC 2.1.	
		Add Bosch firmware requirement.	*Supported PTZ Cameras* (125)
		Edited port number and added note for left vs right handed coordinates.	*Trackable List (Port 17161)* (175), *Cartesian Coordinate System* (292)
K Beta	06/12/21	QORTEX DTC 2.2 Beta.	
		Update overview	*QORTEX DTC 2.3 Improvements* (23)
		Added Basic Settings, a subset of Advanced Settings.	*Adjust Sensor Settings* (96), *GetSimplifiedSettings* (236)

Version	Date	What Changed	Change Location
		PTZ time switching	*Add Object Stitching Rule* (145), *GetCameraTiming API* (218)
		Tracking objects through PTZ cameras.	*Add PTZ Camera* (148)
		Object trail line in Visualizer.	*Central Visualizer Window Elements* (77)
		Vehicle subcategories.	*Activate a License* (47), *Central Visualizer Window Elements* (77), *Object and Capacity Specifications* (26), *Trackable List (Port 17161)* (175)
		Glass reflection. Exclusion zone by sensor.	*Enable Exclusion Zones by Sensor* (112)
		Recording folder labels use UTC.	*Specify Recording Settings* (153)
K Beta 2	09/11/21	QORTEX DTC 2.2 Beta 2.	
		Update overview	*QORTEX DTC 2.3 Improvements* (23), *Process Setup Overview* (27)
		Vehicle sub-class definitions.	*Visualizer Sub-Vehicles* (80)
		New object colors.	*Central Visualizer Window Elements* (77)
		Scheduling Rules.	*Add Network Actions Rule* (143), *Add Recording Rule* (146), *Add PTZ Camera Automatic Tracking Rule* (148)
		Cybersecurity.	*Security Mode* (64), *Enter Configuration Mode* (85), *Process Setup Overview* (27), *Start the Client on a Windows Host* (58), *Start the Client on an Ubuntu Host* (59), *Configuration Mode Options* (84), *Add Network Action Rules* (143), *Automated ID Handover* (310)
		secure.vault	*Start the Server Daemon Service the First Time* (56), *Change User Credentials for Security Mode* (65)
		Inclusion zones.	*Establishing Zones and* Counter Lines (105), *Process Setup Overview* (27), *Visualizer Controls* (81), *Configuration Mode Options* (84)
		TCP messages.	*Encrypted TCP Port Messages* (306)
		Settings with sensor calibration updates.	*Add or Update Multiple Sensors at a Location* (89)
		Zone widgets.	*Common Zone/Line Widgets* (110)
		Zone editing.	*Edit a Zone*/Line (109)

Version	Date	What Changed	Change Location
		Re-calibration keep existing zone and camera configuration.	*Add or Update Multiple Sensors at a Location* (89)
		Unknown PTZ camera custom settings.	*Edit PTZ Camera Custom Settings* (136)
		Automated ID Handling.	*Working with Secure Mode Functionalities* (319)
		Beta Only Configuration.	*Download QORTEX DTC Server Software and Documents* (37), *Install QORTEX DTC Client Software* (43)
K	11/05/21	QORTEX DTC 2.2 Release	
		Download and install QORTEX DTC	*Download QORTEX DTC Server Software and Documents* (37), *Install QORTEX DTC Client Software* (43)
		Cybersecurity API	*Cybersecurity* (255), *Security APIs* (230)
		TCP Listener update	*QORTEX DTC TCP Listener with Security Enabled* (264)
		User log	*Table 20. User Log Statements*
		Edit zone options	*Edit a Zone/Line* (109)
		Simultaneous objects	*Object and Capacity Specifications* (26)
		AIDH update	*Automated ID Handover* (310)
L	03/02/22	QORTEX DTC 2.2 update	
		Added product release updates	*23QORTEX DTC 2.3 Improvements* (23)
		Quanergy download links updated to `https`	*Documentation* (29), *Installing QORTEX DTC Server and Client* (37)
		Clarification Automated ID Handover compatibility	*Object_handover Parameter* (316), *Working with Secure Mode Functionalities* (319)
M	07/05/22	QORTEX DTC 2.2.1 update	
		QORTEX DTC server IP address alias	*Automated ID Handover* > *Configuration* (313)
		AIDH server failure resilience	*Automated ID Handover* (310)
N	09/20/22 10/27/22	QORTEX DTC 2.3 update	
		IPv6 support	*Environment Prerequisites* (30), *Locate IP Address of Linux Host* (35), *Locate IP Address for Windows Host* (36), *Start the Client on a Windows Host* (58), *Start the Client on an Ubuntu Host* (59), *Login to Client for the First Time* (59), *Connect the*

Version	Date	What Changed	Change Location
			Client to the Server (61), *Add Single-Sensor to a Location* (88), *Add or Update Multiple Sensors at a Location* (89)
		Ubuntu 20.04 support	*QORTEX DTC Server Host Computer*, (32) *Install on Ubuntu* (44)
		HTTP API updates	*HTTP User Security Options* (263)
		Removed support for QPU-L7 and QSPU servers.	*Object and Capacity Specifications* (26), *QORTEX DTC Server Host Computer* (32)
		Sensor anti-masking detection.	*Advanced Setting: Anti-Masking Detector Settings,* (98), *Server TCP Ports to Monitor*, (92), *Figure 105. Protobuf Output Format for QORTEX DTC 2.x State List*
		FLIR Systems Saros DM-Series PTZ Camera support	*Supported PTZ Cameras* (125)
		Set world coordinate axis for system	*Visualizer Controls* (81), *Figure 38. Client Bottom Bar*, *World Coordinate Point Cloud Alignment*, (115)
		PointCloud grid view 3D camera controls.	*Table 6. Visualizer Elements and Colors*
		Trackable object updates.	
		• Counter Line new feature	*Establishing Zones and Counter Lines* (105), *Counter Line APIs* (202), *Counter Line APIs* (202)
		• Classification types in zones.	*Zone List* (177)
		• Object Stitching. Detect trackable IDs and merge duplicates.	*Event Zone Rule Type: Object Stitching* (228), *Add Object Stitching Rule* (145)
		Advanced settings updates.	
		• Movement threshold setting for trackable objects.	*Advanced Setting: Movement Threshold* (99)
		• ML SubVehicle classification method.	*Advanced Setting: SubVehicle Classification Method* (100)
		• Vehicle forking/splitting.	*Advanced Setting: Object Forking and Splitting Parameters* (101)
		• Stationary vehicles bouncing.	*Advanced Setting: Stabilize Stationary Vehicles Settings* (102)
		• Fusing point clouds.	*Advanced Setting: Fuse Point Cloud Pipeline Settings* (102)

Version	Date	What Changed	Change Location
		TCP updates.	
		• NDJSON support	*TCP Publishing Format* (92), *Counter Line TCP Publisher Output in NDJSON* (208)
		• View TCP publishing settings	*Edit TCP Port Publishing Settings* (94), *Adjust Sensor Settings* (96)
		Security and user management.	
		• Multiple user types: viewer, editor, admin • Credentials formats	*User Management* (68)
		• Add, Edit, Delete, Disable, Enable users and user credentials	*Login to Client for the First Time* (59), *Create New Users* (69), *Change Usernames and Roles* (71), *Disable and Enable Users* (72), *Delete Users* (72)
		• Enable/Disable security user credentials	*Change User Credentials for Security Mode* (65), *Toggle Security Mode On*/Off (64), *Set User Password Expiry* (70)
		• Credential recovery	*Recover Admin Credentials with Security Mode Enabled* (73)
		Log file updates.	
		• Log file access. User activity log retention.	*User Activity Log* (165), *Logs and Configuration Files* (165)
		• Add timestamp to unique object ID.	*Visibility: Zones/Lines and Object ID* (117)
		• Logging rotation, roll active log file to a new file.	*Log File Rotation* (167)
		• Zip file used for downloading user access log files	*Download User Log Files* (165)
		• Download log files through API	*GetFile API* (243)

Text Styles

In this guide certain fonts are applied to provide a visual means to interpret text. See *Table 1. Text Styles and Meanings*.

Table 1. Text Styles and Meanings

Font Style	Meaning
Blue underline	Hyperlink that opens a file outside of this document. Typically, this is a web page.
Italic underline	Hyperlink that moves you to a location within the document. Typically, this is a section, table, or figure.
Bold	General term for emphasis. Typically, this introduces a definition or description.
Bold	An item in a Graphical User Interface (GUI). This includes page, window, or panel labels, fields where you enter or select information, or checkboxes and buttons you click.
Bold colors	Examples of colors used in GUIs.
Italic	Identifies book or section titles. Provides emphasis for terms or ideas. Also identifies options for selecting from a menu, entering in a field, or replacing in a command string.
`Code italic`	Identifies a variable, where the intent is you provide a literal value in place of the variable. Used in paragraphs and code strings.
`Code`	Examples of commands you enter in a field or terminal, or responses from the system.
`Code on gray`	Examples of commands you enter into a command line interface (CLI) terminal or responses from the system.
Text in box	Notes and Tips. Useful related information. Format used to call attention to the content or to describe side-bar content.
Text in box	Cautions and Warnings. Read these and comply with the information provided. Comply with Cautions or Warnings to prevent equipment damage or injury to humans and other living things.

Contents

Figures

Tables

1. Getting Started

QORTEX DTC™ 2.3 is the software portion of a LiDAR-based solution that provides three-dimensional perception and volumetric sensing. This system detects, tracks, and classifies (DTC) person and vehicle objects to:

- Accurately locates objects to be tracked uniquely in real time.
- Customize Event zones with flexible commands.

With data intelligence and centimeter-level accuracy, QORTEX DTC provides a cost-effective solution enabling Internet of Things (IoT) applications in different sectors, such as security and smart cities and spaces with interests in monitoring traffic and crowd flow and perimeter intrusion.

QORTEX DTC 2.3 Improvements

QORTEX DTC 2.3 added and improved the following features:

- Ubuntu 20.04 and IPv6 Support
- Zone Object Count by Classification Type
- Counter Line Element
- Anti-Masking Detection
- TCP publisher with NDJSON format
- Movement Threshold Filter
- Object Stitching
- Vehicle sub classification
- Vehicle forking and splitting
- Stationary vehicles bouncing
- Early Fusion of Point Cloud
- MQ-8 POE Rev A3 Support
- User Login and Permissions, User Log File updates, and admin Password Recovery
- Settings File Selection, TCP publishers settings display when location is running, Rolling Log File, User Log File Access, Unique Object ID After Server Restart, User Log File Token Removal
- PTZ added support: FLIR, GeoVision[Beta], Hanwha [Beta]
- Client: 3D Move Controls, World Coordinate Axis

Unique Features

QORTEX DTC 2.3 is a combined hardware/software solution that relies on the Quanergy LiDAR M-Series sensors, including all models of M8™ and MQ™-8.

This solution accomplishes its demanding goals through the following features:

- **Extended Detection Range in Real-Time.** M-Series sensors can continuously track static and moving objects accurately and in real time within a 40-meter radius for M8 sensors or within a 70-meter radius for MQ-8 sensors.
- **Classification of Shape Data.** The sensor produces surveillance data the host computer collects, records, visualizes, analyzes, classifies, and outputs to an Object List (which gives the user a basis for further action outside of QORTEX DTC, such as the notification of external alarm systems via LAN, TCP, and HTTP GET).
- **Persistence in Tracking.** The commercial-grade, static LiDAR visualization system senses and displays the movement of objects over time, persisting even through blockages and crowd gaps. This gives users the ability to track and record the historical movements of potential threats.
- **Resilience in Suboptimal Conditions.** The sensor is designed for use indoors and outdoors and in bright or dark conditions, with no infrared signature needed. It also withstands extreme weather, from bitter cold to baking sun, with complete ingress protection from mist, rain, snow, and dust.
- **Fault Tolerance, with Automatic Reconnect.** Designed for persistence, the QORTEX DTC server recovers persistently, since it runs as a daemon service. If a sensor disconnects from a server, the server sends a notification of the event, continues to run with the remaining connected sensors, and auto-reconnects with the sensor without reboot. The QORTEX DTC client also automatically resumes in Monitor mode after restart, with 3D visualization restored.
- **Configurable.** The sensor can be configured through the webserver to adjust the frame rate from 5 to 20 hertz, with single or multiple (up to 3) returns. We strongly recommend that you set the sensor to single return mode when using it with QORTEX DTC. The field of view is adjustable when less than a full 360° and is preferred.

  Configure sensors using the Sensor Settings Management application (webserver) built into the sensor. See *Managing the Sensor* in your sensor-specific user guide. Request the sensor user guide from support@quanergy.com.
- **Leverage Q-View Calibration.** Q-View™ is the Quanergy sensor discovery and management toolkit, which calibrates multiple sensors into the same space and outputs the resulting `transform_alignment.xml` file. QORTEX DTC takes the output file and aligns the overlapping vision of multiple sensors into an enriched Multi-LiDAR Fusion view.

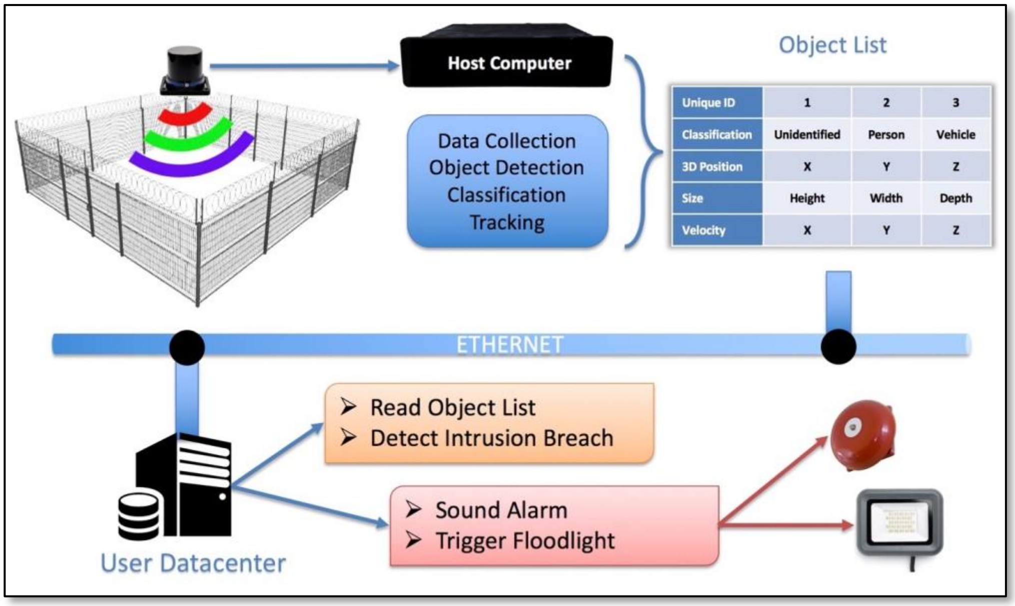
- **Enhances Legacy Systems.** QORTEX DTC reduces or eliminates false positive threats and dramatically boosts the surveillance effectiveness of the Video Management Systems (VMS) by measuring and providing exact 3D coordinates of people.

System Architecture

The QORTEX DTC 2.3 system architecture deploys in a simple distributed scenario that includes the QORTEX DTC server and a QORTEX DTC client or third-party application to consume the server output.

- The server software interfaces with statically installed LiDAR sensors and produces output to your own network infrastructure, including a real-time list of tracked objects accessed through the QORTEX API. See Appendix 1: *QORTEX API* (page 172). Configuration and control of the server software are achieved through the client interface and/or the gRPC API. See Appendix 2: *API for Remote Procedure Calls* (page 184). The gRPC API can be used to develop a customized client-user interface.
- The client software enables control and creation of a location (area of interest), with visualization of the corresponding server output point cloud, detected objects, tracks, classification, zones, and counter lines. Any number of potential third-party host infrastructure applications may subscribe to published data stream outputs for surveillance or visualization. See Appendix 1: *QORTEX API* (page 172).

The QORTEX DTC solution blocks of functionality are shown in *Figure 1. QORTEX DTC System Architecture Overview*. LiDAR sensors that integrate into an existing deployment might be able to leverage that existing power and communication infrastructure, if compatible.



Unique ID	1	2	3
Classification	Unidentified	Person	Vehicle
3D Position	X	Y	Z
Size	Height	Width	Depth
Velocity	X	Y	Z

Figure 1. QORTEX DTC System Architecture Overview

System Components

To use Quanergy QORTEX DTC for monitoring objects in a location requires:

- Physical site installation of the sensors. See the sensor User Guides.
- Computers meeting defined requirements for software installation of QORTEX DTC server, and QORTEX DTC client. See *Preparing the Computing Environment* (page 30).
- Software configuration:
  - **License Manager**. See *Managing the License* (page 45).
  - **Sensor Settings Management** application (web server). See the sensor user guides. Request the sensor user guide from support@quanergy.com.
  - **Q-View**. See the *Q-View User Guide*. Download from https://downloads.quanergy.com/.
  - **QORTEX DTC Server.** See *Download QORTEX DTC Server Software and Documents* (page 37).
  - **QORTEX DTC Client.** See *Install QORTEX DTC Client Software* (page 43).
- All the sensors, PTZ cameras, and QORTEX DTC residing on a single Ethernet network.

Supported Quanergy Sensors

- M8 family: M8, M8 PRIME, MQ-8, MQ-8 PoE+

Object and Capacity Specifications

Table 2. QORTEX DTC Server Specifications lists the scope and range specifications for QORTEX DTC 2.3 support.

Table 2. QORTEX DTC Server Specifications

Parameter	Specification
Object Information	Provides an object list with 3D direction and position, velocity, and classification in Protobuf, JSON, or XML formats
Classification Types	Human, Vehicle, Unknown Additional SubVehicle licensing adds: TWOWHEELER_VEHICLE, PASSENGER_VEHICLE, COMMERCIAL_VEHICLE
Continuous Tracking Range	M8 sensors: 80 meters (40-meter radius) MQ-8 sensors: 140 meters (70-meter radius)
Number of Simultaneous Objects	Depends on the QORTEX DTC server and sensors.
VMS Compatibility	Milestone Xprotect® and Genetec™ Security Center (RSA, PFA, and TTE modules)
Cameras	PTZ: no hard limit, up to 5 cameras tested

Process Setup Overview

To visualize and track objects in a space requires the following setup.

1. Complete installation of the sensors, QORTEX DTC server and QORTEX DTC client on a single Ethernet network. See *Preparing the Computing Environment* (page 30) and *Installing QORTEX DTC Server and Client* (page 37).
2. Verify licenses for PTZ camera and Rules.
   - To add PTZ cameras, check the licensing includes optional item: Number of **PTZ Cameras:** 1 or more.
   - To define Rules for PTZ Camera, Network Actions, or automatic Recording, check the license file includes optional item: **Add-Ons:** Rules

   Check the licensing by logging into the license server or viewing the licensing details through the activated QORTEX DTC server. See *Managing the License* (page 45).
3. Start the QORTEX DTC server and the QORTEX DTC client. Then connect the client to the server. See *Starting and Stopping QORTEX DTC* (page 56).
4. Understand the QORTEX DTC client elements. See *QORTEX DTC Client Interface* (page 74).
5. On first time login to Qortex DTC client, create `Admin` user.
6. Enter **Configuration** mode. See *Entering Configuration Mode* (page 84).

   Optionally, view and verify **TCP Publishing** file format and port settings for object lists, sensor health state, and zone/counter line lists, and optionally change **Add Data Size** or **Network Byte Order** settings.
7. Optionally, add users and enable **Security** mode. See *Security Mode* (page 64) and *User Management* (page 68).
8. **Add sensors** to the QORTEX DTC server. See *Adding Sensors* (page 87).

   The server can be configured to receive data from either a single sensor or multiple sensors for a single location. Sensors must be calibrated before they can be added to a server. Use Q-View to calibrate multiple sensors. Download the *Q-View User Guide* from https://downloads.quanergy.com/.
9. Optionally, add, calibrate, and configure **PTZ cameras** through the QORTEX DTC client. See *Adding PTZ Cameras* (page 125).

   A specific PTZ camera license is required to add PTZ cameras. Calibrate PTZ cameras to align with the sensors to track objects. Configure PTZ camera Field of View (FOV) to help determine the camera coverage area. When idle, PTZ cameras can be used by other applications. Define the PTZ camera Home Location, to which it returns when idle.

Objects assigned an ID from sensors and in a PTZ Camera FOV at a location, can be tracked automatically using Rules or manually if selected through the PTZ camera icon in the Visualizer.

10. Optionally, add **Inclusion zones** with data displaying in the **Visualizer**. See *Establishing Zones and Counter Lines* (page 105).

    If added, QORTEX DTC only monitors the area designated by the Inclusion zone, rather than the entire coverage area.

    In the Visualizer create a polygon shape cover the area to track. Adjust the Z Height minimum (min Z) and maximum (Max Z) for vertical scanning.

11. Add **Event zones** with data displaying in the **Visualizer**. *See Establishing Zones and Counter Lines* (page 105).

    In the Visualizer create a polygon shape cover the area to track. Adjust the Z Height minimum (min Z) and maximum (Max Z) for vertical scanning.

    The sensors and PTZ camera follow objects per the rules set for the Event zone.

12. Add **Exclusion zones**, with data displaying in the **Visualizer**. See *Establishing Zones and Counter Lines* (page 105).

    In the Visualizer create a polygon shape cover the area to exclude from tracking. Adjust the Z Height minimum (min Z) and maximum (Max Z) for vertical scanning.

    The sensors and PTZ camera *do not* follow in the Exclusion zones. Objects in the Exclusion zone are ignored.

13. View the data in the QORTEX DTC client Visualizer. See *Visualizing Data* (page 114).

14. Add **Rules** to track objects. See *Adding Rules* for Event Zones (page 142).

    If the Rules tab is grayed out, verify and update licenses. Licensing **Add-Ons** field must state **Rules**. See *Adding Rules* for Event Zones (page 142) to create a rule.

    Event Rules – Types include:

    - **PTZ Camera rule.** A PTZ camera rule causes the PTZ camera to follow a selected object. The PTZ camera time switching option enables the PTZ camera to switch between tracking of objects over time. These rules override Rule-based Zone priorities. See *Add PTZ Camera Automatic Tracking Rule* (page 148).
    - **Network Action rule.** A network action rule sends a message each time a selected object class (Any, Human, Vehicle, Human and Vehicle, or Unknown) enters or exits the selected area. See *Add Network Actions Rule* (page 143).
    - **Recording rule.** Automatically start recording the Visualizer raw sensor data per Rule settings. Stop recording per Rule settings, plus one (1) minute buffer. This step is for recording data when a rule is triggered. See *Add Recording Rule* (page 146).

15. Optionally, select **Recording** options. See *Recording Data* (page 153).

- **Automatic Recording.** In **Configuration** panel > **Rules** tab > **Record,** toggle to **ON** to enable event recording. Recordings are triggered only when QORTEX DTC client is in **LIVE** mode.
- **Manual Recording.** In **Configuration** panel > **Rules** tab > **Record,** toggle to **OFF**. Click **RECORD** button in top right Visualizer. Recording continues until you manually stop. Click the **RECORD** button again. While manually recording, the **Rules > Record** option remains disabled.
- **Playback a recording.** Toggle the **Sensor Connection** icon to **Disconnected**. Select **PLAYBACK** from the **LIVE/PLAYBACK** menu. Select a recording from the list of recordings. Click **Play** in the Player control panel.

Documentation

Table 3. Related Documents lists the latest versions of the relevant user documents and how to access them.

Table 3. Related Documents

Topic area	**Guide**	**Access**
QORTEX-specific guides	QORTEX *DTC 2.3 User Guide* QORTEX *DTC 2.3 Quick Start Card*	Download from https://downloads.quanergy.com/
Sensor management guides	*Q-View User Guide* *Q-View Quick Start Card*	Download from https://downloads.quanergy.com/
Sensor user guides for each LiDAR sensor model	*User Guide* *Quick Start Card*	Request from support@quanergy.com
Sensor datasheets	*Datasheets*	Download documents from https://quanergy.com/downloads/

2. Preparing the Computing Environment

This chapter described the environment, sensor, and hardware requirements for installing and using QORTEX DTC 2.3. Complete the steps described in the following sections before you install QORTEX DTC 2.3.

Environment Prerequisites

Every location where detection, tracking, and classification needs to occur requires a QORTEX DTC 2.3 system operating with the following specifications:

- **Sensors**.

  The number of sensors effectively used in the QORTEX DTC 2.3 system is affected by two variables:

  - The selection of a computer for hosting the QORTEX DTC 2.3 server. See *QORTEX DTC Server Host Computer* (page 32).
  - The scope of license purchased. See *Managing the License* (page 45).

- **Sensor Software.**

  Sensor Web Server. Quanergy Sensor Settings Management application is a web server provided for each sensor, that is accessed through a web browser using each sensor IP address. See your sensor specific user guide. Request the sensor user guide from support@quanergy.com.

  Q-View™. Quanergy sensor management and visualization software. Provides information about the sensor network and is used to calibrate and align the sensors. Download the *Q-View User Guide* from https://downloads.quanergy.com/.

- **Computers**.

  Host Computer. Each computer or virtual machine hosts one instance of the QORTEX DTC 2.3 server software and accommodates a certain number of LiDAR sensors, depending on the host/VM specifications and capabilities. The QORTEX DTC 2.3 client can be on the same computer as the QORTEX DTC server, but typically is on a separate computer.

  There are a variety of options for the computer(s) hosting the server and client, as discussed in the following sections. For computer parameters and requirements see, *QORTEX DTC Server Host Computer* (page 33) and *QORTEX DTC Client Host Computer* (page 34).

  Computer Accessories. A keyboard, mouse, and monitor are needed to use the client. Depending on how the server software is configured, these accessories may be required when first setting up the server. However, once the daemon service starts, the accessories will not be needed, since the server is designed to run headless.

Hard Drive. Recordings can quickly consume large amounts of memory. If recording, set up the system to prevent filling up the hard drive. If hard drive space becomes scarce, integrate an external hard drive, preferably one that easily exceeds the amount of expected data.

- **Operating System**.

  QORTEX DTC 2.3 server is installed on machine running Linux® distribution of Ubuntu® 20.04.

  QORTEX DTC 2.3 client is installed on typically, on a separate computer from the server, and is running either Microsoft Windows® 10 or Ubuntu 20.04 (LTS Focal Fossa).

- **Network**.

  All sensors must be on the same Ethernet subnet as the QORTEX DTC 2.3 server and client.

  IPv4 and IPv6 addresses are supported.

LiDAR Sensor Prerequisites

For sensors deployed in areas where power outages occur, a backup power system is strongly advised. However, the QORTEX DTC server recovers and the client automatically resumes in Monitor mode after restart, with 3D visualization restored.

1. Understand, set up, and mount the LiDAR sensors.

   a. Ensure the sensors are on the appropriate network.

      LiDAR sensors connect to the installation site local area network (LAN) via TCP/IP protocol. The QORTEX DTC server and QORTEX DTC client connect to the same LAN in order to access the visualization and alert system.

   b. Set the values for **Frame Rate in Hz**, **Return Data Select**, **Scan Field** using the Sensor Settings Management application (web server).

      Sensors run at a configurable frame rate (5-20 Hz) with a single or multiple (up to 3) returns, and their HFOV may be adjusted to less than 360° if preferred.

      See *Managing the Sensor* in your sensor-specific user guide. Request the sensor user guide from support@quanergy.com.

> **Note:** Quanergy recommends selecting only one return to save on network bandwidth. When a single return is selected, a LiDAR sensor generates 20 megabits per second. When all three returns are selected, a LiDAR sensor generates close to 60 megabits per second. The single return option has no performance reduction because the object point cloud published by QORTEX DTC shows only a single return regardless.

c. Configure object list output rate.

   Adjust the parameter through the `settings.xml` file or client. See Appendix 1: *QORTEX API* > *Outputs to Consume* **>** *Object List* **>** Frequency bullet (page 174).

2. Download and install Quanergy Q-View application.

   Download the Q-View executable to the planned QORTEX DTC 2.3 client host computer. Download both the Q-View executable and the *Q-View User Guide* from https://downloads.quanergy.com/. Choose the appropriate OS version of Q-View (Linux or Windows) for your system.

3. For a single LiDAR in the location, use the Q-View application Dashboard tab to discover and note the sensor IP address.

4. For multiple LiDARs in the location, combine the point clouds.

   To set up a location from scratch, use Q-View Dashboard tab to connect to the sensors and the Calibrate tab to align their point clouds. The QORTEX DTC 2.3 client uses the calibration file (`transform_alignment.xml`) output by Q-View.

   The following is a summary of the calibration task completed in Q-View. See the *Q-View User Guide*.

   a. In Q-View, select sensor to connect and calibrate.

   b. Click the (+). Initially, the **Connected** field shows 0. As Q-View completes connection, the **Connected** field shows 1 and the **Sensor** block turns **Green**.

   c. Click Q-View Calibration icon. Open folder, browse, new folder, select folder, click **Next**.

   d. Select sensor, click **Next**. Select the next sensor, click **Next**.

   e. Align sensor using -/+ TRANSLATE and ROTATION panels. For example, moving **Green** to align over **White**. Click **Refine** for minor alignment. Click **Accept**.

   f. Repeat for each sensor. Click **Save** (floppy) icon.

      When completed, the `transform_alignment.xml` file is created.

QORTEX DTC Server Host Computer

The QORTEX DTC server machine can be an off-the-shelf computer can be set up and installed for use with the sensors.

- PC requires certified Linux Ubuntu 20.04 operating system. A list of desktops and laptops is at:

  https://certification.ubuntu.com/

- A Linux virtual machine (VM).

VMware ESXi 6.5 is the recommended VM. Other VM options are listed at:

https://www.digitaltrends.com/computing/best-virtual-machine-apps-for-mac-linux-and-windows-pcs/

- For Docker container, install the Docker engine on the Linux host machine.

  https://docs.docker.com/engine/install/ubuntu/

The recommended maximum number of sensors the hardware can support. This is not a hard limit. The actual number depends on the number of objects being tracked and the overlap between sensors. Depends on the QORTEX DTC server and sensors. For example:

- Small server, 1-4 sensors
- Medium server, 6-10 sensors
- Larger server, 12-18 sensors

An Ubuntu 20.04 certified PC desktop or laptop can host the QORTEX DTC 2.3 server. *Table 4. Minimum Specifications for PC Desktop or Laptop Hosting Server* lists specification recommendations for running a single instance of QORTEX DTC server on the PC.

Table 4. Minimum Specifications for PC Desktop or Laptop Hosting Server

System	Parameter	Specification
Hardware	Central Processing Unit (CPU)	Intel Core i3, Gigabit Ethernet (2 virtual cores + 1 virtual core per sensor) Must support AVX2.
	Random Access Memory (RAM)	4 GB
	Storage Memory (no recording)	100 MB (QORTEX DTC needs < 1 GB drive space for installation.)
	Recording Storage Requirement	150 MB/sensor/min (single return mode)
Ethernet	Network Ports	Local Area Network (LAN) / Wide Area Network (WAN)
Software	Operating System	Ubuntu 20.04 LTS

Migrate from QORTEX DTC Server 1.x, 2.x to QORTEX DTC 2.3

If you are migrating the location from QORTEX DTC server version 1.x, 2.0, or 2.1 to QORTEX DTC version 2.3, copy, then import the current point cloud calibration with any Event and Exclusion zones.

1. Copy required files from the QORTEX DTC server 1.x location to the QORTEX DTC 2.3 client. Copy the `settings.ini` for live configuration. Copy `dataset` folder for playback content. Copy files to: `/home/quanergy/quanergy/qortex`

2. Import the `settings.ini` or `transform_alignment.xml` files into the client. See *Add or Update Multiple Sensors at* a Location (page 89).

   For QORTEX DTC server version 2.0 to 2.3, when the QORTEX DTC server starts, the `settings.xml` file is automatically uploaded and converted to QORTEX DTC version 2.3 format.

   If QORTEX DTC is installed on a different machine, copy the `settings.xml` file from the old machine to the new machine.

3. Reactivate the license. See *Activate a License* (page 47).

QORTEX DTC Client Host Computer

An off-the-shelf computing solution, such as a PC or laptop, can host the QORTEX DTC client software.

For best results, make sure the client host computer complies with the minimum PC specifications. See *Table 5. Minimum Specifications for PC Desktop or Laptop Hosting Client*.

Table 5. Minimum Specifications for PC Desktop or Laptop Hosting Client

System	Parameter	Specification
Hardware	Central Processing Unit (CPU)	Intel Core i3, Gigabit Ethernet Must support AVX2.
	Random Access Memory (RAM)	4 GB
	Installation Drive Space	> 1 GB for the client Software
Ethernet	Network Ports	Local Area Network (LAN) / Wide Area Network (WAN)
Software	Operating System	Windows 10 or Ubuntu 20.04 LTS
Graphics	Graphics Card	OpenGL-compliant card such as GeForce 8800 or Radeon 4770
Display	Resolution Support	1024x768, 1280x800, 1280x1024, 1366x864, 1440x900, 1600x900, 1920x1080, 2048x1536, 2560x1440, 3840x2160, 4096x2160

Connect and Start the Host Computers

Before launching the software, all hardware components must be in place. Connect the computers hosting the QORTEX DTC server and the QORTEX DTC client to power, accessories, sensors, and the network, as follows:

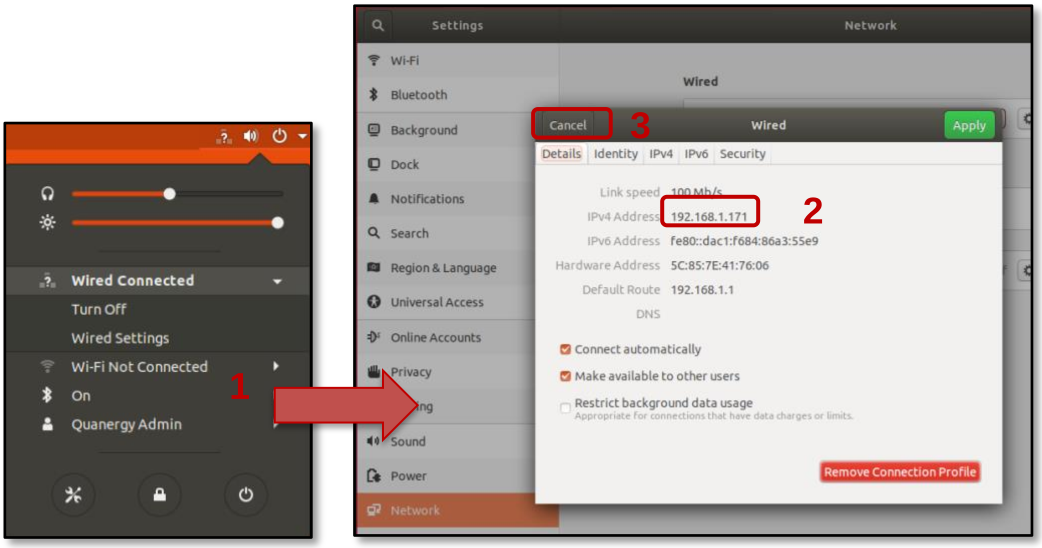
1. Connect the host computer to a power source.
2. If it doesn’t turn on automatically, turn on the power button for the host computer.

3. If it doesn't come up automatically, boot up the Ubuntu operating system.
4. Connect user-supplied accessories (keyboard, mouse, monitor) to USB/HDMI/VGA inputs, or log in remotely.
5. Connect the host computer to the same network the sensors are on, via a cable plugged into an Ethernet port that is not a DHCP server.
6. Log or write down the IP address of the host computer for later use.

Locate IP Address of Linux Host

For the Linux computer hosting the server. See *Figure 2. Discover Linux Host Computer IP Address*.

1. On the menu bar of the Ubuntu desktop, click the **Network Manager** ⇅ menu, and select **Wired Connection**.
2. Write down the host computer **IP address** as listed in the pop-up **Wired** window here: ______________. QORTEX DTC supports both IPv4 and IPv6 IP addresses.
3. Click **Cancel**, as you are not making any changes.



Link speed 100 Mb/s
IPv4 Address 192.168.1.171
IPv6 Address fe80::dac1:f684:86a3:55e9
Hardware Address 5C:85:7E:41:76:06
Default Route 192.168.1.1
DNS

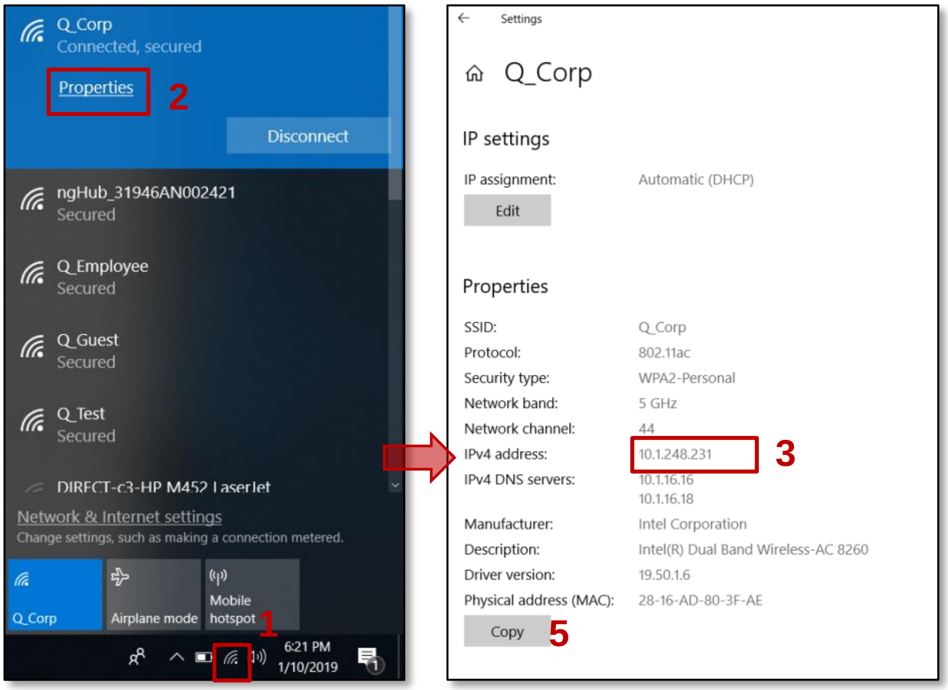
☑ Connect automatically
☑ Make available to other users
☐ Restrict background data usage
Appropriate for connections that have data charges or limits.

Figure 2. Discover Linux Host Computer IP Address

Locate IP Address for Windows Host

For the Windows computer hosting the client. See *Figure 3. Discover Windows Host Computer IP Address*.

1. Click the **Internet Access** icon 📶 in the bottom right task bar.
2. Select the **Properties** link from the pop-up.
3. In the window that appears, scroll down to **IP address**.
4. Write down the host computer IP address here: _______________. QORTEX DTC supports both IPv4 and IPv6 IP addresses.
5. Click **Copy**, then exit the **Settings** page click the return ← arrow. Close the network dialog.



IP assignment: Automatic (DHCP)

SSID: Q_Corp
Protocol: 802.11ac
Security type: WPA2-Personal
Network band: 5 GHz
Network channel: 44
IPv4 address: 10.1.248.231
IPv4 DNS servers: 10.1.16.16
10.1.16.18
Manufacturer: Intel Corporation
Description: Intel(R) Dual Band Wireless-AC 8260
Driver version: 19.50.1.6
Physical address (MAC): 28-16-AD-80-3F-AE

Figure 3. Discover Windows Host Computer IP Address

3. Installing QORTEX DTC Server and Client

After implementing the hardware and connecting it to the same Ethernet subnet, remove the old software (if necessary), download the latest software, and install the downloaded software through the Quanergy Download Center.

For user-supplied host computers, virtual machines, or Docker containers, acquire the QORTEX DTC server and client applications, as well as relevant documents, and follow the procedures in the following subsections.

Download QORTEX DTC Server Software and Documents

Download the QORTEX DTC server software and documents.

1. Login to a machine to download the QORTEX DTC installer. This is typically the QORTEX DTC host machine.
2. In an Internet browser, navigate to the Quanergy Download Center (https://downloads.quanergy.com/), and click the QORTEX DTC icon at the top of the page, or scroll to the QORTEX DTC section. See *Figure 4. Quanergy Qortex Server and Client Download Path*.
3. Download the documentation.

   In the Quanergy Download Center, click links to the QORTEX DTC 2.3 user. For example, download the *QORTEX DTC 2.3 User Guide* and *QORTEX DTC 2.3 Quick Start Card*.
4. Enable the download links.
   a. Click the **End User Software License Terms** link, read it, return to the Quanergy Download Center.
   b. Check the **To download software, please agree to the End User Software License Term** box.
5. Open the QORTEX DTC server `qortex_readme.txt` file.

   From the QORTEX DTC 2.3 section of the Quanergy Download Center, click the **Qortex DTC 2.3 Server** link. This opens the `qortex_readme.txt` installation instructions.
6. Locate the QORTEX DTC server installer download page.
   a. From the `qortex_readme.txt`, scroll to the **SOFTWARE INSTALLATION** section > **Download the Qortex Server** line.
   b. Select the QORTEX DTC server link, then copy/paste the address in a browser to view the QORTEX DTC server download options.

      https://downloads.quanergy.com/qortex/server

This QORTEX DTC `/server` download page lists the current QORTEX DTC 2.3 server software installation packages.

7. From the QORTEX DTC server download page:

   a. Select the QORTEX 2.3 Server installation bundle specific to your host machine.

      - Linux host (physical or virtual): `qortex-server_2.3.xxx_amd64.deb`
      - Docker container: `qortex-server_2.3.xxx.tar.gz` and `settings.xml`

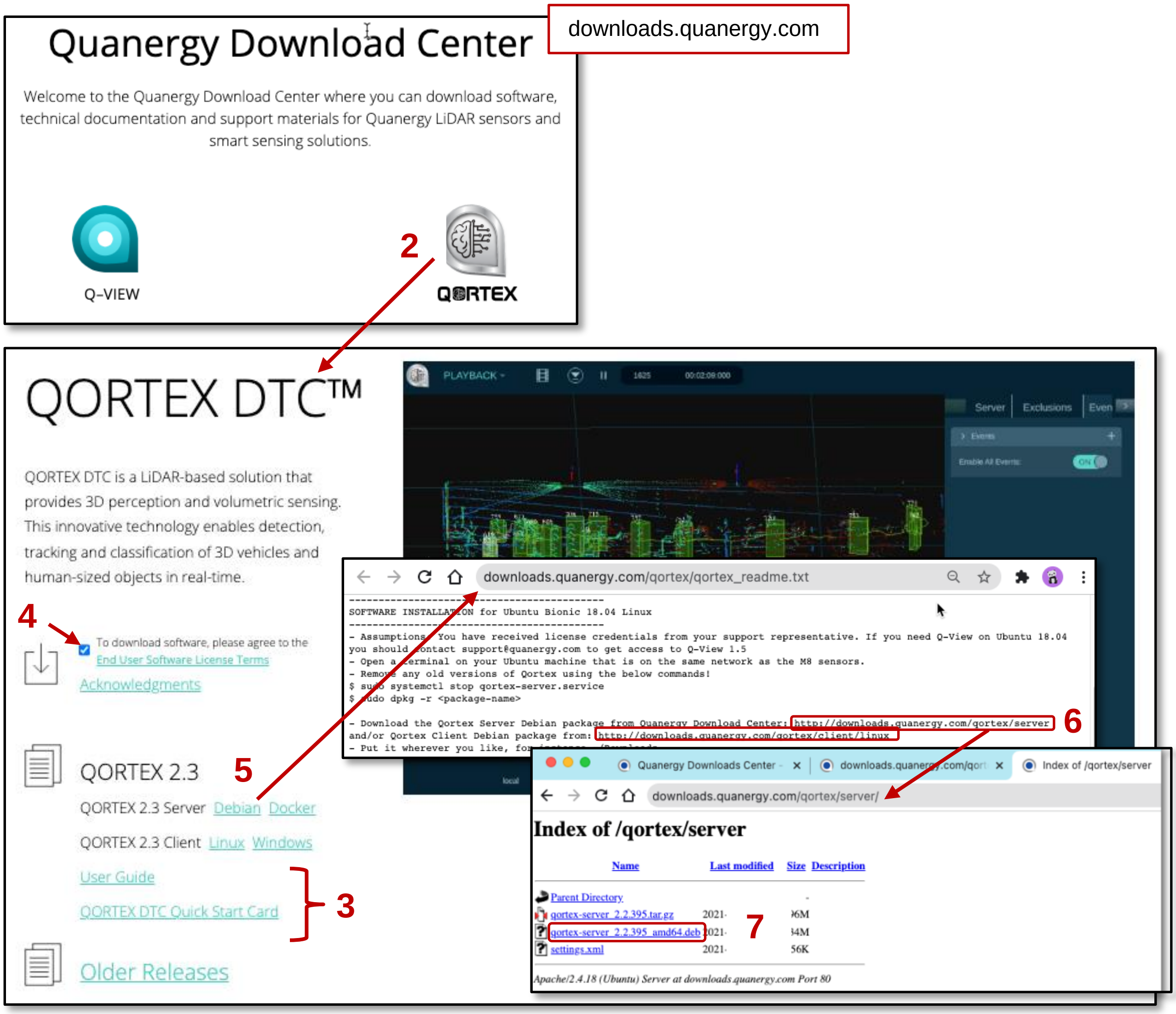
   b. Download the installation bundle to the Downloads folder or another destination.

8. Continue to the next section:

   If you downloaded the QORTEX DTC installer on the QORTEX DTC host machine, skip the next section and proceed to:

   - For Linux: *Install or Update QORTEX DTC with a Debian Package* (page 40)
   - For Docker: *Install QORTEX DTC in a Docker Container* (page 41).

   If you need to move the QORTEX DTC installer to the QORTEX DTC host machine, proceed to *Move QORTEX DTC Installer Package to the Host Computer* (page 39).



Quanergy Download Center

Welcome to the Quanergy Download Center where you can download software, technical documentation and support materials for Quanergy LiDAR sensors and smart sensing solutions.

QORTEX DTC™

QORTEX DTC is a LiDAR-based solution that provides 3D perception and volumetric sensing. This innovative technology enables detection, tracking and classification of 3D vehicles and human-sized objects in real-time.

☑ To download software, please agree to the End User Software License Terms

Acknowledgments

QORTEX 2.3

QORTEX 2.3 Server Debian Docker

QORTEX 2.3 Client Linux Windows

User Guide

QORTEX DTC Quick Start Card

Older Releases

downloads.quanergy.com/qortex/qortex_readme.txt

```
-------------------------------------------
SOFTWARE INSTALLATION for Ubuntu Bionic 18.04 Linux
-------------------------------------------
- Assumptions: You have received license credentials from your support representative. If you need Q-View on Ubuntu 18.04
you should contact support@quanergy.com to get access to Q-View 1.5
- Open a terminal on your Ubuntu machine that is on the same network as the M8 sensors.
- Remove any old versions of Qortex using the below commands!
$ sudo systemctl stop qortex-server.service
$ sudo dpkg -r <package-name>

- Download the Qortex Server Debian package from Quanergy Download Center: http://downloads.quanergy.com/qortex/server
and/or Qortex Client Debian package from: http://downloads.quanergy.com/qortex/client/linux
- Put it wherever you like, for instance ~/Downloads
```

Quanergy Downloads Center - | downloads.quanergy.com/qort | Index of /qortex/server

downloads.quanergy.com/qortex/server/

Index of /qortex/server

Name	Last modified	Size	Description
Parent Directory		-	
qortex-server_2.2.395.tar.gz	2021-	96M	
qortex-server_2.2.395_amd64.deb	2021-	34M	
settings.xml	2021-	56K	

Apache/2.4.18 (Ubuntu) Server at downloads.quanergy.com Port 80

Figure 4. Quanergy Qortex Server and Client Download Path

Move QORTEX DTC Installer Package to the Host Computer

If the QORTEX DTC server installation package is used on a different computer (such as an air-gapped computer) than the one it was downloaded to, do the following:

1. Search the download computer for the (Linux or Docker) package that needs to be moved (where 2.3.*xxx* is the current release version) and note the filepath.

   Linux file:

   ```
   qortex-server_2.3.xxx_amd64.deb
   ```

   Docker files:

   ```
   qortex-server_2.3.xxx.tar.gz
   settings.xml
   ```

2. Transfer the files to an external USB or hard drive.
3. Connect the drive to the destination computer and transfer the relevant files to a filepath comparable to where the Debian had been placed during the original download.
4. Continue to the next section, *Install or Update QORTEX DTC with a Debian Package* (page 40) or *Install QORTEX DTC in a Docker Container* (page 41).

Install or Update QORTEX DTC with a Debian Package

When the QORTEX DTC server installation package is in its final destination, install it as follows.

> **Note:** If you are installing your QORTEX DTC software on a machine that is different than the machine you used to download the software, first complete the steps in *Move QORTEX DTC Installer Package to the Host Computer* (page 39).

1. Open a new terminal window (Ctrl+Alt+T) on the host computer.
2. Remove any old versions of QORTEX DTC.

```
    $ sudo systemctl stop qortex-server.service
    $ sudo dpkg -r <qortex-server_old_version>
```

3. Install QORTEX DTC server.

   - Execute the following command to install the QORTEX DTC server from the downloaded Debian package, where 2.3.*xxx* is the current release version.

```
        $ cd ~/Downloads                    ###or the filepath of the download
```

   - Execute the following command to install the required package only:

```
        $ sudo dpkg -i qortex-server_2.3.xxx_amd64.deb
```

   All programs and libraries are placed where they are needed. To use them, the license must be activated. See *Managing the License* (page 45).

   If this is an update to existing software, reload the software. See *Restart the Server Daemon Service* (page 57) and *Figure 5. Post-Installation Commands for Refreshed Server*.

   If this is a fresh installation, proceed to Step 5.

   a. From a terminal window (Ctrl+Alt+T) on the host computer, execute the command:

```
    $ sudo systemctl daemon-reload
```

   b. In the **Authenticate** pop-up window, enter the password for the server host computer.

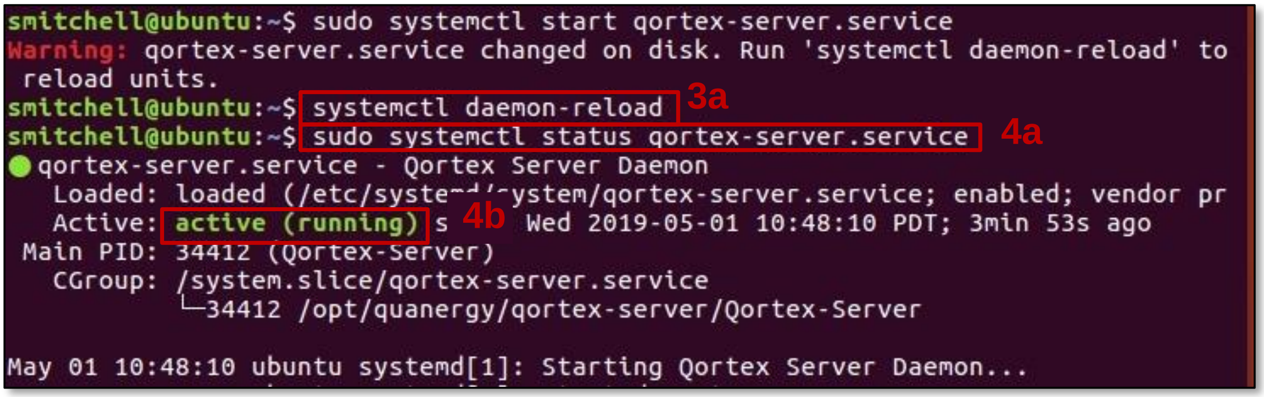
4. Optionally, verify the service is running.

a. Run the command.

```
$ sudo systemctl status qortex-server.service
```

b. Check the status **Active** field states: `active (running)`.

This completes QORTEX DTC server installation using a Debian package.



```
smitchell@ubuntu:~$ sudo systemctl start qortex-server.service
Warning: qortex-server.service changed on disk. Run 'systemctl daemon-reload' to
 reload units.
smitchell@ubuntu:~$ systemctl daemon-reload
smitchell@ubuntu:~$ sudo systemctl status qortex-server.service
● qortex-server.service - Qortex Server Daemon
   Loaded: loaded (/etc/systemd/system/qortex-server.service; enabled; vendor pr
   Active: active (running) s     Wed 2019-05-01 10:48:10 PDT; 3min 53s ago
 Main PID: 34412 (Qortex-Server)
   CGroup: /system.slice/qortex-server.service
           └─34412 /opt/quanergy/qortex-server/Qortex-Server

May 01 10:48:10 ubuntu systemd[1]: Starting Qortex Server Daemon...
```

Figure 5. Post-Installation Commands for Refreshed Server

Install QORTEX DTC in a Docker Container

When the QORTEX DTC server Docker installation package is in its final destination, install it as follows:

1. Load the docker image on the host, from a terminal on the host, run:

```
$ docker load < qortex-server_2.3.xxx.tar.gz
```

2. Verify the docker image is deployed on the host, run:

```
$ docker images
```

   Scan the result for `qortex-server` with tag `2.3.xxx`, part of the docker images command.

3. Add `quanergy` user and set permissions for the user. On the host machine, run:

```
$ sudo adduser --system quanergy --group
$ sudo mkdir -p /home/quanergy/quanergy/qortex/location
$ sudo chown -R quanergy:quanergy /home/quanergy
```

4. Copy the `settings.xml` file to the directory, `/home/quanergy/quanergy/qortex/location`

```
$ cp settings.xml /home/quanergy/quanergy/qortex/location/
```

5. Open the `settings.xml` file, update the sensor IP address, and save the file.

   QORTEX DTC supports both IPv4 and IPv6 IP addresses.

6. Activate the QORTEX DTC Server license.

```
$ docker run --rm --net=host -v
  /home/quanergy/quanergy/qortex:/home/quanergy/quanergy/qortex
  -it qortex-server:2.3.xxx
  --license activate <LICENSE_ID> <PASSWORD>
```

7. Run the container from the docker image. Provide the complete `/qortex` folder path on the host.

```
$ docker run --restart=always --net=host -v
  /home/quanergy/quanergy/qortex:/home/quanergy/quanergy/qortex
  -dit --name qortex-server qortex-server:2.3.xxx
```

8. Verify docker container is running.

```
$ docker ps -a
```

9. Verify the object data output.

```
$ nc 127.0.0.1 17171
```

   If you do not see the listed output then update `settings.xml`:

   a. Open for editing, the `settings.xml` file in the folder, `/home/quanergy/quanergy/qortex/location`.

   b. Update the `<Publisher>` section.

```
<Publisher>
    <Name>QORTEX1_OBJECT_LIST</Name>
    <Format>json</Format>
    <Port>17171</Port>
    <AddDataSize>true</AddDataSize>
    <NetworkByteOrder>false</NetworkByteOrder>
</Publisher>
```

10. Optional commands:

   - To see the logs:

```
$ docker logs qortex-server
```

   - To stop and remove the container:

```
$ docker rm -f qortex-server
```

   - To deactivate the license:

```
$ docker run --rm --net=host -v
    /home/quanergy/quanergy/qortex:/home/quanergy/quanergy/qortex
    -it --name qortex-server qortex-server:2.3.xxx --license deactivate
```

This completes QORTEX DTC server installation using Docker.

Install QORTEX DTC Client Software

The QORTEX DTC client software can be installed on a Windows or Linux operating system.

Install on Windows

From the QORTEX DTC client Windows machine, install the QORTEX DTC client software, as follows:

1. Download the QORTEX DTC 2.3 Windows client executable to the **Downloads** folder or another destination on the Windows QORTEX DTC 2.3 client host. See *Figure 4. Quanergy Qortex Server and Client Download Path*.
    a. Open an Internet browser and navigate to https://downloads.quanergy.com/.

    b. Click the QORTEX 2.3 Windows client.

    c. From the https://downloads.quanergy.com/qortex/client/windows/ page, select the QORTEX 2.3 Windows client executable.

    ```
    qortex-client_2.3.xxx_win64.exe
    ```

    d. Download the QORTEX 2.3 Windows client to the **Downloads** folder or another destination.
2. From your Windows machine, click the **QORTEX DTC 2.3 client** executable.
3. Before the Wizard opens, a dialog box appears at the bottom of the window. Select the **Save** button to download the executable to the computer **Downloads** folder.
4. Open the **Downloads** folder, then double-click the **QORTEX 2.3 DTC client** file.
5. If a pop-up window interrupts the Wizard to ask, **“Do you want to allow this app to make changes...?”** click the **Yes** button to continue.
6. If a pop-up window suggests uninstalling a previous version of the software
    a. Click **Yes**.

    b. In the **Wizard Uninstall** window, click the **Uninstall** button to confirm.

    c. In the **Wizard Uninstallation Complete** window, click the **Close** button to confirm.
7. For the Wizard setup sequence, agree to the terms of each step, then click the **Finish** button to dismiss the Wizard.

    A shortcut QORTEX DTC client icon is placed on the desktop as a quick way to start the client.

Install on Ubuntu

From the QORTEX DTC client Ubuntu machine, install the QORTEX DTC client software, as follows:

> **Note:** Whenever you're prompted, type the password for your host computer.

1. Download the QORTEX DTC Linux client installation package to the **Downloads** folder or another destination on the Ubuntu QORTEX DTC 2.3 client host. See *Figure 4. Quanergy Qortex Server and Client Download Path*.

   a. Open an Internet browser and navigate to https://downloads.quanergy.com/.

   b. Click the QORTEX 2.3 Linux client.

   c. From the https://downloads.quanergy.com/qortex/client/linux/ page, select the QORTEX 2.3 Ubuntu client executable.

   ```
   qortex-client_2.3.xxx_amd64.deb.
   ```

   d. Download the QORTEX 2.3 Linux client to the **Downloads** folder or another destination.

2. Open a terminal window on your client host computer and change directory to your download location. For example:

   ```
   $ cd ~/Downloads
   ```

3. Execute the command:

```
    $ sudo dpkg -i qortex-client_2.3.xxx_amd64.deb
```

4. The QORTEX DTC client installation starts, follow the prompts.

   When prompts finish, you have completed installing the QORTEX DTC client.

4. Managing the License

The QORTEX DTC 2.3 server (not the client) requires a valid license tied to both its host machine and a version of the server software before the application is permitted to run.

License Parameters

Type. The Premium license enables the software to detect, track, and classify humans and vehicles.

Sensors. Each license is approved for use with a specified number of sensors. This number is based on your purchase history.

Duration. There are two durations of licenses:

- **Annual licenses.** These expire based on the issue date.
- **Perpetual licenses.** These never expire for a particular version of the software.

Tasks. The server includes a License Manager (command-line interface utility) which manages the licenses for the following tasks:

- *Activate a License* (page 47)
- *Refresh a License* (page 49)
- *Deactivate a License* (page 50)
- *Renew a License* (page 51)
- *Check a License* (page 51)
- *Check a License Version* (page 53)

Credentials. Before activating a license, an email from a support representative containing a License ID and Activation Password is sent to you. If you do not receive the email, contact a support at support@quanergy.com.

Offline. The simplest way to complete the licensing tasks is through the Internet. However, if Internet access is not available on the QORTEX DTC server air-gapped host machine, an Internet-connected third machine can be used to retrieve the file that manages the license on the air-gapped host machine. See *Manage the License of an Air-Gapped Computer* (page 53).

License Portal

After the software is downloaded and installed, acquire and activate the QORTEX DTC server license. The License Portal provides an organized place to access software, retrieve license keys, manage licenses, update contact information, and find support contact information.

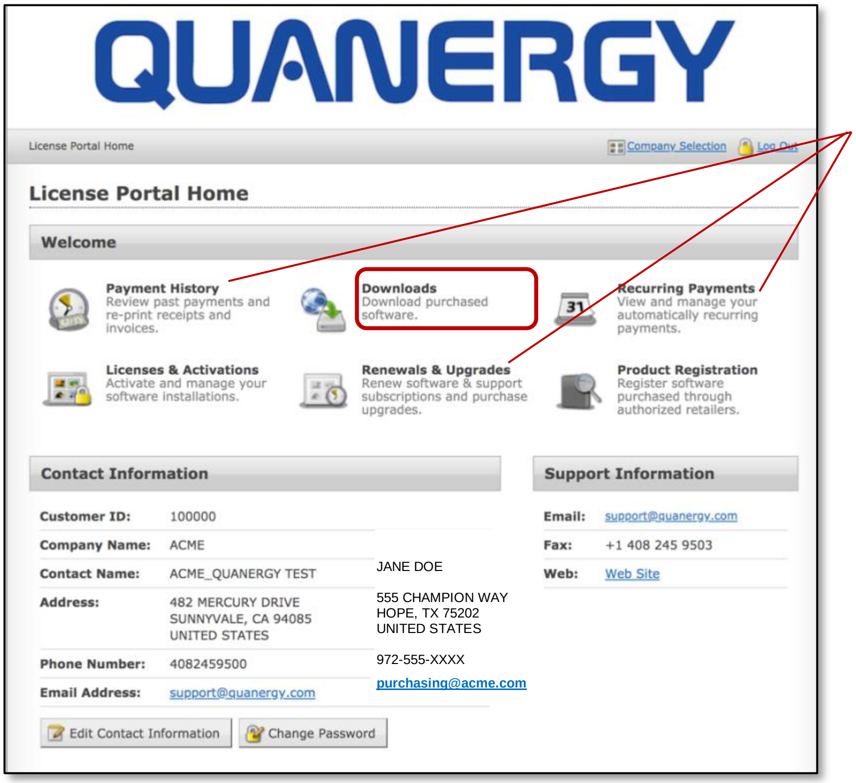
Using the License Portal

A support representative creates a user account and emails the following items to the user:

- Link and credentials necessary for logging in to the License Portal.
- License ID and Activation Password necessary for using the License Manager. See *License Parameters* (page 45).

Use the emailed information to log in to the License Portal, as follows:

1. In a browser, enter the link supplied in the email to navigate to the **License Portal**. https://license.quanergyworks.com/solo/customers/Default.aspx
2. Enter the **License Portal** credentials provided in the email.
3. Click the **Downloads** button to download the License Manager and check on the list of licenses. See *Figure 6. Customer License Portal*.



QUANERGY

License Portal Home   Company Selection   Log Out

License Portal Home

Welcome

Payment History Review past payments and re-print receipts and invoices.

Downloads Download purchased software.

Recurring Payments View and manage your automatically recurring payments.

Licenses & Activations Activate and manage your software installations.

Renewals & Upgrades Renew software & support subscriptions and purchase upgrades.

Product Registration Register software purchased through authorized retailers.

Contact Information

Customer ID:	100000	
Company Name:	ACME	
Contact Name:	ACME_QUANERGY TEST	JANE DOE
Address:	482 MERCURY DRIVE SUNNYVALE, CA 94085 UNITED STATES	555 CHAMPION WAY HOPE, TX 75202 UNITED STATES
Phone Number:	4082459500	972-555-XXXX
Email Address:	support@quanergy.com	purchasing@acme.com

Edit Contact Information   Change Password

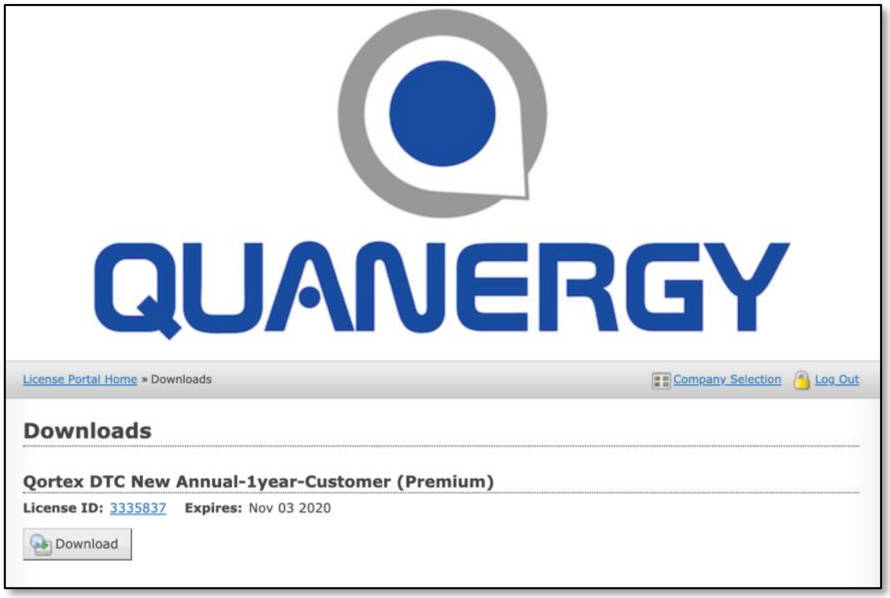
Support Information

Email:	support@quanergy.com
Fax:	+1 408 245 9503
Web:	Web Site

Payment History, Renewals & Upgrades, Recurring Payments are not used.

Figure 6. Customer License Portal

4. On the License Portal Downloads page, click the **Download** button for access to QORTEX DTC 2.3 software and documentation. See *Figure 7. Customer License Portal Downloads Page*.



License Portal Home » Downloads

Company Selection Log Out

Downloads

Qortex DTC New Annual-1year-Customer (Premium)

License ID: 3335837 Expires: Nov 03 2020

Figure 7. Customer License Portal Downloads Page

Activate a License

To activate the license, you need the credentials emailed from the support representative:

1. On the QORTEX DTC server host computer, open a terminal (or `ssh`) and log in.
2. Enter the proper directory:

```
    $ cd /opt/quanergy/qortex-server
```

3. Execute the following command:

```
    $ ./Qortex-Server --license activate
```

4. Respond to each prompt, and press **Enter**, as follows:

```
------------------ licensing ------------------
Do you want to process your request over the internet? (y/n) (Default: y) [type y]
Please enter your License ID:
        [type 8-digit license number, e.g., 12345678]
Please enter your Activation Password:
        [type 8-digit alphanum combination, e.g., A1B2C3D4]
Please enter your installation name (optional):
        [blank, or type site name, e.g., MyLab-2]
```

5. After pressing **Enter** for the last question, this message returns to confirm that the license is activated and provides a summary of options purchased and configured for your license.

```
qortex license has been successfully activated.
Status: Activated, registered to:    [usernames configured for license]
```

```
License id:                               [license number entered]
License type:                     [purchased license option]
Add-ons: Rules SubVehicle         [If None is listed, then both Rules or SubVehicle
                                          features are not supported]
Number of licensed sensors:       [number of sensor connections to your QORTEX
                                          DTC server allowed.]
Number of licensed cameras:       [If value is 0, PTZ cameras are not supported.
                                          Value of 1 or more is number of PTZ camera
                                          connections to your QORTEX DTC server
                                          allowed.]
Expiration Date:
```

Required items: Status, License id, License, type, and Number of licensed sensors.

Optional items are:

- **Add-ons**:
  - If **Rules** are not listed, then Rules configuration is not supported and the Rules tab is grayed out in the Configuration panel.
  - If **SubVehicle** is not listed, vehicle object subcategories are not identified or labeled. Without this license all vehicle types are labeled simply as `vehicle`.
- **Numbers of licensed PTZ cameras**: If the number is zero, then PTZ cameras are not supported and the PTZ tab is grayed out in the Configuration panel.

6. Start the server daemon service. See *Starting and Stopping QORTEX DTC* (page 56) and *Figure 8. License Activate from the Ubuntu Terminal*.

```
user:/opt/quanergy/qortex-server$ ./Qortex-Server –license activate
----------------- licensing --------------
Do you want to process your request over the internet? (y/n) (Default y) Y
Please enter your License ID: #########
Please enter your Activation Password: ########
Please enter your installation name (optional):

Qortex license has been successfully activated.
Status: Activated, registered to:
USER-NAME
License id: ######
License type: Quanergy premium
Number of license sensors: 15
Expiration Date: 2020-05-20T00:00:00Z
```

Figure 8. License Activate from the Ubuntu Terminal

Refresh a License

As long as the QORTEX DTC server is connected to the Internet, the software refreshes the license every day automatically. However, if any of the following events occur:

- An annual license was renewed.
- The sensor allowance was changed.

The license must be manually refreshed as follows. See *Figure 9. License Refresh from the Ubuntu Terminal*.

1. On the QORTEX DTC server host computer, open a terminal (or `ssh`) and log in.

   Enter the username and password for the host machine, if required.

2. Enter the proper directory:

```
    $ cd /opt/quanergy/qortex-server
```

3. Execute the refresh command

```
    $ ./Qortex-server --license refresh
```

4. Respond to the prompt, and press the Enter key, as follows:

```
    Do you want to process your request over the internet? (y/n) [type y]
```

5. This message returns to confirm the refresh:

```
    qortex license has been successfully refreshed.
```

```
user:/opt/quanergy/qortex-server$ ./Qortex-Server --license refresh
----------------- licensing --------------
Status: Activated, registered to:
USER-NAME
License id: ######
License type: Quanergy premium
Number of license sensors: 15
Expiration Date: 2020-05-20T00:00:00Z
Do you want to process your request over the internet? (y/n) (Default y)

Qortex license has been successfully refreshed.
Status: Activated, registered to:
USER-NAME
License id: ######
License type: Quanergy premium
Number of license sensors: 15
Expiration Date: 2020-05-20T00:00:00Z
```

Figure 9. License Refresh from the Ubuntu Terminal

Deactivate a License

Deactivating a license permits the transfer of the license from one QORTEX DTC server host machine to another.

> **Note:** Even while deactivated, an annual license continues its expiration countdown without pausing.

Deactivate the license as follows:

1. On the QORTEX DTC server host computer, open a terminal (or `ssh`) and log in.

   Ener the username and password for the host machine, if required.

2. Enter the proper directory:

```
    $ cd /opt/quanergy/qortex-server
```

3. Execute the deactivate command. See *Figure 10. License Deactivate from the Ubuntu Terminal*.

```
    $ ./Qortex-server --license deactivate
```

4. Respond to the prompt, and press **Enter**, as follows:

```
Do you want to process your request over the internet? (y/n)  [type y]
```

   After pressing the **Enter** key, the License Manager deactivates the license.

   This message returns to confirm that the process was successful:

```
qortex license has been successfully deactivated.
```

5. When ready to reactivate the license, see *Activate a License* (page 47).

```
user:/opt/quanergy/qortex-server$ ./Qortex-Server --license deactivate
----------------- licensing --------------
Status: Activated, registered to:
USER-NAME
License id: ######
License type: Quanergy premium
Number of license sensors: 15
Expiration Date: 2020-05-20T00:00:00Z
Do you want to process your request over the internet? (y/n) (Default: y)

Qortex license has been successfully deactivated.
```

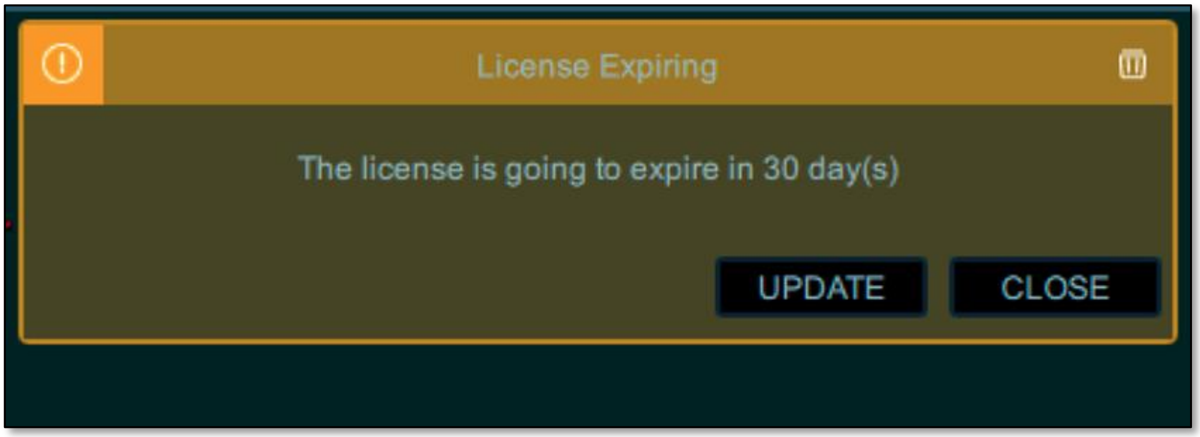
Figure 10. License Deactivate from the Ubuntu Terminal

Renew a License

You are responsible for renewing the annual license in a timely manner. If the license expires, then the QORTEX DTC server shuts down and the QORTEX DTC client stops functioning.

QORTEX DTC displays warning notifications concerning an expiring license in the **Status** panel. See *Figure 11. License Expiration Notification from the Status Panel*. Regarding the **Status** panel, see *Status Button* (page 82).

If the license expires, the QORTEX DTC application immediately terminates, logs a message in the server log, and sends a message to the client. To renew the license, contact your sales representative or support@quanergy.com and supply a new Purchase Order. Then activate the license. See *Activate a License* (page 47) or *Manage the License of an Air-Gapped Computer* (page 53).



License Expiring

The license is going to expire in 30 day(s)

Figure 11. License Expiration Notification from the Status Panel

Check a License

1. On the QORTEX DTC server host computer, open a terminal (or `ssh`) and log in.

   Enter the username and password for the host machine, if required.

2. Enter the proper directory:

```
    $ cd /opt/quanergy/qortex-server
```

3. Execute the license info command to see a report of the license details. See *Figure 12. License Info Report from the Ubuntu Terminal*.

```
    $ ./Qortex-server --license info
```

Details about the license can also be found in the following sections:

- List of licenses. See *Using the License Portal* (page 46).
- Starting the server daemon service. See *Start the Server Daemon Service the First Time* (page 56).

  A valid license returns a report similar to *Figure 12. License Info Report from the Ubuntu Terminal*.

  An invalid license returns a report similar to *Figure 13. License Invalid Report from the Ubuntu Terminal*.

```
user:/opt/quanergy/qortex-server$ ./Qortex-Server --license info
----------------- licensing --------------
Status: Activated, registered to:
USER-NAME
License id: ######
License type: Quanergy premium
Number of license sensors: 15
Expiration Date: 2020-05-20T00:00:00Z
2020-02-14 00:11:16 [info] Log: Reload is called in license validation
```

Figure 12. License Info Report from the Ubuntu Terminal

```
user:/opt/quanergy/qortex-server$ ./Qortex-Server
Looks like software license is not valid.
Error 9203: The license has expired.
Please obtain a valid license and try again.
User the – license command line option to activate.
Contact Quanergy support for more information: support@quanergy.com

2020-02-14 00:11:16 (error) Log: Looks like software license is not valid.
Error 9203: The license has expired.
Please obtain a valid license and try again.
Use the –license command line option to activate.
Contact Quanergy support for more information: support@quanergy.com

2020-02-14 00:11:16 (error) Log: Qortex terminating because of license issue . . .
The license has a problem. Exiting . . .
2020-02-14 00:11:16 (error) Log: The License has a problem. Exiting . . .
```

Figure 13. License Invalid Report from the Ubuntu Terminal

Check a License Version

To check the version of the licensed software:

1. On the QORTEX DTC server host computer, open a terminal (or `ssh`) and log in.

   Enter the username and password for the host machines, if required.

2. Enter the proper directory:

```
    $ cd /opt/quanergy/qortex-server
```

3. Execute the version command, then check the return.

   Windows

```
$ ./Qortex-server –version
```

   Ubuntu

```
user:/opt/quanergy/qortex-server$ ./Qortex-Server –version 2.3.136
```

Manage the License of an Air-Gapped Computer

If the server host computer is air-gapped (no Internet access), use an offline process for managing the licenses. The offline license management process requires two separate computers:

- QORTEX DTC server air-gapped host computer
- Internet-connected computer

1. On the QORTEX DTC server host computer, open a terminal (or `ssh`) and log in.

   Enter the username and password for the host machine, if required.

2. Enter the proper directory:

```
    $ cd /opt/quanergy/qortex-server
```

3. Execute the following command, and enter one of the options at the end:

```
    $ ./Qortex-server --license [activate|refresh|deactivate]
```

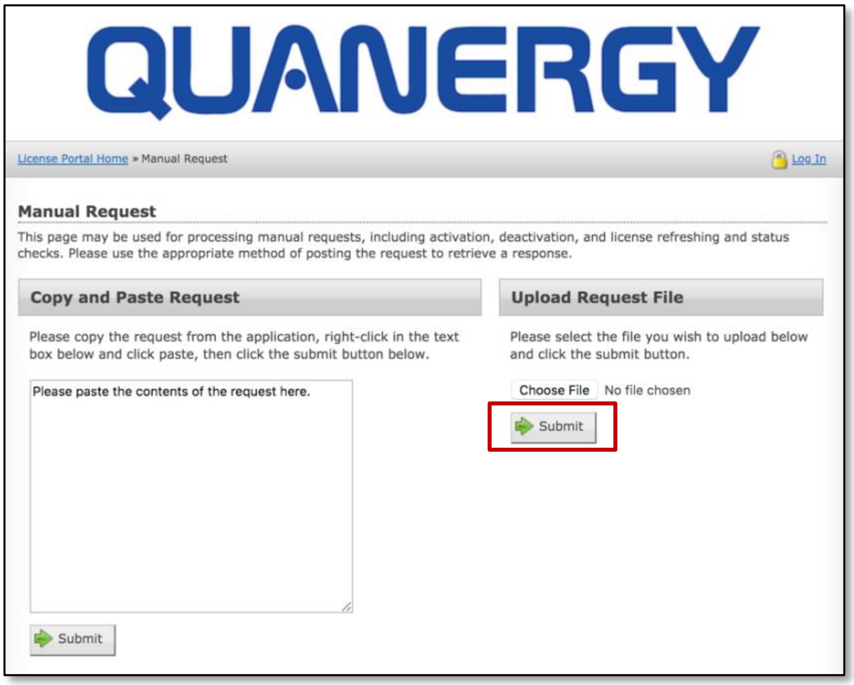
4. Respond to each prompt, and press **Enter**, as follows:

```
Do you want to process your request over the internet? (y/n) [type n]
To proceed without an internet connection, you need a response.xml file
Do you already have a response file? (y/n) [type n]
No worries! We will walk you through obtaining your response.xml file.
For this, you'll first need to generate a request.xml file.
Please enter your License ID: [type 8-digit license number, e.g., 12345678]
Please enter your Activation Password: [type 8-digit alphanum combination, e.g., A1B2C3D4]
Please enter your installation name (optional): [type site name, e.g., MyLab-2]
Wrote License Request to request file at:
home/quanergy/quanergy/qortex/license/request.xml
```

5. Move the `request.xml` file to the Internet-connected computer.
6. In a browser, navigate to this page of the **Customer License Portal**. See *Figure 14. Customer License Portal Manual Request*.

   https://license.quanergyworks.com/solo/customers/ManualRequest.aspx

7. In the **Upload Request File** area, click the **Choose File** button.
8. In the file browser that appears, select the `request.xml` file.
9. In the **License Portal Upload Request File** area, click the **Submit** button.
10. After the credentials are validated, a `response.xml` file is generated by the license server.



Manual Request

This page may be used for processing manual requests, including activation, deactivation, and license refreshing and status checks. Please use the appropriate method of posting the request to retrieve a response.

Copy and Paste Request

Please copy the request from the application, right-click in the text box below and click paste, then click the submit button below.

Please paste the contents of the request here.

Upload Request File

Please select the file you wish to upload below and click the submit button.

Figure 14. Customer License Portal Manual Request

11. Download the `response.xml` file.
12. Move the response.xml file to the target host computer in this filepath:

    `/home/quanergy/quanergy/qortex/license/request.xml`

13. Process the `response.xml` file and select the option needed, as follows:

```
   $ cd /opt/quanergy/qortex-server
   $ ./Qortex-server --license [activate|deactivate|refresh] [Enter preferred option]
```

Respond to each prompt, and press **Enter**, as follows:

```
Do you want to process your request over the internet? (y/n) [Type n]
To proceed without an internet connection, you need a response.xml file
Do you already have a response file? (y/n) [Type y]
Default path to the response.xml file is:
/home/quanergy/quanergy/qortex/license/request.xml
If you wish to use a different path, enter the absolute path to the response.xml
file; otherwise just press enter: [Press the Enter key.]
```

This message returns to confirm that the process was successful:

```
Qortex license has been successfully [activated|deactivated|refreshed].
```

5. Starting and Stopping QORTEX DTC

After all licensing, hardware, and software components are in place, the QORTEX DTC 2.3 system is ready for operation.

The server host computer is designed to run persistently through a Linux daemon service, a non-interactive background program. Even if the server exits unexpectedly, it will immediately restart its processes. Therefore, the procedures in this user guide assume the server is running. The client is able to freely connect to and disconnect from the server at any time, while the server continues publishing surveillance data.

Start QORTEX DTC Summary

1. Login to the QORTEX DTC server host. Open a terminal to access and start the QORTEX DTC server.
2. Login to the QORTEX DTC client host. Open an Ubuntu terminal or click the Windows QORTEX DTC icon and start the QORTEX DTC client.
3. From the QORTEX DTC client, connect to the QORTEX DTC server.
4. If this is the first time any user logs into the QORTEX DTC client, they are prompted to create an admin user with a new password.

Start the Server Daemon Service the First Time

1. Logon to the server host machine using the host machine credentials and open a terminal window.
2. Start the QORTEX DTC daemon service by executing the following command. See *Figure 15. Server: Start, Stop, Status Commands to Control Daemon Service*.

```
$ sudo systemctl start qortex-server.service
```

Although no messages are returned, the QORTEX DTC software is working behind the scenes to set up a new essential computer directory, `/home/quanergy/quanergy/qortex`, where the following important directories and files are placed:

- `datasets` directory stores each recording as a timestamped folder, which contains sequentially numbered index files (*.i00, *.i01, ...), sequentially numbered qlog recording files (*.q00, *.q01, ...), and a copy of the `settings.xml` file in force at the time of recording. See *Recording Data* (page 153).
- `license` directory stores the license files as `qortex_licenseAlias1.lfx`, `qortex_licenseAlias2.lfx`, `qortex_licenseFile.lfx`, and `secure.vault.lfx`. Leave these files as they are. See *Managing the License* (page 45).

- `location` directory stores the `settings.xml` file that is created and updated based on changes made in Configuration mode. See *Entering Configuration Mode* (page 84).
- `qortex.server.status.xml` file is used by the server to recover its status in the event of a crash. Leave this file as is. See *Restart the Server Daemon Service* (page 57).

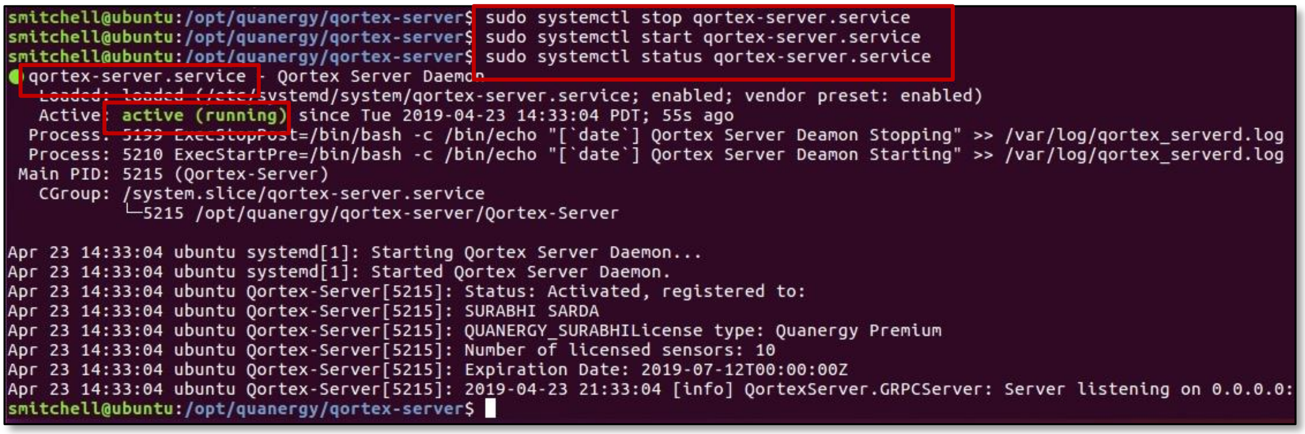
Restart the Server Daemon Service

If the server-client interaction behaves in unexpected ways, it might help to restart the daemon service. See *Figure 15. Server: Start, Stop, Status Commands to Control Daemon Service*.

1. Logon to the server host machine using host machine credentials and open a terminal window.
2. Execute the `restart` command:

```
$ sudo systemctl restart qortex-server.service
```

If the restart is successful, no messages are returned.



```
smitchell@ubuntu:/opt/quanergy/qortex-server$ sudo systemctl stop qortex-server.service
smitchell@ubuntu:/opt/quanergy/qortex-server$ sudo systemctl start qortex-server.service
smitchell@ubuntu:/opt/quanergy/qortex-server$ sudo systemctl status qortex-server.service
● qortex-server.service - Qortex Server Daemon
   Loaded: loaded (/etc/systemd/system/qortex-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2019-04-23 14:33:04 PDT; 55s ago
  Process: 5199 ExecStopPost=/bin/bash -c /bin/echo "[`date`] Qortex Server Deamon Stopping" >> /var/log/qortex_serverd.log
  Process: 5210 ExecStartPre=/bin/bash -c /bin/echo "[`date`] Qortex Server Deamon Starting" >> /var/log/qortex_serverd.log
 Main PID: 5215 (Qortex-Server)
   CGroup: /system.slice/qortex-server.service
           └─5215 /opt/quanergy/qortex-server/Qortex-Server

Apr 23 14:33:04 ubuntu systemd[1]: Starting Qortex Server Daemon...
Apr 23 14:33:04 ubuntu systemd[1]: Started Qortex Server Daemon.
Apr 23 14:33:04 ubuntu Qortex-Server[5215]: Status: Activated, registered to:
Apr 23 14:33:04 ubuntu Qortex-Server[5215]: SURABHI SARDA
Apr 23 14:33:04 ubuntu Qortex-Server[5215]: QUANERGY_SURABHILicense type: Quanergy Premium
Apr 23 14:33:04 ubuntu Qortex-Server[5215]: Number of licensed sensors: 10
Apr 23 14:33:04 ubuntu Qortex-Server[5215]: Expiration Date: 2019-07-12T00:00:00Z
Apr 23 14:33:04 ubuntu Qortex-Server[5215]: 2019-04-23 21:33:04 [info] QortexServer.GRPCServer: Server listening on 0.0.0.0:
smitchell@ubuntu:/opt/quanergy/qortex-server$
```

Figure 15. Server: Start, Stop, Status Commands to Control Daemon Service

Control the Server Daemon Service

Once daemon service is active, the server runs persistently (and restarts automatically as necessary). It should never need to be manually started or stopped unless downloading a new version or dealing with the license server is necessary.

To restart the service (which both stops and starts the service), or discretely stop and then start the service:

1. Logon to the server host machine using host machine credentials and open a terminal window.
2. Execute the `restart`, `start`, or `stop` command, as needed.

```
    $ sudo systemctl [restart|start|stop] qortex-server.service
```

If the command is successful, no messages are returned. See *Figure 15. Server: Start, Stop, Status Commands to Control Daemon Service*.

Check the Status of the Server Daemon Service

To see the status of the daemon, including any error messages:

1. Logon to the server host machine using host machine credentials and open a terminal window.
2. Execute the commands:

```
    $ sudo systemctl status qortex-server.service
```

If the daemon is running (server is operational), this return appears. See *Figure 15. Server: Start, Stop, Status Commands to Control Daemon Service*.

```
    qortex-server.service  - Qortex Server Daemon
    ...
    Active: active (running) since <Date>
    Main PID: <number>
```

If the daemon is not running (server is not operational), this return appears.

```
    qortex-server.service
    ...
    Active: inactive (dead)
```

Start the Client on a Windows Host

1. On the Qortex DTC server host, verify the Qortex DTC server is running. See *Check the Status of the Server Daemon Service* (page 58).

```
$ sudo systemctl status qortex-server.service
```

In the response, look for the phrase `Active: active (running) since <Date>.`

2. Logon to the client Windows host using host machine credentials.
3. Start the client. Double-click the QORTEX DTC client icon on the Windows Desktop.

   Additional Windows 10 methods to locate the QORTEX DTC client application include: file browser, **Search** field, or Windows key.

   The QORTEX DTC client and client terminal open in their own window. See *Figure 32. Client Start Options and Initial Blank View*.

By default, the QORTEX DTC client does not require a login. The initial display is a blank screen. To view any data requires you connect the client to a server. See *Connect the Client to the Server* (page 61).

4. If this is the first time anyone has logged into the Qortex DTC client, you are prompted to create an `Admin` user. See *Login to Client for the First Time* (page 59).
5. If **Security** mode is enabled, enter your credentials in the **Sign into your account** window.

Start the Client on an Ubuntu Host

1. On the Qortex DTC server host, verify the Qortex DTC server is running. See *Check the Status of the Server Daemon Service* (page 58).

```
$ sudo systemctl status qortex-server.service
```

In the response, look for the phrase `Active: active (running) since <Date>.`

2. Logon to the client Ubuntu host and open a terminal window.
3. Execute the command from the root directory:

```
    $ /opt/quanergy/qortex-client/Qortex-Client
```

The QORTEX DTC client and client terminal open in their own window. See *Figure 32. Client Start Options and Initial Blank View*.

By default, the QORTEX DTC client does not require a login. The initial display is a blank screen. To view any data requires you connect the client to a server. See *Connect the Client to the Server* (page 61).

4. If this is the first time anyone has logged into the Qortex DTC client, you are prompted to create an `Admin` user. See *Login to Client for the First Time* (page 59).
5. If **Security** mode is enabled, enter your credentials in the **Sign into your account** window.

Login to Client for the First Time

During the first-time login, you are directed to create a new user. This is an `Admin` user by default.

After the first-time login, Admin or Editor user type credentials are required to access the QORTEX DTC server through the QORTEX DTC client.

1. Start the QORTEX DTC client.
   - Windows: double-click the QORTEX DTC client icon.
   - Ubuntu: from a terminal, run: `/opt/quanergy/qortex-client/Qortex-Client`

2. From the **Configuration** mode panel, click **CONFIRM**.
3. From the **Server** tab > **Security** panel, toggle the **Security Mode ON**.
4. In the **Sign into your account** panel, enter your **License ID** and **License password**.

   If you do not know your license ID and license password:

   a. Click **Locate Server License**.

   b. Browse through to find your Quanergy license and locate the License ID and License Password.

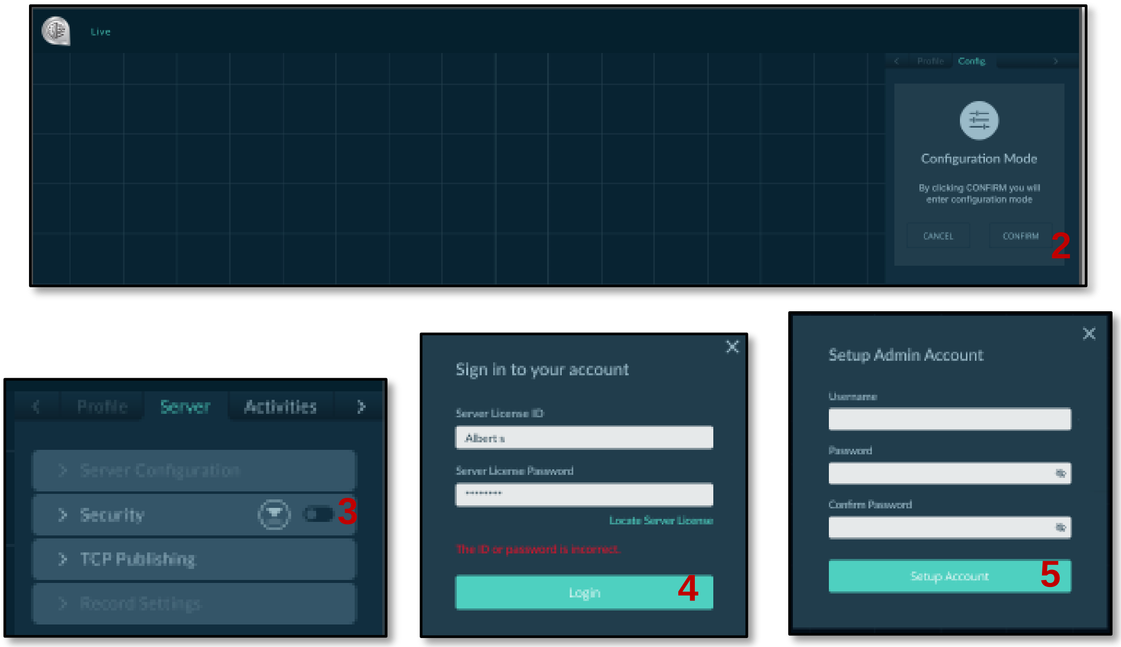
   c. From the command line, retrieve the license number.

```
$ ./Qortex-Server -license info
```

   d. From the License Portal, enter your license number and retrieve the Activation Password.

5. In the **Setup Admin Account**, enter a username and password and click **Setup Account**. Credential requirements are:
   - **Username criteria**: Contains at least 6 characters and contain alphabetic, numeric, and special characters. It cannot contain a space.
   - **Password criteria:** Contains at least 6 characters and must contain at least one upper case letter, 1 lowercase letter, 1 numeric, and 1 special character. It cannot contain a space. Do not include the username as part of the password.

   These credentials are saved on the configured QORTEX DTC server. The next time the new user connects the QORTEX DTC client to the QORTEX DTC server, they are prompted to provide these login credentials for that QORTEX DTC server.



Server License ID: Albert s

Server License Password: ••••••••

The ID or password is incorrect.

Figure 16. First Time Login – Create Admin User

6. Verify the new user credentials.
   a. From the **Profile** tab, verify the new user credentials. Click **Change** to alter them.
   b. Click **Logout**.
   c. In the **Login** panel, enter the credentials and click **Login**.

Stop the Client

On the client host, stop and dismiss the client software using any of these methods. See *Figure 17. Stop Client via Interface Window Close Menu or Close Button*.

1. Click the red Close ☒ button at the right top corner.
2. Click the top left corner of the client window title bar to open the standard Windows menu, then select the **Close** item (or press the Alt+F4 keyboard equivalent.
3. Click the command / terminal window that launched the client, then press **Ctrl+C** (Windows) or **Ctrl-Q** (Ubuntu) on the keyboard.

   You might need to resize or minimize the client to see the command / terminal window.

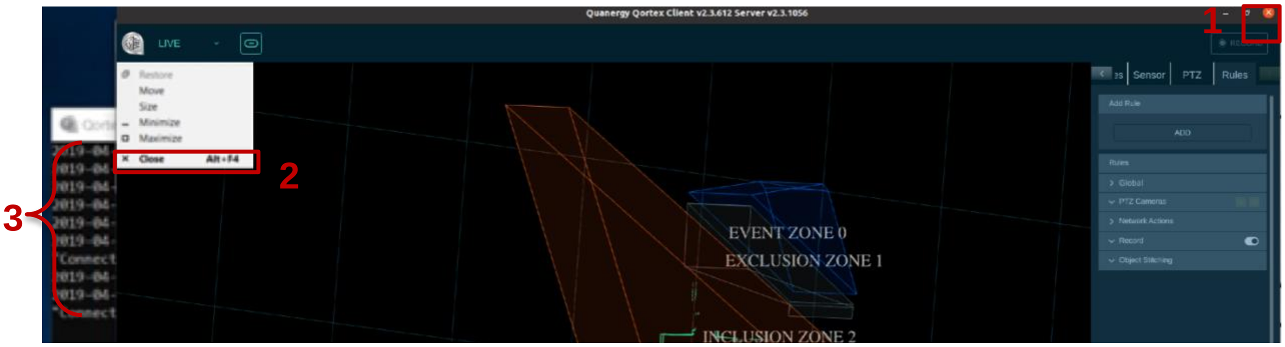


Figure 17. Stop Client via Interface Window Close Menu or Close Button

Connect the Client to the Server

If the client is on an Ubuntu system, you provide the server IP address when the client is started for the first time. If the client is on a Windows system, the server is connected after the client starts. See *Figure 18. Client Window First Time Start*.

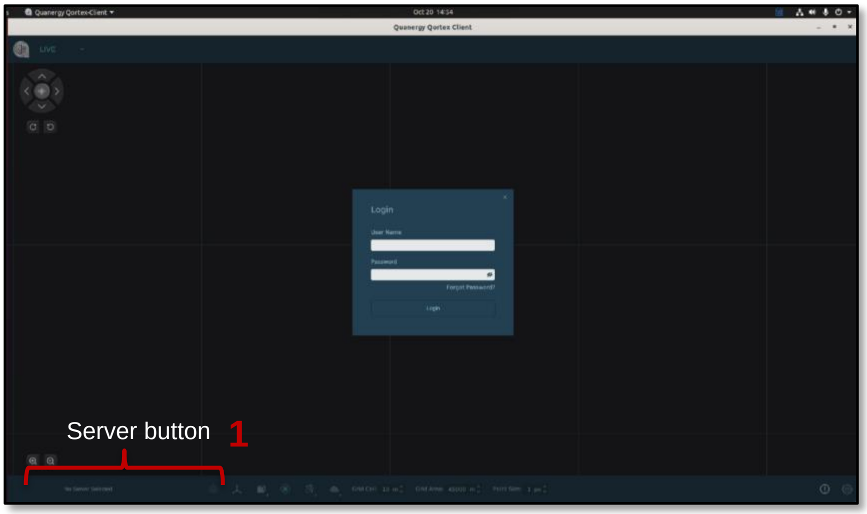


Figure 18. Client Window First Time Start

A client can list one or more servers to connect to, but the client can connect to only one server at a time. It receives data from the selected server only.

Specify a server the client can connect to:

1. From the QORTEX DTC client interface, click the **No Server Selected** button. See *Figure 18. Client Window First Time Start*.

   If servers were already added and configured, the button displays the list of servers previously added.

2. From the **Add Server** panel, click the **Add New Server** button. See *Figure 19. Add Server Sequence*.

3. Specify the server information.

   a. In the **Server Name** field, type a nickname for the server.

   b. In the **Server IP address** field, type the server IP address.

      QORTEX DTC supports both IPv4 and IPv6 IP addresses.

      For discovery of the IP address, see *Connect and Start the Host Computers* (page 34).

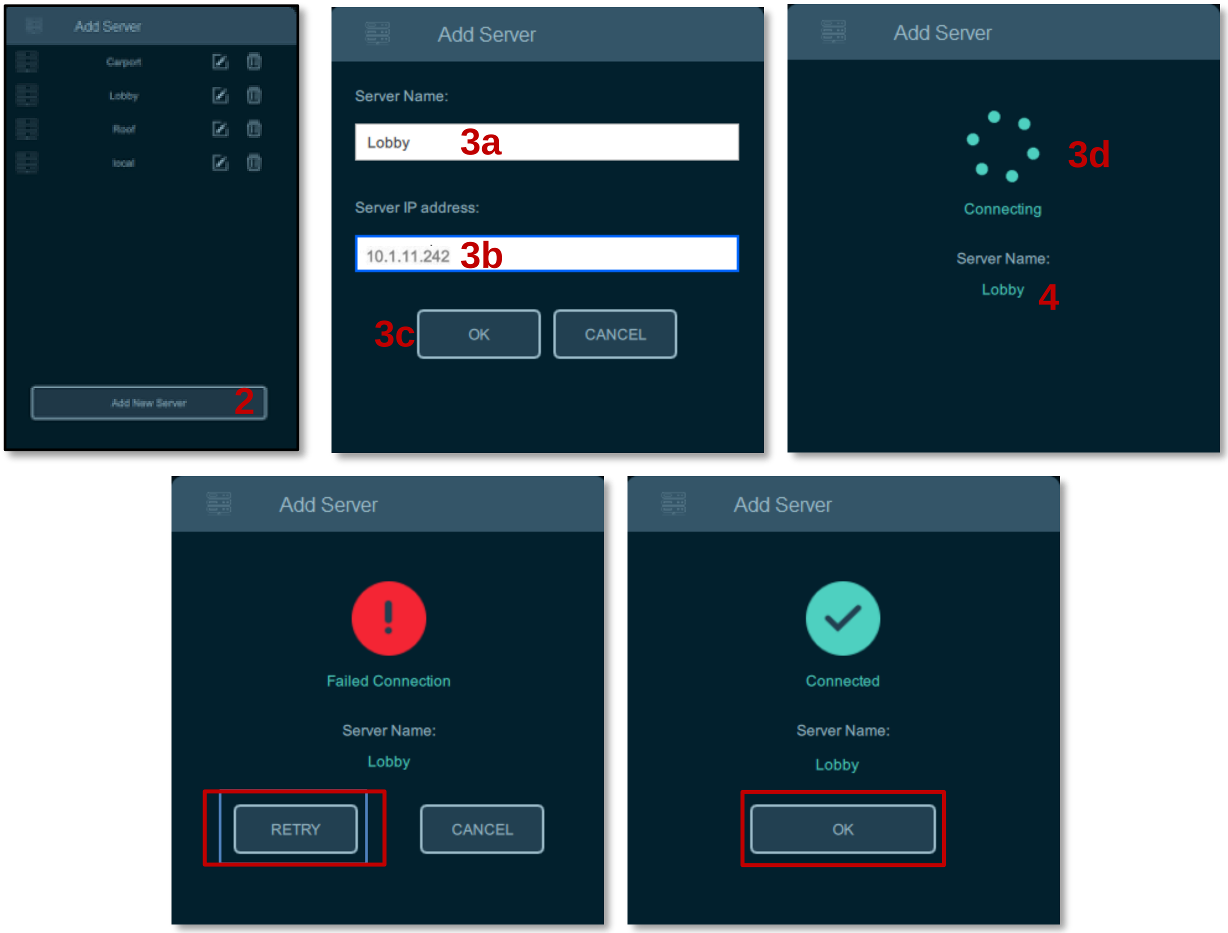
   c. Click **OK**. The client immediately tries to connect to the server. It displays a **Busy** ⁙ button until it connects or fails to connect.

      If the message is **Connected**, click **OK**.

      If the message is **Failed Connection**, choose a method:

      - Click the **RETRY** button to try the same settings again.
      - Click **CANCEL** button > **Edit** icon ✎ and correct the server IP address.

4. Click the **Esc** button on the keyboard to minimize a populated panel. The name of the connected server is the button that reopens the panel.



Server Name: Lobby

Server IP address: 10.1.11.242

Connecting

Server Name: Lobby

Failed Connection

Server Name: Lobby

Connected

Server Name: Lobby

Figure 19. Add Server Sequence

6. Security Mode

QORTEX DTC 2.3 adds support for user authentication and data encryption. This is an optional mode for the QORTEX DTC server. By default, **Security** mode is `off` . Enable it through the QORTEX DTC client in **Configuration** mode.

- **HTTP Digest** authentication protocol—Internet access is not required. Certificate of Authority (CA) is not required. Authentication setting applies for the duration of the QORTEX DTC client session.
- **AES-256** encryption—The object list, zone list, counter line, and sensor health APIs are encrypted.

When Security mode is enabled, user authentication is required for the following actions:

- Connecting the QORTEX DTC client to the QORTEX DTC server
- Connecting any third-party client to the QORTEX DTC server
- Sending and receiving gRPC API calls.
- Reading QORTEX DTC API (such as, object list, zone list, counter line, sensor health).
- Anytime an application is reading the QORTEX API TCP ports. The Quanergy TCP listener includes HTTP digest and AES-256 support. See *Server TCP Ports to Monitor* (page 92).

To enable the QORTEX DTC server **Security** feature, install QORTEX DTC server and Debian, and activate or refresh the Quanergy license.

Toggle Security Mode On/Off

1. From the **Configuration** mode panel > **Server** tab, expand **Security Login** panel header, click the **Security** toggle. See *Figure 24. Security: Login Toggle On*

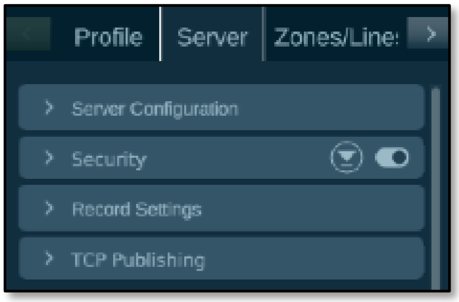
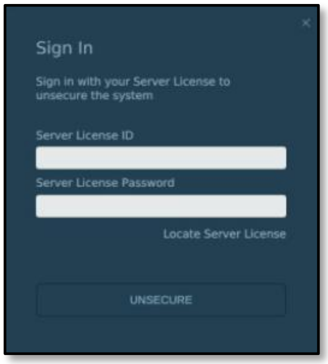


Figure 20. Security: Login Toggle On

2. You are prompted to sign in as an `Admin` user.
3. Click to confirm the change. See *Figure 25. Security: Toggle On and* SET SECURED.

- Toggle Security mode OFF , click **UNSECURE**.



Sign In

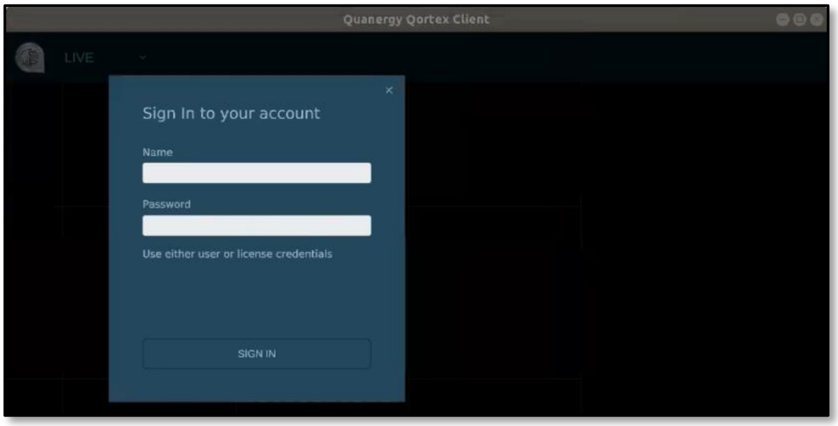
Sign in with your Server License to unsecure the system

Figure 21. Security: Toggle Off and UNSECURE

Sign In to QORTEX DTC Server with Security Mode Enabled

When **Security** mode is enabled, when you login to the QORTEX DTC client and want to connect to the QORTEX DTC server, you are prompted to provide your login credentials. See *Figure 22. Security: Sign-In*.

To sign in to the QORTEX DTC server, enter either the credentials you created in the **Create User** panel of the Security Setup flow or use your license ID and license password. See ***Error! Reference source not found.*** (page 65). *Login to Client for the First Time* (page 59).



Sign In to your account

Use either user or license credentials

Figure 22. Security: Sign-In

Change User Credentials for Security Mode

You can change the username or password through the **Security Login** panel. It is not used unless the **Security** mode is ON . See *Figure 23. Security: Change User Name or Password*.

1. From the **Configuration** mode panel > **Profile** tab.

2. Change the Username. Click **Change** in the **User Name** panel.
3. Change the Password. Click **Change** in the **Password** panel.
4. The Security Login panel expands to list the authentication methods.
5. Enter new **User Name** or **New Password**
6. Click **Apply All** at the bottom of the **Server** tab.

   After you change the username or password, you are returned to **Monitor** mode. Enter **Configuration** mode again, as needed. Restarting the QORTEX DTC client is not required.

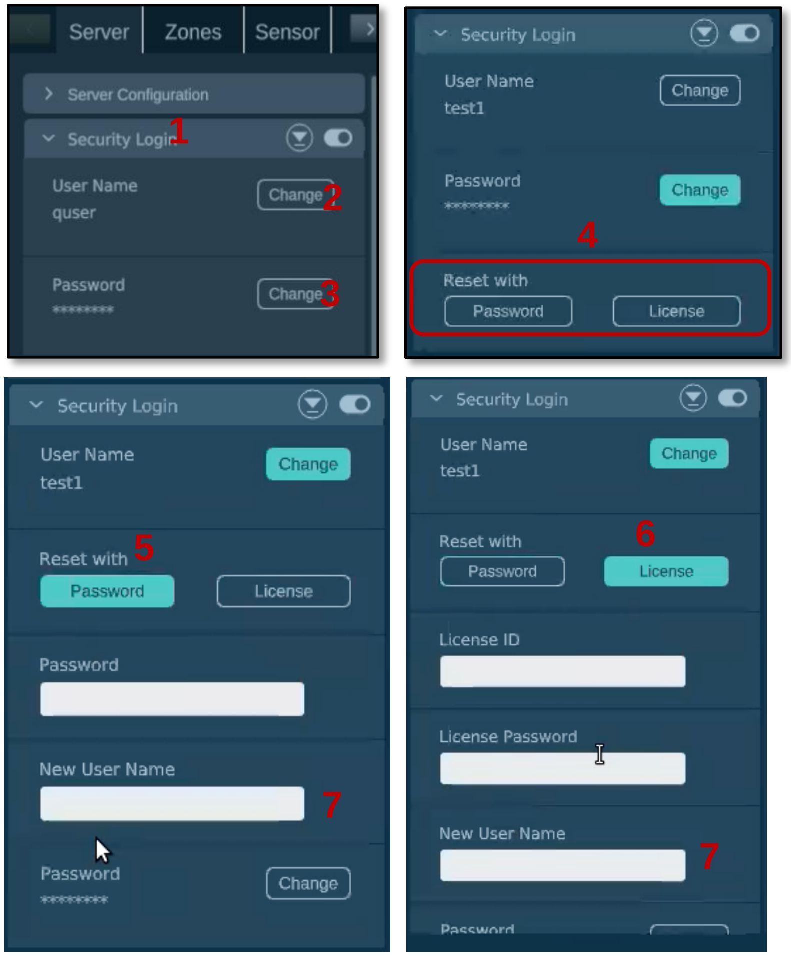


Figure 23. Security: Change User Name or Password with Security Mode On

7. From the **Configuration** mode panel > **Server** tab, in **Security** panel header, click the **Security** toggle. See *Figure 24. Security: Login Toggle On*

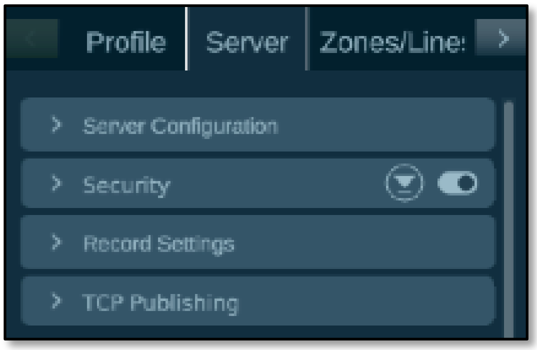


Figure 24. Security: Login Toggle On

8. You are prompted to **Login** as an `Admin` user.

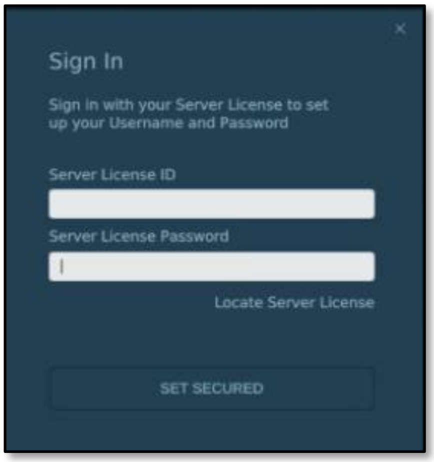
   Sign in using your `Admin` user credentials.

9. In the Configuration Mode panel, click **CONFIRM**.

10. Click the **Security** toggle again to turn Security mode ON .

11. Click to confirm the change. See *Figure 25. Security: Toggle On and SET SECURED*.

    If you toggle Security mode ON, click **SET SECURED**.



Sign In

Sign in with your Server License to set up your Username and Password

Server License ID

Server License Password

Locate Server License

SET SECURED

Figure 25. Security: Toggle On and SET SECURED

7. User Management

User management include creating, updating, enabling users with Secure mode and without Secure mode.

User Types and Permissions

QORTEX DTC client has to user types based on roles:

- Viewer—General user with permissions to:
    - Monitor QORTEX DTC.
    - Request a password change.
    - Change your password.
- Editor—General user with permissions to:
    - Monitor QORTEX DTC.
    - Configure QORTEX DTC. Re-enter credentials to access Configuration Mode.
    - Request a password change.
    - Change your password.
- Admin—Administrative user has permissions to:
    - Monitor QORTEX DTC.
    - Configure QORTEX DTC. Re-enter credentials to access Configuration Mode.
    - Manage QORTEX DTC users.
        - Add, delete users
        - Enable, disable users
        - Reset user passwords
        - Edit usernames and roles
        - Edit own username and password
    - Download, view, and edit the User Log.
    - Enable/disable Security mode.
    - Modify global QORTEX DTC Settings
        - Secure TCP ports
        - Password expiry

User Credential Requirements

Username criteria: Contains at least 6 characters and contain alphabetic, numeric, and special characters. It cannot contain a space.

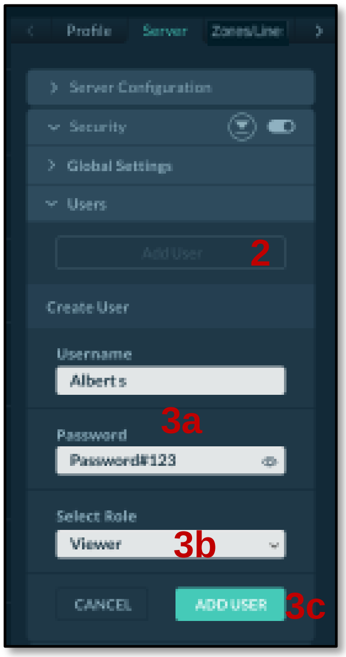
Password criteria: Contains at least 6 characters and must contain at least one upper case letter, 1 lowercase letter, 1 numeric, and 1 special character. It cannot contain a space. Do not include the username as part of the password.

When a user is created, these credentials are saved on the configured QORTEX DTC server. The next time the new user connects the QORTEX DTC client to the QORTEX DTC server, they are prompted to provide these login credentials for that QORTEX DTC server.

Create New Users

1. Login to Qortex DTC client as an `Admin` user.
2. Select **Server** tab > **Users** tab and click **Add User** button.
3. In the **Create User** panel:
   a. Enter a **Username** and **Password**.
   b. From the menu select, **User Role** for the new user. The options are `Viewer`, `Editor`, or if you are an Admin user you also might see the `Admin` option.
   c. Click **Add User**.

   The new user is listed under the **Server** > **Users** panel.



Username: Alberts

Password: Password#123

Select Role: Viewer

Figure 26. Add User Button and Create User Panel

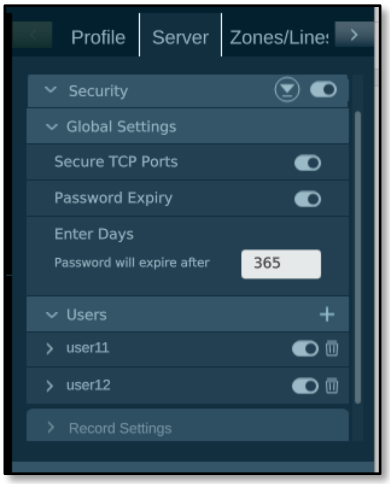
Set User Password Expiry

The timing for users to reset their passwords is applied globally.

> **Note:** User passwords expires only when Security Mode is ON.

1. Login to Qortex DTC client as an `Admin` user.
2. Select **Server** tab > **Global Settings**.
3. Toggle **Password Expiry** to **On**.
4. Enter the how often passwords must be reset, in the **Password will expired after** field.

   This is defined by number of days. For example, if you enter 90 days, then users must reset their password every 90 days.



Secure TCP Ports

Password Expiry

Enter Days

Password will expire after 365

Figure 27. User Password Expiration Global Settings Panel

Reset Your Own Password

When password expiry is on, when you login to the Qortex DTC client, you receive a message in the Status panel that notifies you when your password is set to expire.

To change your unexpired password:

1. Login to Qortex DTC client with your username.
2. Select **Profile** tab and click **Change**.
3. In the expanded **Profile** panel, enter your current (old) password.
4. Enter your new password in both the **New Password** and **Confirm Password** fields.
5. Click **Logout**.
6. Relog back in with the new credentials.

To change your expired password, contact your administrator to reset your password for you.

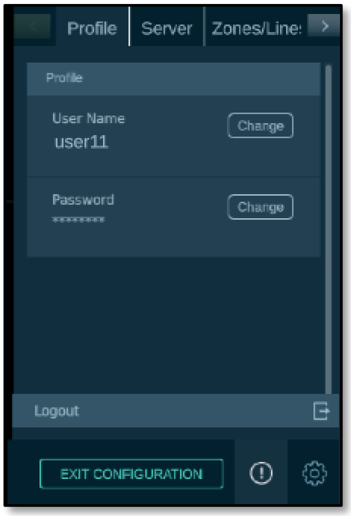


Figure 28. User Profile Panel

Reset User Passwords

1. Login to Qortex DTC client as an `Admin` user.
2. Select **Server** tab > **Users** tab.
3. Expand the username to edit, click **Reset Password**.
4. In the **New Password** field, enter a **Password**.
5. Click **Apply Changes**.

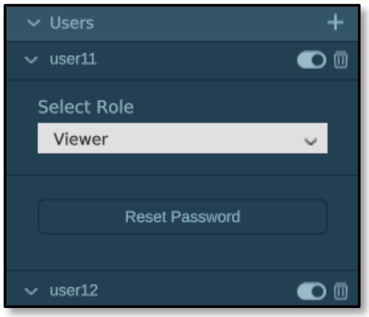
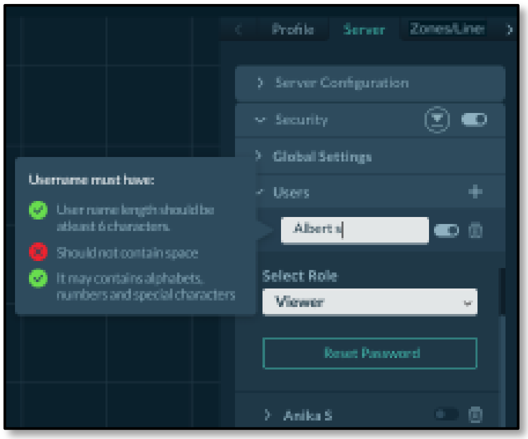


Figure 29. User Reset Password

Change Usernames and Roles

1. Login to Qortex DTC client as an `Admin` user.
2. Select **Server** tab > **Users** tab.

3. Expand the username to edit.
4. Click the username to open the username field. Type the new username.
5. Open the **Select Role** menu and select an alternative role for the user.
6. Click **Apply Changes**.



Username must have:

- User name length should be atleast 6 characters.
- Should not contain space
- It may contains alphabets, numbers and special characters

Albert s

Select Role: Viewer

Figure 30. Edit Username and Role User Panel

Disable and Enable Users

1. Login to Qortex DTC client as an `Admin` user.
2. Select **Server** tab > **Users** tab.
3. Click the toggle icon **On** or **Off** next to the username.
   - Toggle **On** enables the user.
   - Toggle **Off** disables the user.

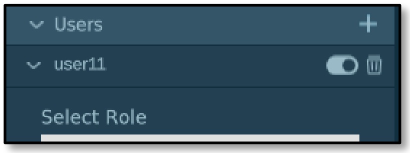


Figure 31. User Disable and Delete Options

Delete Users

1. Login to Qortex DTC client as an `Admin` user.
2. Select **Server** tab > **Users** tab.
3. Click the trashcan icon next to the username you are deleting.

   The user is deleted from the system.

Recover Admin Credentials with Security Mode Enabled

From the QORTEX DTC server command line as an `admin` user, enter:

```
./Qortex-Server --auth reset-admin --lic-id 7779898 --lic-pwd KXM39PFC
   --adm-id UserName563@ adm-pwd PwdPwd!!34
```

Entering the QORTEX DTC license ID and license password enables you to create a new Admin username and password.

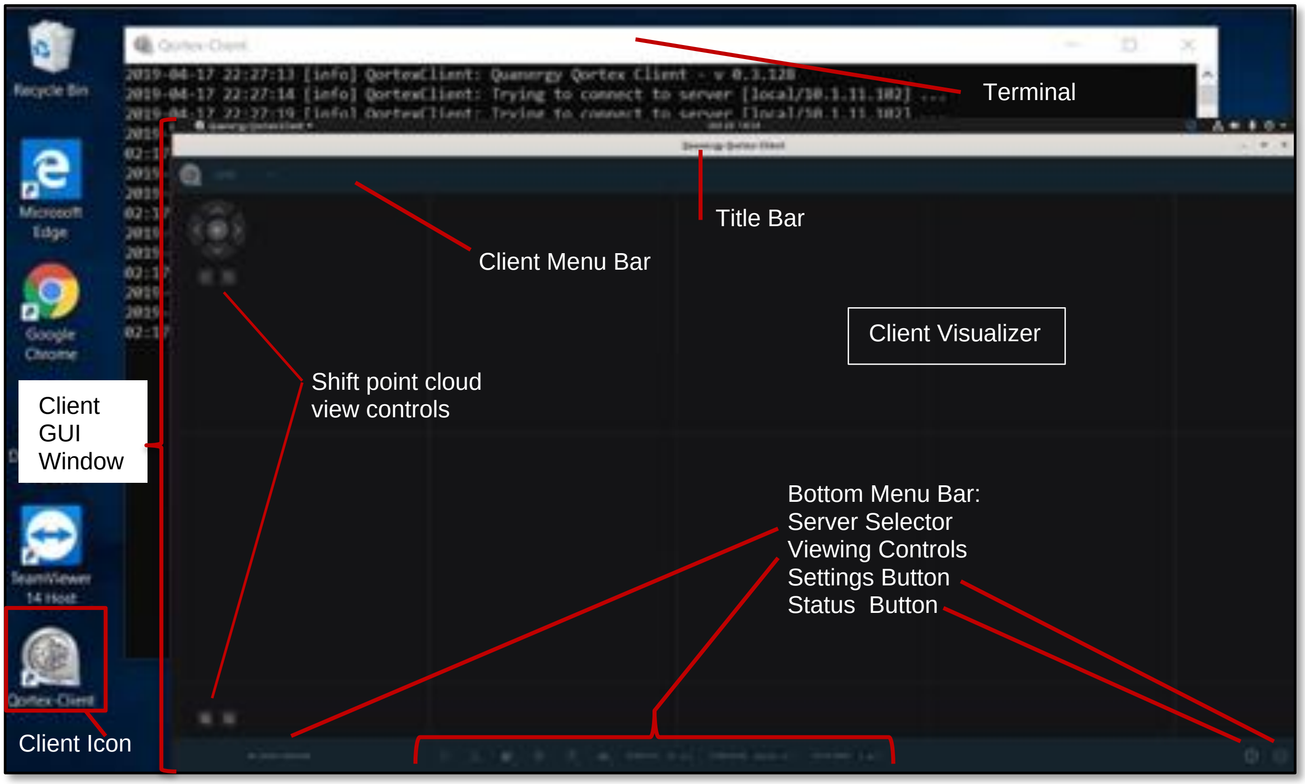
8. QORTEX DTC Client Interface

Launching the QORTEX DTC client from the client starts the client processes and opens the QORTEX DTC client graphical user interface (GUI). It consists of several areas of functionality, which are described in this section.

Figure 32. Client Start Options and Initial Blank View shows one of the QORTEX DTC client start options and the initial QORTEX DTC client window. For starting the client, either through a terminal or the client icon, see *Start the Client on a Windows Host* (page 58) and *Start the Client on an Ubuntu Host* (page 59).

The first time the QORTEX DTC client starts, you need to add a QORTEX DTC server. When the QORTEX DTC client starts subsequently, the automatically connects to the last connected server.

Version information is stated in the title bar. When the client is started for the first time, it is in an unconnected and unconfigured state, only the client version is displayed. When the client is connected to a server, the server version is also displayed.



```
2019-04-17 22:27:13 [info] QortexClient: Quanergy Qortex Client - v 0.3.128
2019-04-17 22:27:14 [info] QortexClient: Trying to connect to server [local/10.1.11.102] ...
2019-04-17 22:27:19 [info] QortexClient: Trying to connect to server [local/10.1.11.102]
```

Figure 32. Client Start Options and Initial Blank View

Hover Tool Tips

Hovering over most interface widgets display a tool tip pop up that summarizes its purpose. See *Figure 33. Client Tool Tip Example*.

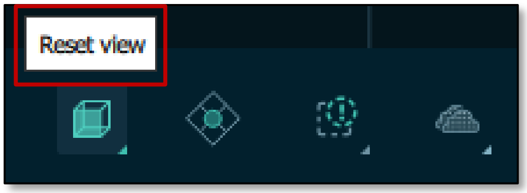


Figure 33. Client Tool Tip Example

Top Bar Options

The QORTEX DTC client top menu bar has both static decorations and interactive widgets. See *Figure 34. Client Top Bar, Data Selector: LIVE or PLAYBACK, Connect*.

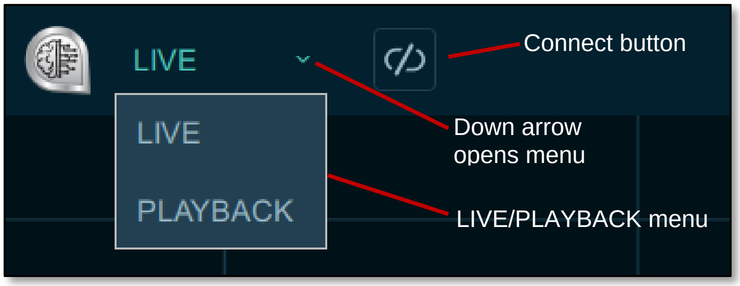


Figure 34. Client Top Bar, Data Selector: LIVE or PLAYBACK, Connect

Data Selector

Next to the QORTEX DTC icon is the **Data Selector** area. Select and view the type of data being visualized:

- **LIVE** (real-time data mode) is displayed by default. **LIVE** data is visualized according to the `settings` file generated in **CONFIGURATION** mode. See *Entering Configuration* Mode (page 84) and *Visualizing Data* (page 114).
- **PLAYBACK** (previously recorded data mode) provides a list of recordings. Recordings play according to their own `settings` file. See *View Recorded (PLAYBACK) Data* (page 156).

Connect Button

When a server is selected and the client is in **LIVE** data mode, the **Connect** button appears. The Connect button is a toggle that connects and disconnects the server from the sensors. It has three states:

Connected status indicates the server is connected to sensors. When connected, data is visualized from the sensor location. Click **Connected** button to disconnect from the sensors.

A **Busy** button appears while the server tries to connect to sensors, or when a sensor connection is busy.

Disconnected status indicates the server is disconnected from sensors. When disconnected, the server remains connected to the client, but live data from the sensors is not visualized. Configuration mode requires the server be disconnected from the sensors. Click **Disconnected** button to reconnect to sensors.

Record Button

The Record button indicates if the LIVE data is being recorded. It also indicates in you can manually record LIVE data. See *Recording Data* (page 153).

The Record button is gray when inactive. The Record button is red and has a time elapsed counter when active. See *Figure 35. Client Record Active / Inactive Button*.

Manual recording requires the event recording option, under **Configuration** panel **> Rules** tab **> Record** toggle is `OFF`. See *Manually Start a Recording* (page 155).

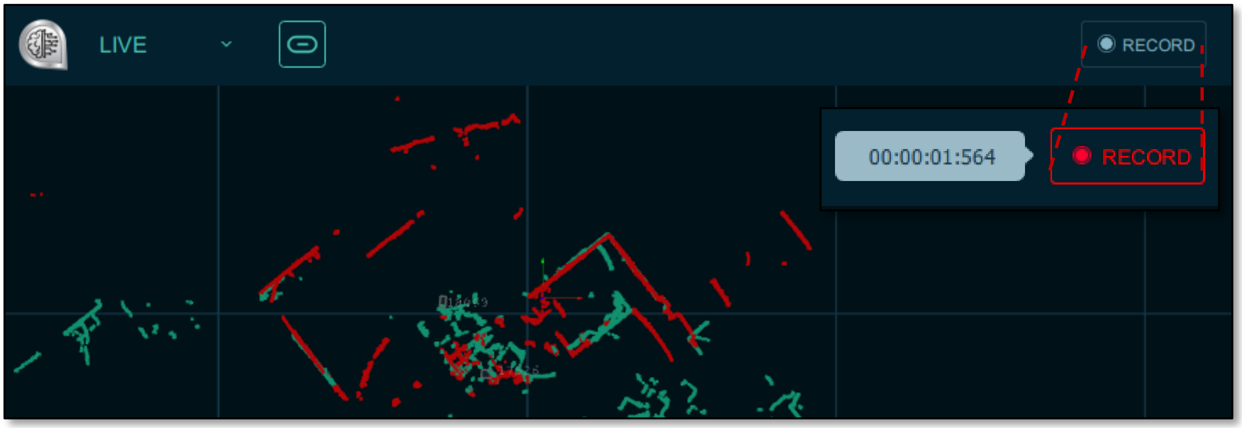
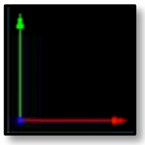
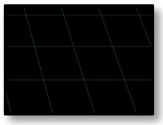
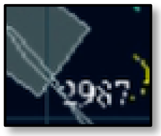
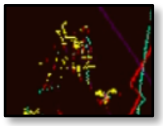
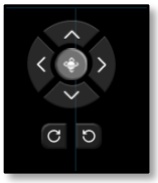
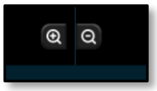
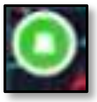
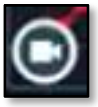
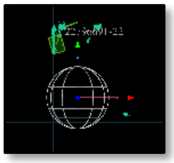


Figure 35. Client Record Active / Inactive Button

Central Visualizer Window Elements

The client Visualizer window is where data that is collected by the sensor and processed by the server is visualized. This Visualizer displayed data, whether displayed in real time or replayed from QLogs. See *Figure 36. Client Object Types and Zones*, *Figure 37. Client Visualizer Window,* and *Table 6. Visualizer Elements and Colors*.

Table 6. Visualizer Elements and Colors

Element	Color	Description
Axes		Show position/orientation for each sensor when QORTEX DTC server is in disconnected state or Sensor Health is turned off.
Background grid		Indicates the scale for the visualized data. Blue lines on black background.
Object unique ID numbers		Assigned when a new object is detected
Point clouds		Collection of data points, displayed in **CONFIGURATION** mode only. One color assigned to each sensor.
Point cloud move 3D view controls	 	Shift the point cloud view. Shift left, right, up, down. Set top front view. Enlarge or shrink view.
Sensor and PTZ Camera icons	 	Indicate the position of sensors and PTZ cameras.
World coordinate axis		Display where the World Coordinate origin and axis are aligned relative to a physical location in the point cloud.

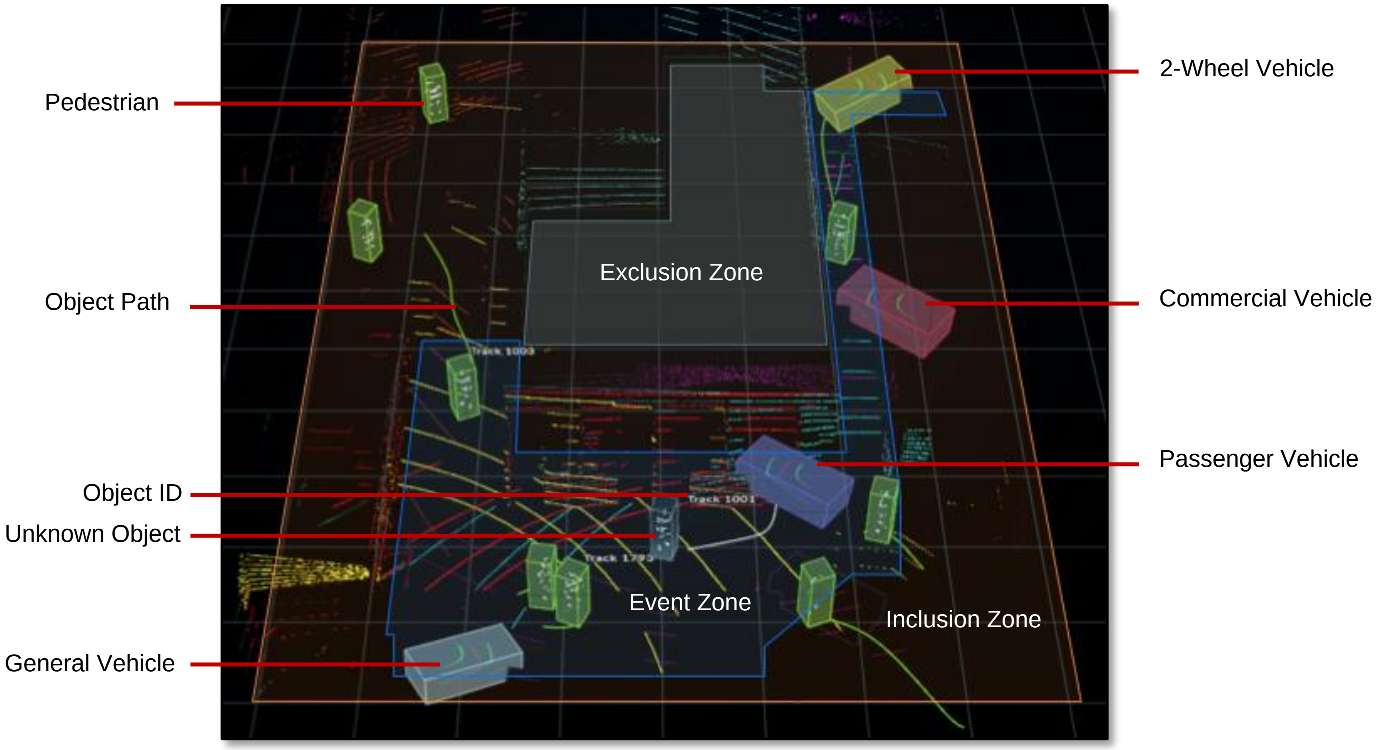


Figure 36. Client Object Types and Zones

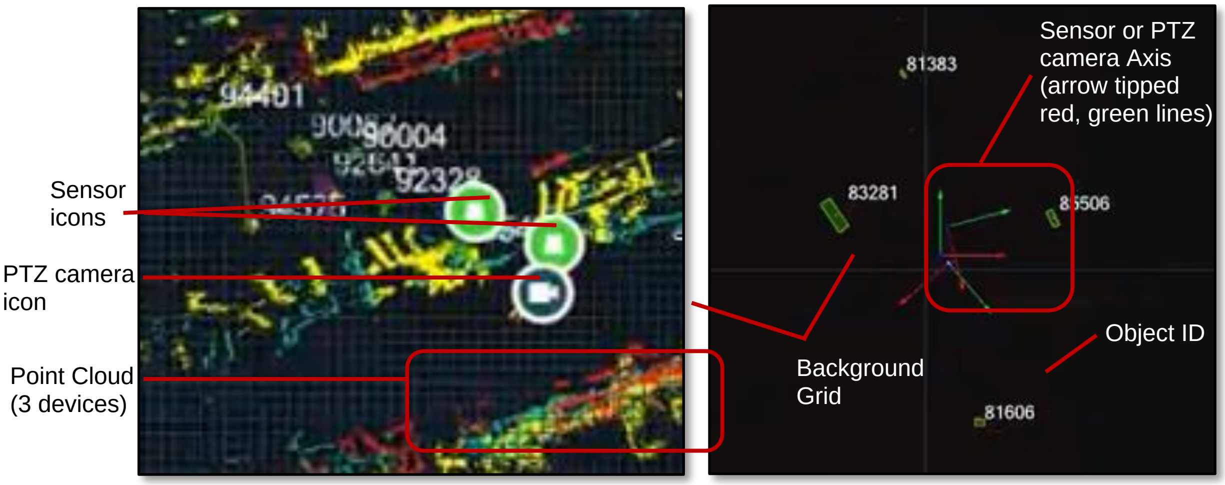
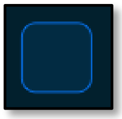
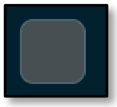
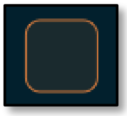
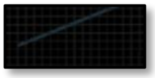


Figure 37. Client Visualizer Window Elements

Visualizer Zones/Lines

Zones/Lines are areas for tracking or excluding tracking objects. See *Table 7. Visualizer Zone Types, Counter Line, and Colors*.

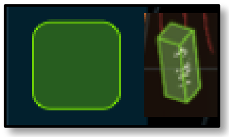
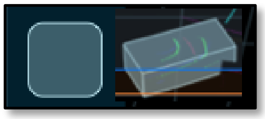
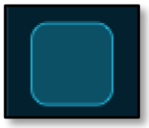
Table 7. Visualizer Zone Types, Counter Line, and Colors

Element	Color	Description
Event zones		Areas for triggering actions (Events) based on Rules.
Exclusion zones		Areas where points and tracks are filtered out.
Inclusion zones		QORTEX DTC only monitors this area, areas outside of the Inclusion zone are filtered out. Event zones and Exclusion zones can be layered within Inclusion zones.
Counter lines		Counter lines indicate a path where objects are counted when crossing the line in one direction or the other.

Visualizer Tracked Objects

Tracked objects are represented as cubes or rectangles with trailing tails to show movement. Objects are tracked as they pass from one LiDAR field of view (FOV) to another. They are classified by their bounding box as indicated. See *Table 8. Visualizer Tracked Objects and Colors*.

Table 8. Visualizer Tracked Objects and Colors

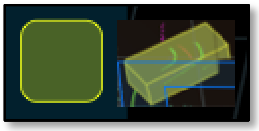
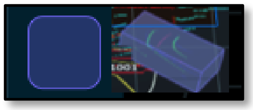
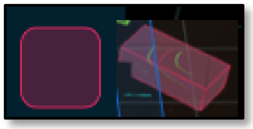
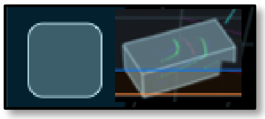
Element	Color	Description
Pedestrian		**Green** a person.
Vehicle		**Gray** a general vehicle if the subvehicle categories are not activated or is unknown.
Unknown		**Blue** an unknown object.

Visualizer Sub-Vehicles

Sub-vehicles use additional colors to identify them. The sub-vehicle classes supported are based on the United State Department of Transportation, Federal Highway Administration, Traffic Monitoring Guide.

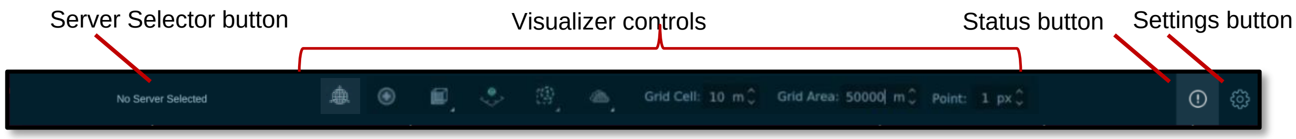
If the `SubVehicle` license is activated and the `useSubVehicleClassication` toggle is `True`. QORTEX DTC sub-classifies vehicles, based on size, into three categories: **two-wheeler** vehicles, **passenger** vehicles, and **commercial** vehicles. If a proper subclassification cannot be determined, QORTEX DTC classifies a vehicle in the parent **general** vehicle class. See *Activate a License* (page 47) and *Table 9. Visualizer Sub-Vehicles and Colors*.

Table 9. Visualizer Sub-Vehicles and Colors

Element	Color	Description
Two-wheel vehicle		**Light Green, Class 1 Motorcycles**
Passenger vehicles		**Dark Blue, Class 2 and 3 Passenger Cars**
Commercial vehicles		**Magenta, Class 4 and above Buses and Trucks**
General vehicle		**Gray**, if sub-class cannot be determined.

Bottom Bar Options

The bottom bar is loaded with important features and essential functionality to help make the most of the client interface. See *Figure 38. Client Bottom Bar*.



No Server Selected

Grid Cell: 10 m

Grid Area: 50000 m

Point: 1 px

Figure 38. Client Bottom Bar

Server Selector

Use **Server Selector** to connect a QORTEX DTC server to the client. When a server is not connected to the client, the **Server Selector** label states **No Server Selected**. When a server is selected, the button label lists the name of the assigned server. See *Connect the Client to the Server* (page 61).

It is important to know that each server represents just one location, and that location includes either a single sensor or a set of multiple calibrated sensors observing the same contiguous geographic space representing the area of interest. When an PTZ camera is added, it is included in the same geographic space, defined by the sensors. See *Adding Sensors* (page 87) and *Adding PTZ Cameras* (page 125).

Visualizer Controls

In the bottom middle of the client interface are the viewing control buttons, toggles, and menus. These adjust the viewing options in the Visualizer window. See *Visualizing Data* (page 114) and *Table 10. Visualizer Buttons, Toggles, and Menus*.

Table 10. Visualizer Buttons, Toggles, and Menus

Name	Image	Description
World Coordinate Toggle		Orient the point cloud to a physical coordinate. For example the ground or a wall.
Sensor Health Toggle		Check the status of each of the sensors. Requires **LIVE** data mode. See *Sensor Health Status Options* (103).
Top/Side View Toggle		Toggle the display between Top View or Side View perspective relative to the sensor. See *Reset View Toggle* (page 115).
Orthographic Projection Toggle		Toggle between on and off mode. ON switches to 3D view. Off switches to 2D view. See *Orthographic Projection Toggle* (page 116).

Name	Image	Description
Visibility Menu		Displays a check list to select whether to include Inclusions, Exclusions, Events, Inclusions, Track ID and/or Counter Lines in the display. See *Visibility: Zones/Lines and Object ID* (page 117).
Point Cloud/Color Picker Toggle		Toggle between multi-sensor viewing options: **No Cloud**, **Color Picker**, **Split**, and **Combined/Fused**. Requires LIVE data mode. See *Point Cloud Display* (page 119).
Grid Cell Menu	Grid Cell: 10 m	Adjust the background grid spacing in the display. Measured in meters. See *Grid Cell* (page 120).
Grid Area Menu	Grid Area: 50000 m	Adjust the size of the monitored area. The measurement applies to all sides of the monitored area. See *Grid Area* (page 120).
Point Size Menu	Point: 1 px	Adjust the size of the points in the display. Measured in pixels. See *Point Size* (page 121).

Settings Button

The **Settings** button opens the **CONFIGURATION** panel on the right side of the client interface. If **Security** mode is enabled, provide credentials. Entering **Configuration** mode requires confirmation, to exit **Monitor** mode and enter **Configuration** mode.

This button is not available until a server is selected.

The **Settings** button differentiates between **Monitor** mode and **Configuration** mode. See *Visualizing Data* (page 114) and *Entering Configuration Mode* (page 84).

Status Button

The **Status** button which provides informational (colorless), warning (yellow), and critical error (**red**) notifications about the licensing and operation of QORTEX DTC.

To use **Status** button:

1. Click the **Status** button to open the **Status** panel.

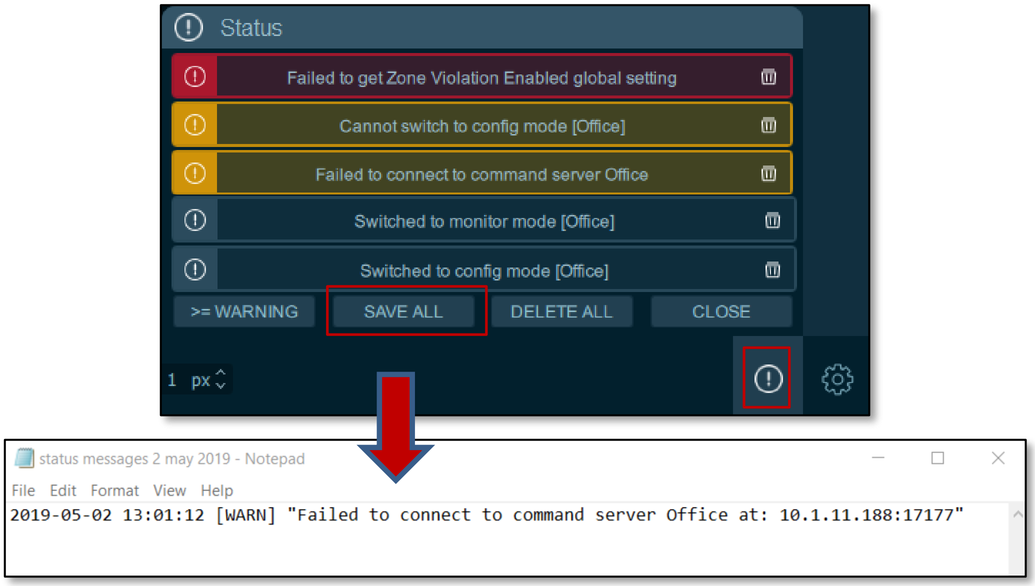
   Critical notifications pop up from the **Status** panel automatically.

2. Click one of the notifications to expand it to show more detail.

3. Click the bottom left button on the **Status** panel to cycle through three types of notifications: >= INFO , >= WARNING , and >= ERROR .

4. Click 🗑 to delete a specific notification or click DELETE ALL to delete the entire list.
5. Click SAVE ALL to save the current list to the filepath and filename specified in the file browser. See *Figure 39. Status Button to Access Notifications (top), Saved Messages (bottom)*.

   The panel displays only the most recent five statuses, deleting old ones as new ones replace them. **Saved Messages** are saved as `.txt` files.
6. Click CLOSE to dismiss the **Status** panel.


Status
>= WARNING
SAVE ALL
DELETE ALL
CLOSE
1 px
status messages 2 may 2019 - Notepad
File Edit Format View Help

Failed to get Zone Violation Enabled global setting

Cannot switch to config mode [Office]

Failed to connect to command server Office

Switched to monitor mode [Office]

Switched to config mode [Office]

```
2019-05-02 13:01:12 [WARN] "Failed to connect to command server Office at: 10.1.11.188:17177"
```

Figure 39. Status Button to Access Notifications (top), Saved Messages (bottom)

9. Entering Configuration Mode

Configuration mode supports only one client at a time in a first come, first served protocol. If a client is in **Configuration** mode, it becomes the controlling client, and no other client can configure the server, create or update zones, counter lines, or view the point cloud until the controlling client has clicked the **EXIT CONFIGURATION** button.

Configuration Mode Options

The **Configuration** panel offers tabs with panels, fields, and options to configure the QORTEX DTC server

- **Server tab** has settings for:
  - **Server Configuration** panel, provides Sensor Setup, Import Calibration, and Location Template. See *Add Single-Sensor* (page 88) or *Add or Update Multiple Sensors at* a Location (page 89) (default).
  - **Security Configuration** panel, to setup security with username and password, enable or disable security mode, change security credentials (username and password), and download user log file.
  - **TCP Publishing** panel, provides options to change the TCP format for Object list, Zone list, and Counter Line list. See *TCP Publishing* (page 92).
  - **Record Settings** panel, provides a toggle to turn compression ON/OFF and identifies the folder where recordings are stored. By default `/home/quanergy/quanergy/qortex/dataset/Recordings` directory in the server is used. This directory is created automatically when the server software is first installed on the server host machine.
- **Zones/Lines tab** has panels for:
  - **Exclusions panel** to set up, enable, or disable Exclusion zones. The zone definition is saved to the `settings.xml` configuration file. See *Establishing Zones and Counter Lines* (page 105).
  - **Events panel** to set up, enable, or disable Event zones. The zone definition is saved to the `settings.xml` configuration file. See *Establishing Zones and Counter Lines* (page 105).
  - **Inclusion panel** to set up, enable, or disable Inclusion zones. The zone definition is saved to the `settings.xml` configuration file. See *Establishing Zones and Counter Lines* (page 105).
  - **Counter lines panel** to setup, enable, or disable Counter lines. The counter line definition is saved to the settings.xml configuration file. See See *Establishing Zones and Counter Lines* (page 105).

- **Sensor tab** provides the **Refresh Tracker** and **Template Parameter** tabs.

  Use **Refresh Tracker tab > Refresh** button to refresh all tracks to clear the noise of lagging artifacts. See *Refresh Tracks* (page 158).

  Expand the **Sensor** tab **> Template Parameter** tabs: **ClusterPipeline, MultiLidarPipeline**, **OutputFilter**, **StaticCloudPipeline**, or **TrackerPipeline** to configure parameters for defining server setting values. See Appendix 3: *Template Parameter Settings* (page 274).

- **PTZ tab** provides the panel for adding PTZ cameras to the configuration. See *Adding PTZ Cameras* (page 125).

- **Rules tab** provides access to create and edit Rules for **PTZ Camera**, **Network Action**, and **Recording**. See *Adding Rules* for Event Zones (page 142).

  - **Add Rule** tab for creating new rules.
  - **Rules** tab lists the existing PTZ Camera and Network Actions rules.
  - **Record** tab provides toggle to turn On/Off Rule-based Recordings. When **ON**, enables Rule based recording. When **OFF**, enables manual recording.

Enter Configuration Mode

Figure 40. Configuration Mode Sequence shows the steps listed below.

1. Connect the QORTEX DTC client to a QORTEX DTC server.

   a. In the bottom of the client, click **Server Selector** to view a list of added servers or **Add New Server** button to add a server. See *Adding Sensors* (page 87).

   b. To confirm connection between sensors and server, click **Server Selector**. Expand the **Servers** panel to view all available servers.

   c. If you have not added any QORTEX DTC servers and if **Security** mode is enabled on the QORTEX DTC server, enter your credentials at the prompt to access the QORTEX DTC server.

   Credentials can be either `username/user password` or `license name/license password`.

2. Disconnect the server from the sensors.

   In the top left corner of the client, click the **Connected** button. Wait until the status changes to **Disconnected**. If the server is already disconnected, no additional action is required.

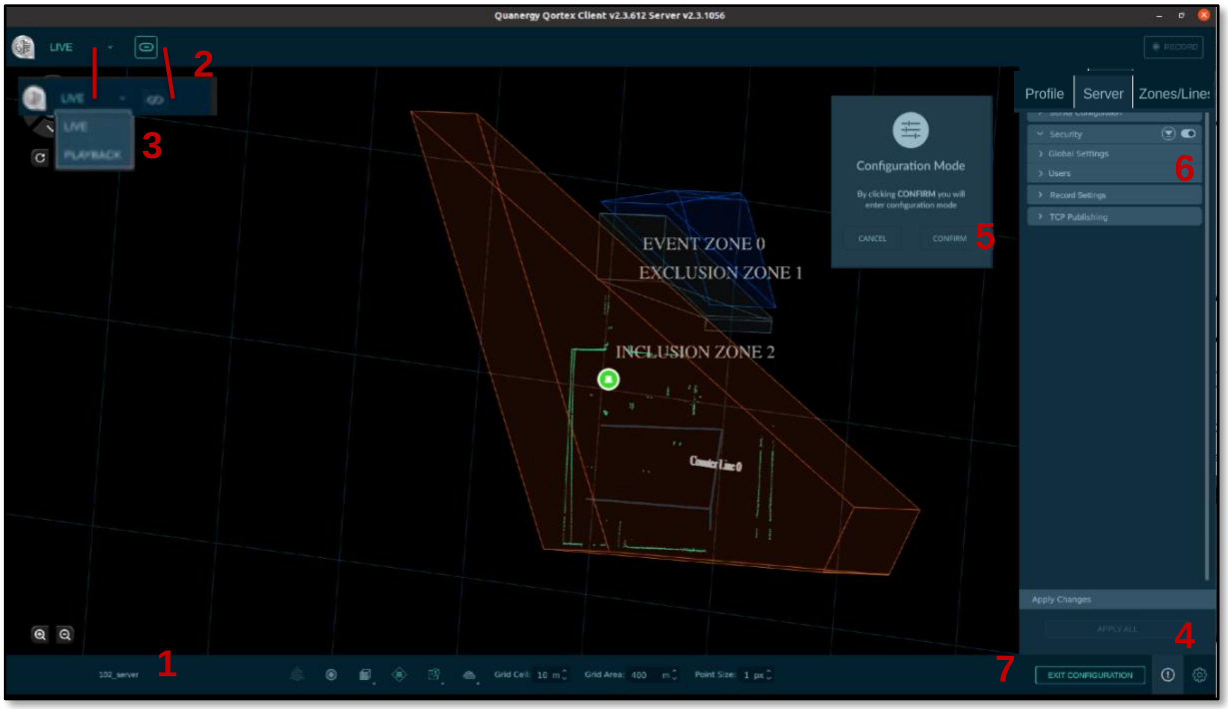
   Changing configuration can be in either LIVE vs PLAYBACK mode.

3. Click the **Settings** button ⚙.

4. Click the **CONFIRM** button.

5. The **Configuration** panel offers tabs with panels, fields, and options to configure the QORTEX DTC server.
   a. Scroll to and click a tab.
   b. Expand panels in a tab.
   c. Enter changes.
   d. Click **Apply Changes**.

      Click **Apply Changes** button remains disabled until a change is made to one of the options.
6. When configuration efforts are complete:
   a. Click **APPLY ALL** to save the configuration changes to the selected Configuration tab.
   b. Click the **Settings** button ⚙. The **Configuration** panel is dismissed, the point cloud continues to be displayed, and other clients continue to remain locked out of **Configuration** mode.
   c. Click the **EXIT CONFIGURATION** button. The client relinquishes control of **Configuration** mode and enters **Monitor** mode, the point cloud is no longer displayed, other clients can enter configuration mode.



By clicking CONFIRM you will enter configuration mode

Figure 40. Configuration Mode Sequence

10. Adding Sensors

Each QORTEX DTC server supports one location at a time. The server can be configured to receive data from a single sensor or from multiple sensors, but not both. Even if a location has multiple sensors, you have the option to add a single sensor to the server.

The button indicating **Connected** or **Disconnected** status in the upper left corner of the interface window is a toggle mechanism that connects the server to, or disconnects the server from, the sensors and PTZ cameras.

Add Sensors Overview

1. Configure the sensors to add to the server.

   Configure each sensor through its web server (Sensor Settings Management). Configure multiple sensors through Q-View. When using multiple sensors, you need to calibrate them using Q-View. Q-View creates the calibration file, `transform_alignment.xml`.

   See the sensor User Guide and the *Q-View User Guide*.

2. Enter **Configuration** mode. See *Entering Configuration Mode* (84).
3. Add sensors to the server through the client. From the **Configuration** panel, select **Server** tab **> Server Configuration** tab.

   Choose to add a single sensor or multiple sensors.

4. To add one sensor, click **Single**.

   Enter sensor IP address and a reference name for the sensor. QORTEX DTC supports both IPv4 and IPv6 IP addresses.

5. To add multiple sensors, click **Multiple**, then click **Browse**.
   a. In Browse window, locate the folder containing the calibration file.
   b. Select the `transform_alignment.xml` file. Click **Open**.
   c. In **Server Configuration > Lidars To Use**: click checkbox for each sensor to add.
6. Select a **Location Template** from the pull-down.
7. Click **OK**. Click **Connect** button.

   The data for each added sensor is displayed as a different color in the Visualizer.

Add Single-Sensor to a Location

Add a single sensor to a server. The sensor defines the location area that the server monitors. See *Figure 41. Single Sensor Configuration Sequence*.

1. Prepare to add a single sensor to the server.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From the **Configuration** panel, select the **Server** tab and click **Single**.
3. In the **Sensor IP Address** field, enter the address of the sensor whose data will be visualized. QORTEX DTC supports both IPv4 and IPv6 IP addresses.
4. In the **Sensor Name** field, enter a name for the sensor, such as a serial number or location.
5. In the **Location Template** section, click the drop-down menu and select whichever of the three templates optimizes for the type of object most likely to be encountered:
   - For an indoor setting, such as an airport baggage claim area, select the `person_tracking_settings.xml` file.
   - For an outdoor setting, such as at a congested stretch of freeway, select the `vehicle_tracking_settings.xml` file.
   - For mixed setting, such as the crosswalk at the entrance to an airport, select the `general_tracking_settings.xml` file.
6. At the bottom of the **Server Configuration** section, click the **OK** button.
7. After the **Server Configuration** panel displays a **Success** ✓ message, the client remembers this single-sensor server configuration (until its `settings` are overwritten by repeating this procedure again) and automatically uses it to visualize LIVE data. See *Visualize Live Data* (page 122).

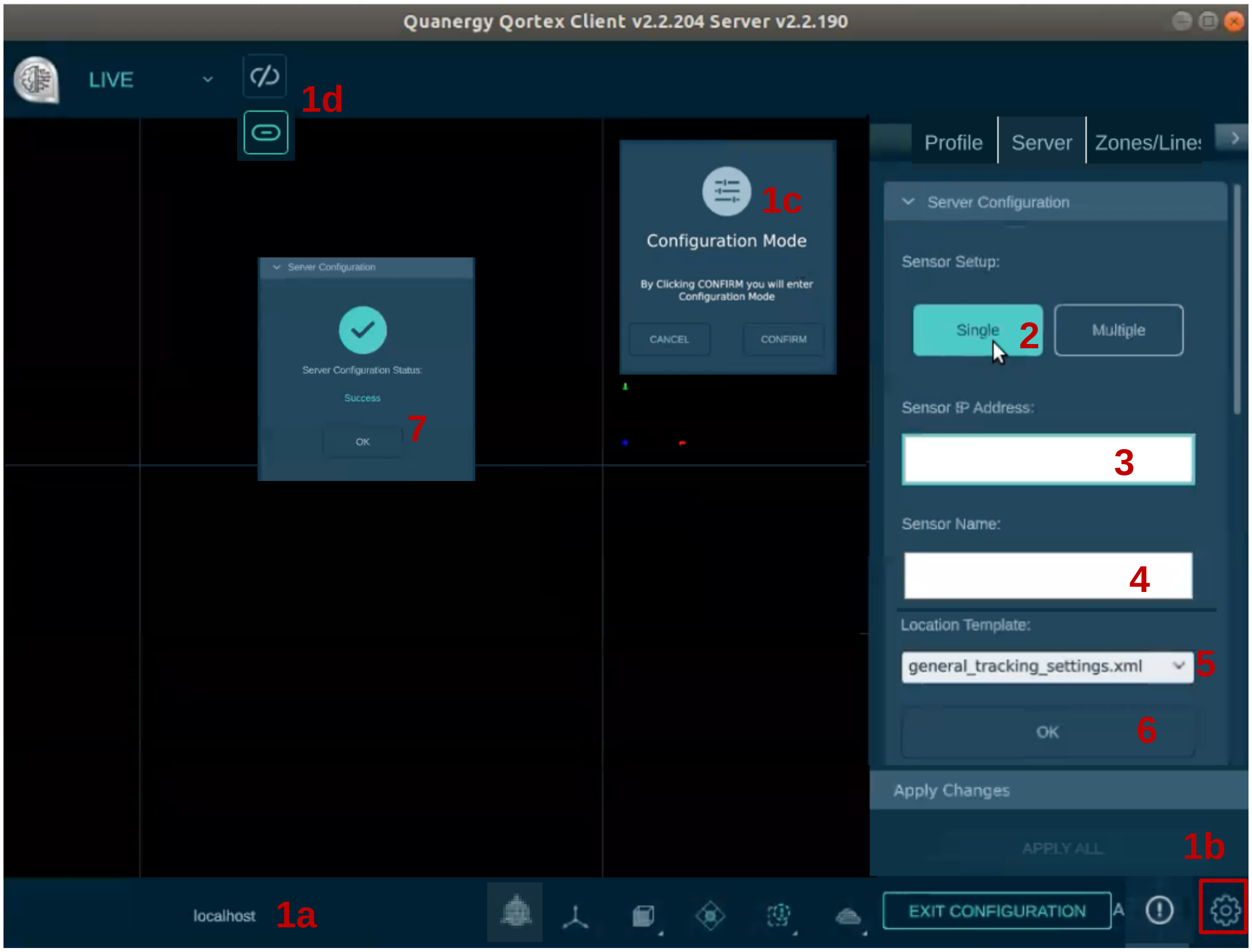


Figure 41. Single Sensor Configuration Sequence

Add or Update Multiple Sensors at a Location

Add multiple sensors to a server. The combined sensors define the location area that the server monitors.

Also, if you re-calibrated a sensor through Q-View or are replacing a sensor connected to the QORTEX DTC server, use the **Sensor** tab.

See *Figure 42. Multiple Sensor Configuration Using transform_alignment.xml File*.

1. Prepare to add multiple sensors to the server.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From the **Configuration** panel, select the **Server** tab > **Multiple**.

3. Browse to import the calibration file created through Q-View. See *LiDAR Sensor Prerequisites* (page 31).

   In the **Import Calibration** section, click the **Browse** button to open a file browser and select the calibration file for your sensors.

   Use Q-View calibration `transform_alignment.xml` file to set up zones from scratch.

4. If you select `transform_alignment.xml` file, select LiDAR sensors.

   These are the sensors whose data will be visualized. In the **Lidars To Use** section, click the checkboxes next to associated LiDAR sensors.

5. Optionally, if you are adding a sensor or recalibrating a sensor, Click the **Keep Location Settings** and/or **Keep LiDAR PTZ Calibration** toggles `ON` or `OFF`.

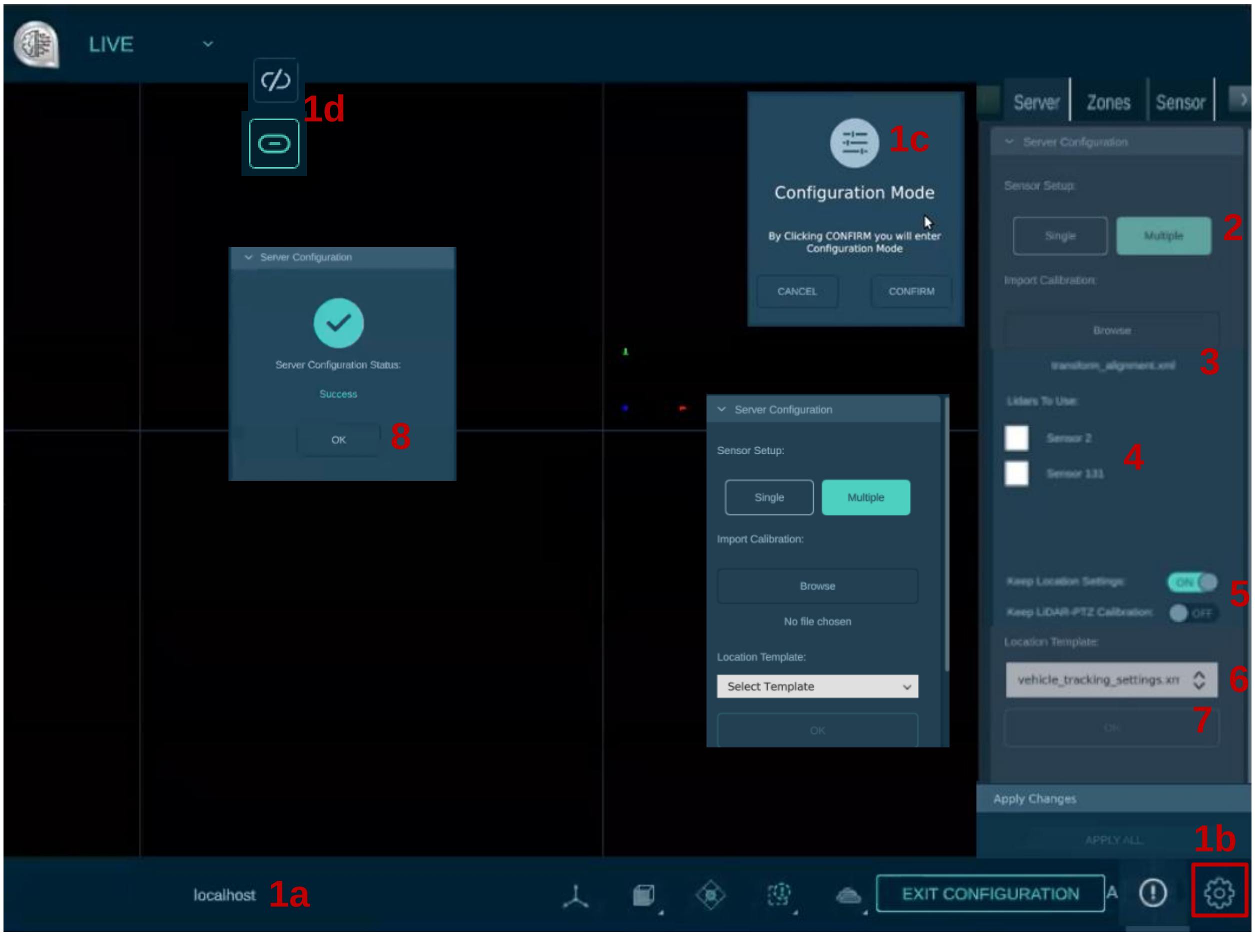
   These toggles allow you to recalibrate the sensors without losing other information such as zones, rules, and algorithm parameters. This way only the Sensor and Transforms are updated.

6. In the **Location Template** section, click the drop-down menu and select the template that optimizes for your expected typical object type. See Appendix 3. *Template Parameter Settings* (page 274).

   - For an indoor setting, such as an airport baggage claim area, select the `person_tracking_settings.xml` file.
   - For an outdoor setting, such as at a congested stretch of freeway, select the `vehicle_tracking_settings.xml` file.
   - For mixed setting, such as the crosswalk at the entrance to an airport, select the `general_tracking_settings.xml` file.

7. At the bottom of the **Server Configuration** section, click **OK** (scroll down, if needed).

8. After the **Server Configuration** panel displays a **Success** ✓ message, the client remembers this multi-sensor server configuration (until its `settings` are overwritten by repeating this procedure) and automatically uses it to visualize LIVE data. See *Visualize Live Data* (page 122).



By Clicking CONFIRM you will enter Configuration Mode

Server Configuration Status: Success

Server Configuration

Sensor Setup: Single [✓ Multiple]

Import Calibration: Browse

No file chosen

Location Template: Select Template

OK

Server Configuration

Sensor Setup: Single [✓ Multiple]

Import Calibration: Browse

transform_alignment.xml

Lidars To Use:

□ Sensor 2

□ Sensor 131

Keep Location Settings: ON

Keep LIDAR-PTZ Calibration: OFF

Location Template: vehicle_tracking_settings.xn

OK

Figure 42. Multiple Sensor Configuration Using transform_alignment.xml File

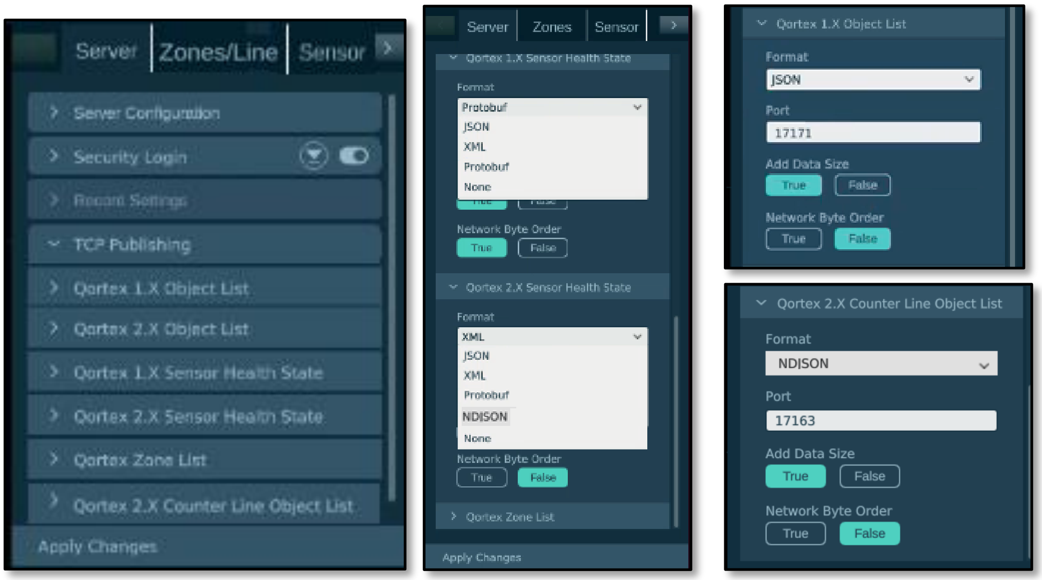
11. QORTEX DTC and Sensor Configuration Options

Additional configuration `settings` include:

- *TCP Publishing Format* (page 92)
- *Adjust Sensor Settings* (page 96)
- *Sensor Health Status Options* (103)

TCP Publishing Format

As of QORTEX DTC 2.3 you have the option to change the TCP format. This includes port, data size, and network byte order through QORTEX DTC. See *Figure 43. TCP Publishing Settings: Object, Sensor Health, Zone, Counter Line*.



Qortex 1.X Object List

Format: JSON

Port: 17171

Add Data Size: True

Network Byte Order: False

Qortex 2.X Counter Line Object List

Format: NDJSON

Port: 17163

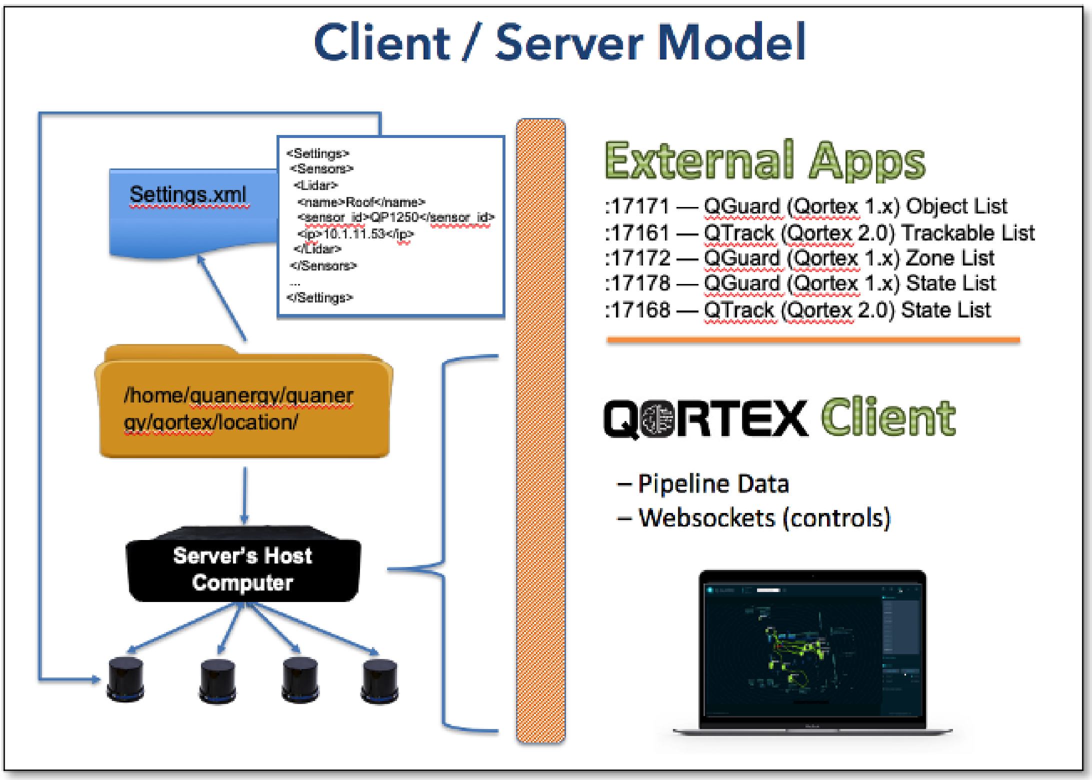
Add Data Size: True

Network Byte Order: False

Figure 43. TCP Publishing Settings: Object, Sensor Health, Zone, Counter Line

Server TCP Ports to Monitor

Quanergy encourages the use of third-party applications that directly consume the server output, which can then be reappropriated in other ways. LiDAR sensors are identified by IP address in the `settings.xml` file connected to the server host computer. See *Figure 44. External Applications Access Output on Server Host Computer*. The external applications can monitor specific TCP ports on the server and access the data published there directly.



Client / Server Model

Settings.xml

```
<Settings>
 <Sensors>
  <Lidar>
   <name>Roof</name>
   <sensor_id>QP1250</sensor_id>
   <ip>10.1.11.53</ip>
  </Lidar>
 </Sensors>
 ...
</Settings>
```

/home/quanergy/quanergy/qortex/location/

Server's Host Computer

External Apps

:17171 — QGuard (Qortex 1.x) Object List
:17161 — QTrack (Qortex 2.0) Trackable List
:17172 — QGuard (Qortex 1.x) Zone List
:17178 — QGuard (Qortex 1.x) State List
:17168 — QTrack (Qortex 2.0) State List

QORTEX Client

– Pipeline Data
– Websockets (controls)

Figure 44. External Applications Access Output on Server Host Computer

To aid the user, Quanergy created a Port Monitoring Tool and identified a publication-confirmation utility.

Quanergy QORTEX DTC Application Programming Interface (API) is an Ethernet interface available on the local network where the server is running. This API delivers the needed data to the TCP port and in Protobuf, JSON, or XML format specified in the `settings.xml` file. These ports are configurable, but a typical scenario is outlined below:

- Port 17171 publishes QORTEX DTC 1.x object list in Protobuf, JSON, NDJSON, or XML.
- Port 17161 publishes QORTEX DTC 2.x trackable list in Protobuf, JSON, NDJSON, or XML.
- Port 17172 publishes QORTEX DTC 1.x and QORTEX DTC 2.x zone list data in Protobuf, JSON, NDJSON, or XML.
- Port 17178 publishes QORTEX DTC 1.x state list data in Protobuf, JSON, NDJSON, or XML.
- Port 17168 publishes QORTEX DTC 2.x state list in Protobuf, JSON, NDJSON, or XML. Qortex DTC 2.3 or greater includes sensor masking detection

Edit TCP Port Publishing Settings

1. Prepare to edit the TCP port publishing settings on the server.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From the **Configuration** panel, select the **Server** tab and expand the **TCP Publishing** tab. Click the **v** arrow on the tab.

3. Expand the TCP publishing option to edit. The choices are separated by Qortex DTC version in some cases:

   - QORTEX 1.x Object List
   - QORTEX 2.x Object List
   - QORTEX 1.x Sensor Health State
   - QORTEX 2.x Sensor Health State
   - QORTEX Zone List
   - QORTEX 2.x Counter Line Object List

4. For each of the TCP option, the configurable items are:

   - **Format**— Default is None, which means QORTEX DTC server does not publish data. To enable publishing data for a particular list or state, select a file format: JSON, NDJSON, XML, and Protobuf.

   - **Port**—The port number assigned to the TCP item that is listening for this content: object lists, sensor health states, or zone lists.

   - **Add Data Size**—**True** adds the data packet size to the output, **False** does not include the data size. False is the default.

   - **Network Byte Order**— **True** transmits the data in Big Endian order. **False** transmits the data in Little Endian order. False is the default. This option applies when PTZ cameras are connected to the QORTEX DTC server.

These changes are applied to the settings.xml file.

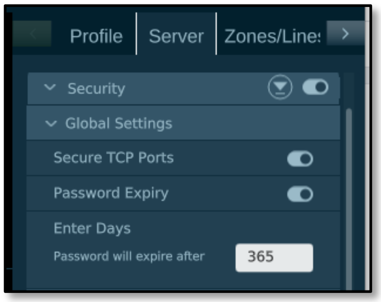
```
<TCPPublishers>
  <!-- TCPPublishers section lists all publishing APIs, each publisher section has:
   - Fixed Name which is a string referenced in code to identify specific API, it
     could be :
          QORTEX1_OBJECT_LIST,
          QORTEX1_ZONE_LIST, ...
    - Publishing Format which could be 'json', 'xml', 'protobuf', ‘NDJson’ or
'none';
      Optional parameter in case of default is 'none'
    - Publishing TCP Port, Optional parameter in case of default uses a hard-coded
      default value for that
  -->
```

```
        <Publisher>
            <Name>QTRACK_COUNTER_LINE_OBJECT_LIST</Name>
            <Format>json</Format>
            <Port>17163</Port>
            <AddDataSize>false</AddDataSize>
            <NetworkByteOrder>false</NetworkByteOrder>
        </Publisher>
        <Publisher>
            <Name>QORTEX1_OBJECT_LIST</Name>
            <Format>none</Format>
            <Port>17171</Port>
            <AddDataSize>false</AddDataSize>
            <NetworkByteOrder>false</NetworkByteOrder>
        </Publisher>
        <Publisher>
            <Name>QORTEX1_ZONE_LIST</Name>
            <Format>none</Format>
            <Port>17172</Port>
            <AddDataSize>false</AddDataSize>
            <NetworkByteOrder>false</NetworkByteOrder>
        </Publisher>
        <Publisher>
            <Name>QTRACK_OBJECT_LIST</Name>
            <Format>none</Format>
            <Port>17161</Port>
            <AddDataSize>false</AddDataSize>
            <NetworkByteOrder>true</NetworkByteOrder>
        </Publisher>
        <Publisher>
            <Name>SENSOR_HEALTH_STATE</Name>
            <Port>17168</Port>
            <Format>none</Format>
            <AddDataSize>false</AddDataSize>
            <NetworkByteOrder>false</NetworkByteOrder>
        </Publisher>
        <Publisher>
            <Name>QORTEX1_STATE</Name>
            <Port>17178</Port>
            <Format>none</Format>
            <AddDataSize>false</AddDataSize>
            <NetworkByteOrder>false</NetworkByteOrder>
        </Publisher>
        <!-- Data intended for Qortex Client -->
        <Publisher>
            <Name>CONFIGURATION_CLIENT_DATA</Name>
            <Port>17175</Port>
        </Publisher>
        <Publisher>
            <Name>MONITOR_CLIENT_DATA</Name>
            <Port>17176</Port>
        </Publisher>
    </TCPPublishers>
```

Figure 45. Sample TCPPublishers portion of settings.xml

Secure TCP Port Data

1. Login as a secure user. See *Create New Users* (page 69.)
2. From the **Configuration** panel, select the **Sensor** tab and expand **Advanced Settings > Global Settings**.
3. Enable **Secure TCP Ports**. Click the toggle to **ON** .



Secure TCP Ports

Password Expiry

Enter Days

Password will expire after 365

Figure 46. Global Settings > Secure TCP Ports

Adjust Sensor Settings

Use the Sensor tab to adjust the sensor settings. See *Figure 47. Basic and Advanced Sensor Settings in Sensor Tab*.

> **Note:** Though you can view the settings, the functionality is only available for a LIVE location.

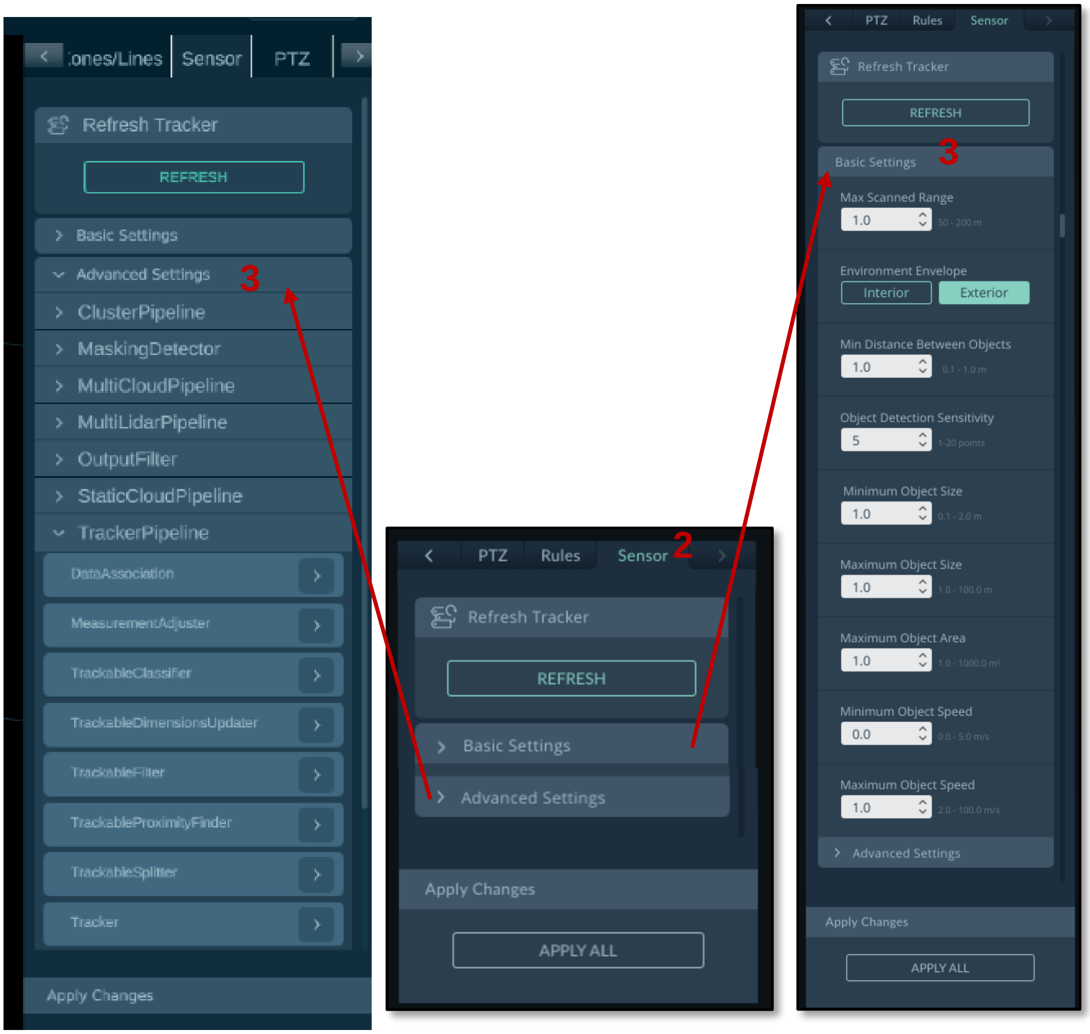
1. Prepare to adjust sensor settings.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).
2. From the **Configuration** panel, select the **Sensor** tab and expand **Basic Settings** or **Advanced Settings**.
3. Adjust the optimized values in a settings template (person, vehicle, or general). Expand the relevant panel and edit the fields listed.

- **Basic Settings**, are designed for general users. They are a subset of the Advanced Settings. Editing Basic Settings is recommended for most use cases. See *Table 11. Basic Sensor Settings Descriptions*.
- **Advanced Settings**, the options include: ClusterPipeline, ClusterFilter, ClusterSplitter, MultiLidarPipeline, OutputFilter, StaticCloudPipeline, and TrackerPipeline. See Appendix 3: *Template Parameter Settings* (page 274).

  Editing Advanced Settings is recommended for advanced users who have a deeper knowledge of the QORTEX DTC parameter settings.

4. Click **APPLY ALL** at the bottom of the Sensor tab. Click **REFRESH** to return all values to default.



Max Scanned Range 1.0 50 - 200 m

Environment Envelope ○ Interior ● Exterior

Min Distance Between Objects 1.0 0.1 - 1.0 m

Object Detection Sensitivity 5 1-20 points

Minimum Object Size 1.0 0.1 - 2.0 m

Maximum Object Size 1.0 1.0 - 100.0 m

Maximum Object Area 1.0 1.0 - 1000.0 m²

Minimum Object Speed 0.0 0.0 - 5.0 m/s

Maximum Object Speed 1.0 2.0 - 100.0 m/s

Figure 47. Basic and Advanced Sensor Settings in Sensor Tab

Table 11. Basic Sensor Settings Descriptions

Field	Range	Description	Advanced Setting Parallel
Max Scanned Range	50-200 m	The maximum distance the sensor can distinguish between background (static points) and foreground (moving points). Anything that exists beyond the setting is classified as background and is not tracked as an object.	`BackgroundFilter > maxBinDistance`
Environment Envelope	Interior or Exterior	Set to Exterior when there is open space beyond the maximum LiDAR range. Set to `Interior` when the environment is within (less than) the maximum LiDAR range.	`BackgroundFilter > useNanAsUnobstructed`
Min Distance Between Objects	0.1-1.0 m	Objects detected within this distance are merged together.	`ClusterPipeline > CCCClusterer > cellSize`
Object Detection Sensitivity	1-20 points	Set the minimum number of points required to define a cluster as an object. Setting low values increases sensitivity and can return false positives. Setting high values increases the possibility of missing objects.	`ClusterPipeline > ClusterFilter > minNumClusterPoint`
Minimum Object Size	0.1-2.0 m	Objects smaller than this value are filtered out.	`OutputFilter > minSize`
Maximum Object Size	1.0-100.0 m	Objects larger than this value are filtered out.	`OutputFilter > maxSize`
Maximum Object Area	1.0-1000.0 m^2	Objects square area (length * width) is larger than this are filtered out.	`OutpuFilter > maxArea`
Minimum Object Speed	0.0-5.0 m/s	Objects traveling at speeds (m/s) slower than this are filtered out.	`OutputFilter > minSpeed`
Maximum Object Speed	2.0-100.0 m/s	Objects traveling at speeds (m/s) faster than this are filtered out.	`OutputFilter > maxSpeed`

Advanced Setting: Anti-Masking Detector Settings

Anti-Masking detection sends an alert when the point cloud is compromised. For example, when the sensor cap is blocked. This feature is set to **False** (off) by default. See *Figure 48. Anti-Masking Detector Panel*.

1. From the **Configuration** panel, select the **Sensor** tab > expand **Advanced Settings** > expand **MaskingDetector**. Click the **v** arrow on the tab.

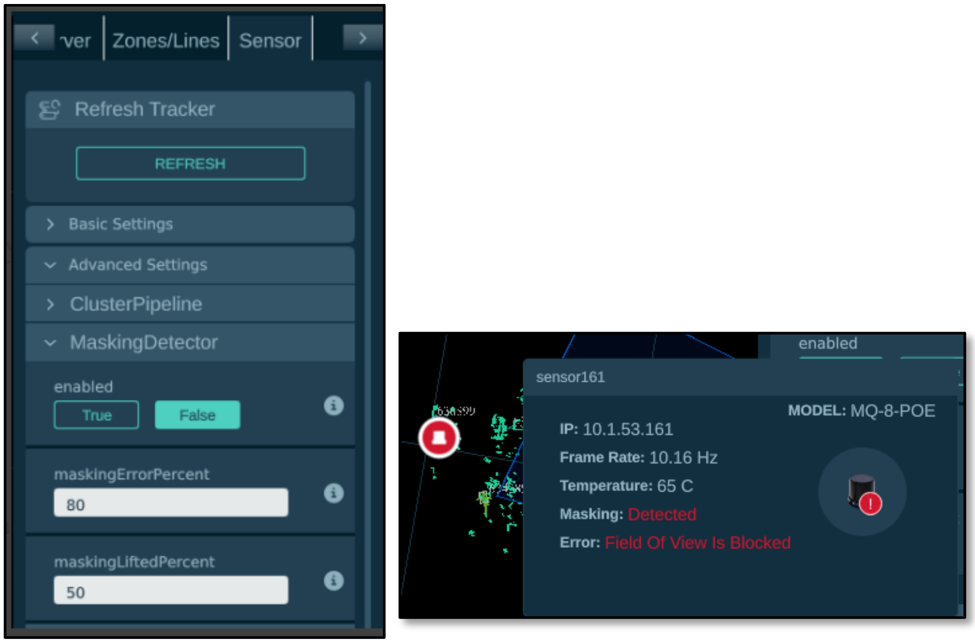
2. Turn on **Masking Detector**. Set **enable** to **TRUE**.

3. Set masking notification threshold.

- **Masking ErrorPercent** – The percentage of point in the point cloud with no detected return must be greater than this threshold. When threshold reaches or exceed this value, notification is sent that the sensor is masked, masking is detected. Valid range 1.0 to 100%. Default is 80%.
- **Masking ListedPercent** – The percentage of points in the point cloud with no detected return must be less than or fall below this threshold. When the threshold reaches or drops below this value, notification is sent that the sensor is no longer masked, masking is no longer detected. Valid range 1.0 to 100%. Default is 50%.

To prevent bouncing, set these values to so they have greater than a 10% difference.

4. Click **Apply Changes**.



enabled: False

maskingErrorPercent: 80

maskingLiftedPercent: 50

sensor161

MODEL: MQ-8-POE

IP: 10.1.53.161

Frame Rate: 10.16 Hz

Temperature: 65 C

Masking: Detected

Error: Field Of View Is Blocked

Figure 48. Anti-Masking Detector Panel and Masking Detected Sample

Advanced Setting: Movement Threshold Filter

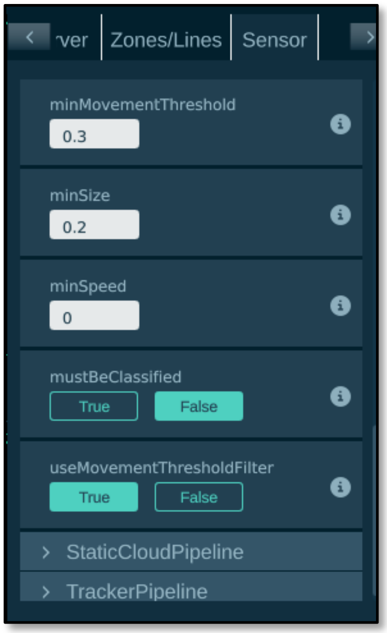
Use the Movement Threshold to identify static objects, such as grass and foliage. The Movement Threshold is an object filter that keeps track of where each object was first seen and calculates the distance with its current position. When an object does not meet the threshold values, it is considered static and not assigned an object ID.

1. From the **Configuration** panel, select the **Sensor** tab > expand **Advanced Settings** > **Output Filter** > **Movement Threshold**.

2. Set **MovementThreshold** values to define and enable the threshold filter.

- **minMovementThreshold** – distance of detected movement measured in meters.
- **minSize –** Objects smaller than this value are filtered out.
- **minSpeed –** Objects traveling at speeds (m/s) slower than this are filtered out.
- **mustBeClassified –** Object classified as “unknown” are filtered out.
- **useMovementThresholdFilter** – to enable Movement Threshold, set to **True**.

3. Click **Apply Changes**.



minMovementThreshold: 0.3

minSize: 0.2

minSpeed: 0

mustBeClassified: True [✓ False]

useMovementThresholdFilter: [✓ True] False

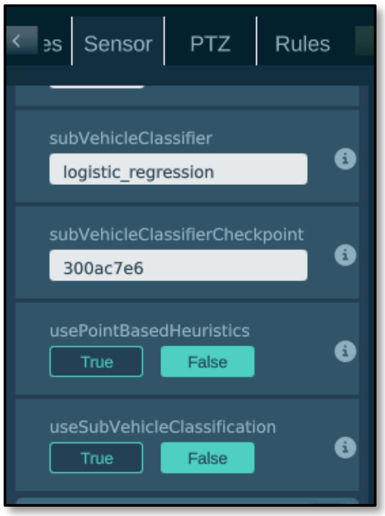
Figure 49. Minimum Threshold Filter Panel

Advanced Setting: SubVehicle Classification Method

QORTEX DTC 2.3 provides an option to use a machine learning based subvehicle classifier rather than the logic classifier of earlier QORTEX DTC versions. The ML model and the logic classifier both differentiate between the same three vehicle subclasses:

- Class 1 – Two-wheeler (motorcycle, bicycles, scooters)
- Class 2 and 3 – Passenger vehicles (passenger cars, trucks, and vans)
- Class 4 and above – Commercial vehicles (busses, commercial trucks)

1. From the **Configuration** panel, select the **Sensor** tab > expand **Advanced Settings** > **Output Filter** > **SubVehicle Classification**.
2. Set the subVehibcle classifier mode. In the **subVehicleClassifier** field, select
    - `logic` – this uses the logic classifier
    - `logic_regression` – this uses the ML classifier
3. If using ML classifier, set learning script. In the **subVehicleClassifierCheckpoint** field, select from the options: `528ef191`, `ff902fb0`, `300ac7e6`.
4. If in a cluster configuration, set to use heuristics. In the **usePointBasedHeuristics** field select True.
5. Enable subVehicle classifier. This setting and a license for subVehicles is required to enable subVehicle classification. In the **useSubVehicleClassification** field, select **True**.



subVehicleClassifier

logistic_regression

subVehicleClassifierCheckpoint

300ac7e6

usePointBasedHeuristics

True [✓ False]

useSubVehicleClassification

True [✓ False]

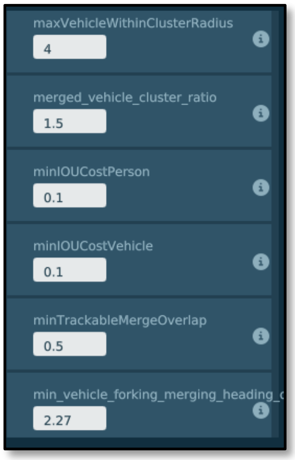
Figure 50. SubVehicle Classifier Panel

Advanced Setting: Object Forking and Splitting Parameters

Sometimes as objects approach each other, then move apart, they are assigned different object IDs. To keep consistent object IDs, adjust the parameters: `min_vehicle_forking_merging_heading_diff` and `merged_vehicle_cluster_ratio`.

Decreasing `merged_vehicle_cluster_ratio` and increasing `min_vehicle_forking_mergins_heading_diff` can cause false positives.

1. From the **Configuration** panel, select the **Sensor** tab > expand **Advanced Settings** > **Output Filter**.
2. Adjust the values in the fields.
   - `merged_vehicle_cluster_ratio` – set value to greater than 2.27
   - `min_vehicle_forking_merging_heading_diff` – set value to less than 1.5
3. Apply **Changes**.



maxVehicleWithinClusterRadius

4

merged_vehicle_cluster_ratio

1.5

minIOUCostPerson

0.1

minIOUCostVehicle

0.1

minTrackableMergeOverlap

0.5

min_vehicle_forking_merging_heading_c

2.27

Figure 51. Merged and Forking Object Panel

Advanced Setting: Stabilize Stationary Vehicles Settings

Sometimes the visualization of objects sitting still for a time, wobble. To reduce this bouncing of stationary object, enable `StabiitizeStationaryVehicles` and adjust the parameters: `velocityThreshold` and `windowSize`.

1. From the **Configuration** panel, select the **Sensor** tab > expand **Advanced Settings** > **MeasurementAdjuster > StabilizeStationaryVehicles**.
2. Lower the values in the fields when vehicles are showing as stationary when they are not.
   - `velocityThreshold` – example value 1.5
   - `min_vehicle_forking_merging_heading_diff` – example value 10
3. Enable the StabilizeStationaryVehicles feature. Select **True**.
4. Apply **Changes**.

Advanced Setting: Fuse Point Cloud Pipeline Settings

When there is a large number of objects, and you want to reduce processing time, enable `MultiCoudPipeline`. This feature uses a Deep Learning based perception pipeline that connects to raw data pipeline and merges the point cloud from multiple lidars into one merged point cloud.

1. From the **Configuration** panel, select the **Sensor** tab > expand **Advanced Settings** > **MultiCloudPipeline**.
2. Enable the `MultiCloudPipeline` feature. Select **True**.
3. If needed, specify the name of the multiCloud to enable. In the configuration file, the parameter is `<name>`*`multiCloud`*`</name>`.
4. Apply **Changes**.

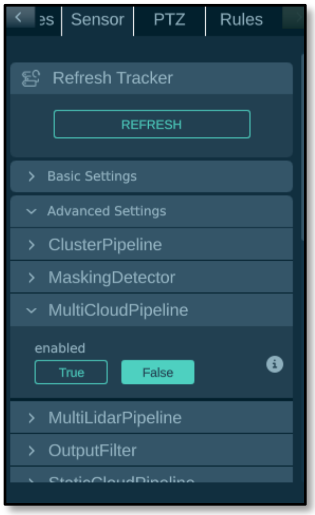


Figure 52. Fuse MultiCloud Pipeline Panel

Sensor Health Status Options

When the QORTEX DTC client is displaying **LIVE** data, use **Sensor Health** to check the status of each sensor. See *Figure 53. Sensor Health: Axes (1), Disconnected (2), Healthy (3), Malfunctioning (4)*.

Enable sensor health mode

Click the **Sensor health** toggle to display or hide the health and diagnostics data for all sensors and PTZ cameras added to the QORTEX DTC server. In the Visualizer window, the colored axes change to become **green** healthy , **red** malfunctioning , or grey disconnected sensor icon.

Sensor health for the PTZ camera relays information about the PTZ camera connectivity states, including Disconnected, Connected, IP not reachable, and Other error.

Display an informational tool tip

Hover over the **Sensor health** button to see the list of options.

View the diagnostics panel

1. Hover over one of the **green** (or **red**) sensor icon to display a tooltip with instructions and the sensor or PTZ camera name.
2. Click one of the **green** (or **red**) sensor icon to select a sensor or PTZ camera.
3. Review the selected sensor health data as it appears in a single panel on the Visualizer window.
4. Click a different sensor icon to see another sensor data.

Dismiss the diagnostics panel

Click in the Visualizer window, not on the sensor icon.

Disable sensor health mode

Click the **Sensor health** button . In the Visualizer window, the now inactive sensor icons return to displaying as 3D-colored axes, and the **Sensor health** button resumes its disabled appearance.

Sensor Health Diagnostics Panel

The **Diagnostics** panel reports these important data points pertaining to the selected sensor:

- **Sensor name**

  Typically, this is the sensor serial number. See *View PTZ Camera Details* (page 140).

- **Sensor model and image**.

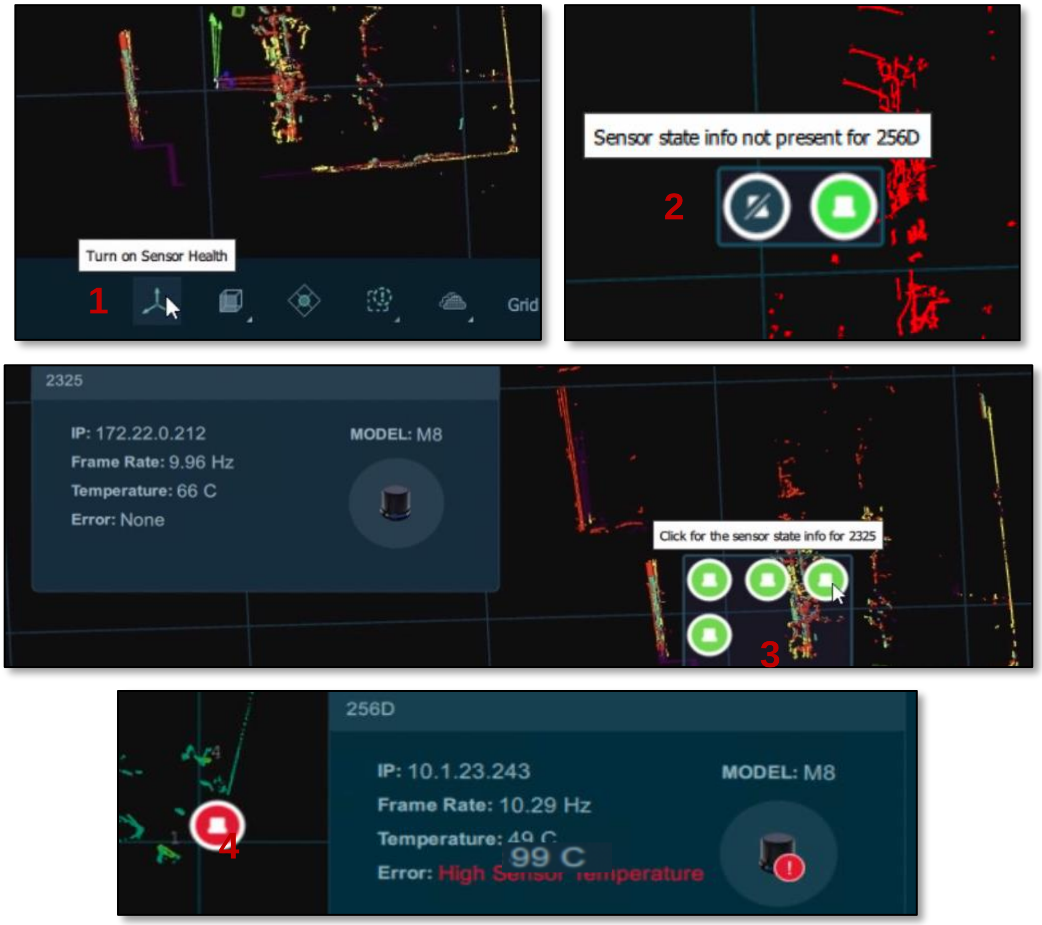
- **IP address**, such as 10.1.24.137. QORTEX DTC supports both IPv4 and IPv6 IP addresses.
- **Frame rate** is the rotational speed of the turret (stated as the number of cycles per second in hertz). This is adjustable through the Sensor Settings Management application (webserver) built into the sensor. See *Edit Settings* in your sensor-specific user guide. Request the sensor user guide from support@quanergy.com.
- **Temperature** reported in degrees Celsius.

  Error reported as None or High Sensor Temperature, for example. The error message is in **red**, and the malfunctioning sensor icon appears in **red** in the client interface for identification.

  If the M8 sensor Rev D4P and higher returns an error, a message printed in red describes the problem. If the M8 sensor Rev D4 and lower returns an error, it cannot report it in the **Diagnostics** panel.

  Table 18. System Status and Error Messages lists the possible sensor state messages.

  For possible causes and solutions for the reported error, see *Troubleshooting Issues* in your sensor-specific user guide. Request the sensor user guide from support@quanergy.com.



Sensor state info not present for 256D

2325

IP: 172.22.0.212

Frame Rate: 9.96 Hz

Temperature: 66 C

Error: None

MODEL: M8

Click for the sensor state info for 2325

256D

IP: 10.1.23.243

Frame Rate: 10.29 Hz

Temperature: 49 C 99 C

Error: High Sensor Temperature

MODEL: M8

Figure 53. Sensor Health: Axes (1), Disconnected (2), Healthy (3), Malfunctioning (4)

12. Establishing Zones and Counter Lines

Typically, a site area of interest is not uniformly important. Secured sites have areas of minimal and maximal importance, and QORTEX DTC enables treating each of those spaces with appropriate consideration. See *Figure 54. Event, Exclusion, Inclusion Zones, and Counter Lines in Visualization Pane*.

- **Event zones** are special areas of importance for security and smart cities and spaces with interests in monitoring pedestrian flow, valuable resources, or sensitive equipment. QORTEX DTC Rules apply to Event zones only. Third party software can ingest the zone list output and define triggered actions — highlighting tracked objects or turning lights or alarms on and off through HTTP GET network command — to target suspicious activities before actual problems occur.
- **Exclusion zones** are visually noisy areas (flapping flags or swaying branches, for example) that pose zero threat and therefore no interest. These are areas that would be wasting bandwidth for detecting, visualizing, tracking, classifying, or recording the moving objects. Exclusion zones can be used in conjunction with other zones to exclude an area within another zone.
- **Inclusion zones** limit the QORTEX DTC processing area. Everything outside of the Inclusion is ignored. If 0 Inclusions zones are drawn, the QORTEX DTC server processes the entire area covered by the sensors except for Exclusion zone areas. If 1 or more Inclusion zones are drawn, the QORTEX DTC server only processes the point cloud in the Inclusion zones. Event zones and Exclusion zones can overlap Inclusion zones.
- **Counter Lines** track movement of objects across counter lines. Set direction for the count left/right, in/out. To count the number of people crossing into or out of an area, add a counter line. Set counter lines in doorways or metal detectors. This is useful for counting occupancy in bathrooms, conferences room, pedestrian occupancy crossing streets, or entering buildings such as stores and restaurants from the street. Counter Lines can overlap all other zones.

Edit zones and counter lines through the **Zone/Lines** tab in the **Configuration** panel. Zone and Counter Line definitions are stored in the `settings.xml` file. The `settings.xml` file is produced, managed, and updated based on actions performed in the **Configuration** panel.

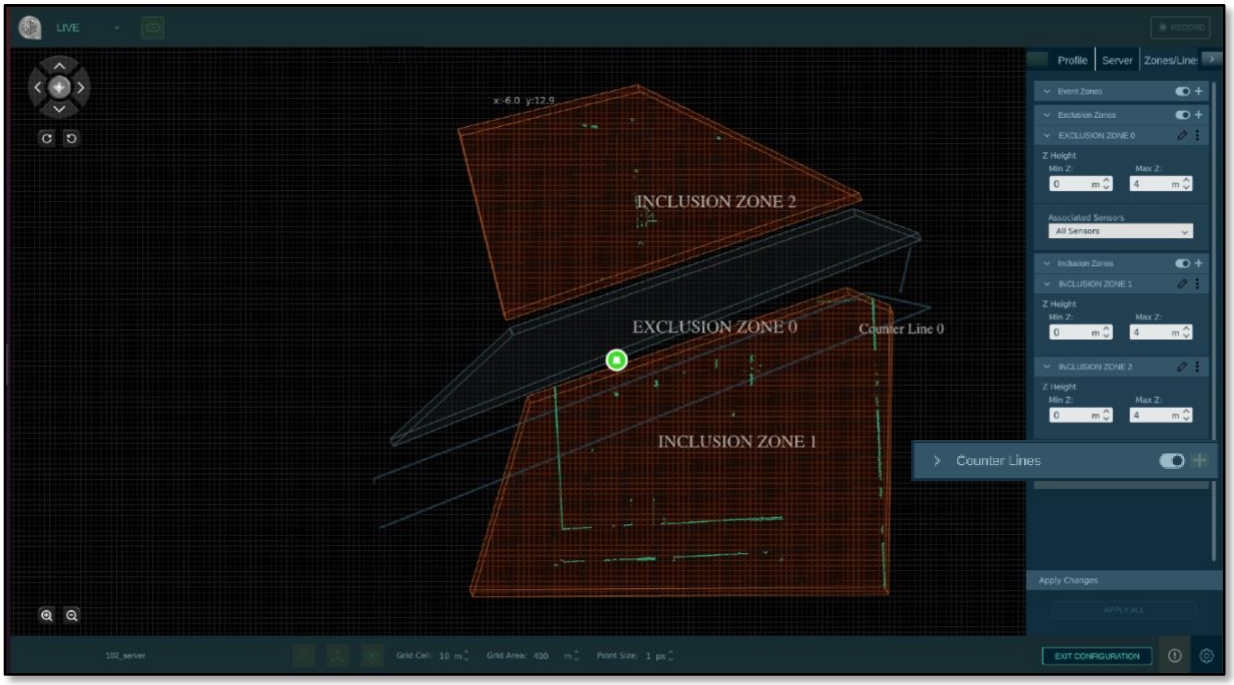


Figure 54. Event, Exclusion, Inclusion Zones, and Counter Lines in Visualization Pane

Create Zone/Counter Line Overview

1. *Prepare to Define Zone/Lines* (page 107).

   From the QORTEX DTC client, enter **Configuration** mode.

2. *Add a Zone/Line* (page 107).

   Select the tab for the zone/counter line to draw, either **Event**, **Exclusion**, **Inclusion**, or **Counter Line**.

   Click in the **Visualizer** to add connected points of the zone. Counter Lines do not need to be closed.

3. *Adjust Vertical Distances* (page 111).

   Define the height (Z-axis) of the zone, up and down relative to the position of the sensors.

Prepare to Define Zone/Lines

After a server is selected and set up with a single-sensor or multi-sensors, define a zone/counter line. See *Figure 55. Zone/Line Definition Initial Steps*.

1. From the **Configuration** panel, scroll to and click the **Zones/Lines** tab.
2. Expand the **Event**, **Exclusion, Inclusion**, or **Counter Line** panels. Click the expand panel › icon
3. Continue to *Add a Zone/Line* (page 107)

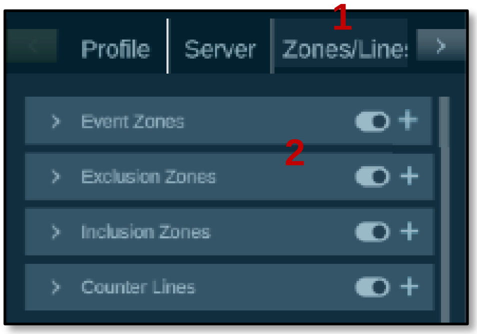


Figure 55. Zone/Line Definition Initial Steps

Add a Zone/Line

Add one or more Zones/Lines to the sensor defined area. See *Figure 56. Add Zone/Lines Buttons*.

1. From the **Configuration** panel, select **Zones/Lines** tab **> Event**, **Exclusion**, **Inclusion**, or **Counter Line** panel click the **Add (+)** button. Each new instance of a zone/counter line panel object increments the default name number.

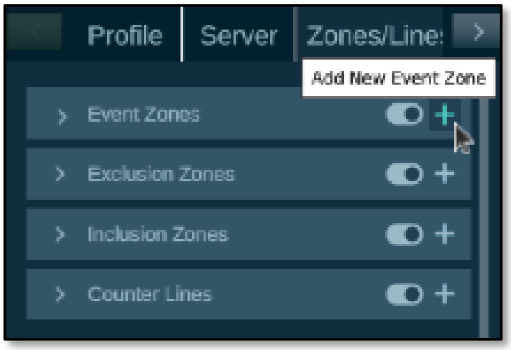


Figure 56. Add Zone/Lines Buttons

2. **Draw the zone/counter line.**

   See *Figure 57. Three Zone Types and a Counter Line*.

   a. Click in the Visualizer window.

   b. Move and click a second point.

   c. Continue to move and click until the polygon is complete.

   Right-click to deselect a previously placed point.

d. When the shape is defined by at least three points, complete the zone/counter line. Choose a method:

- Select Save/Exit from the right panel.
- Left double-click the zone in the Visualizer window.
- Click the green **Close** button in the Visualizer window.

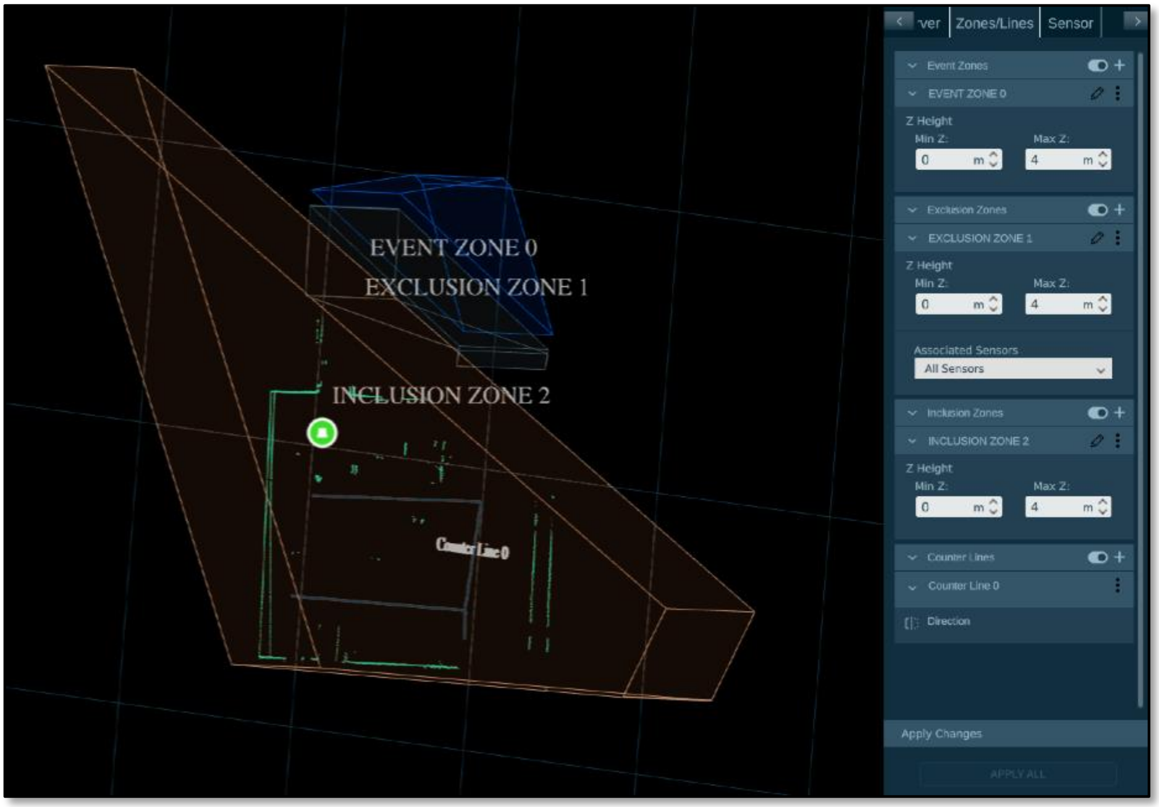
3. For Counter Lines, toggle the direction that define ‘In’ and ‘Out’ across the Counter Line. Click the **Direction** icon to switch the direction.

4. Click **Apply Changes** to save the zone/counter line and update the configuration file on the QORTEX DTC server. This ensures the zone/counter line shows up persistently in each session.

5. Click **APPLY ALL**, when you are finished creating zones/counter lines.

6. Toggle the zone/counter line to ON ●.

7. For Zones, continue to the *Adjust Vertical Distances of the Zone* (page 111).



Z Height
Min Z: 0 m
Max Z: 4 m

Z Height
Min Z: 0 m
Max Z: 4 m

Associated Sensors: All Sensors

Z Height
Min Z: 0 m
Max Z: 4 m

Figure 57. Three Zone Types and a Counter Line

Edit a Zone/Line

All zone types and counter lines can be edited.

To change the shape of an existing zone/counter line:

1. Open the zone/counter line editor:

   From the Visualizer, in **Configuration** mode, use your mouse and keyboard to manipulate the zone/counter line, and enter **Edit** mode. Choose a method:

   - Left click in the zone/counter line.
   - From the **Configuration > Zones/Lines** tab, click the edit (pencil) ✎ icon.

   The vertices of the zone/counter line display as points. See *Figure 58. Zone/Line Edit Icon and Editing Vertex Points*.

2. Apply changes to the zone/line:

   - **Move an entire zone/line:** Single left-click and hold, then move the mouse to the new position and left-click. QORTEX DTC moves the entire zone/counter line.

   - **Edit a zone/line vertex point:** Double right-click on an individual vertex point to select it. Move the mouse to the new location and single left-click.

   - **Add a zone/line vertex point:** Single left-click at new vertex point location. Qortex DTC automatically adds the point into the zone/counter line, relative to the two nearest points in the zone/line.

   - **Remove a selected vertex point:** Double right-click on an individual vertex point to select it. Single right-click.

3. Close zone/counter line editing: Double left-click.

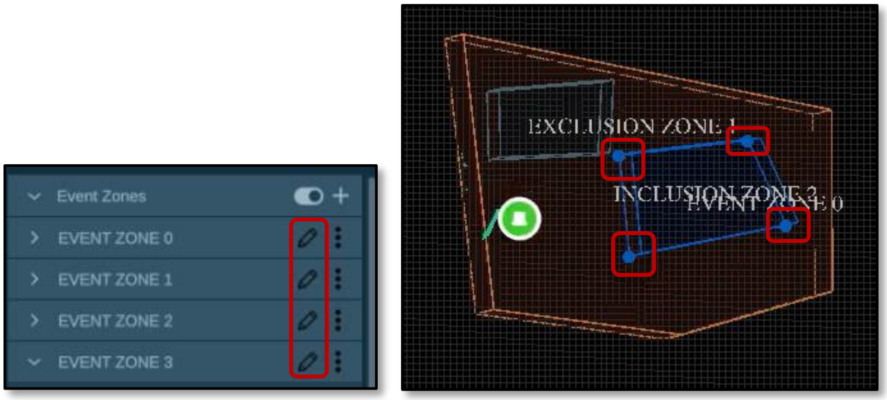


Figure 58. Zone/Line Edit Icon and Editing Vertex Points

Save and Exit Editing a Zone/Line

To save changes and exit zone/counter line Edit mode:

1. Close the zone area. Counter lines do not need to be closed.
2. Double left-click anywhere in the Visualizer.

Cancel and Exit Editing a Zone/Line

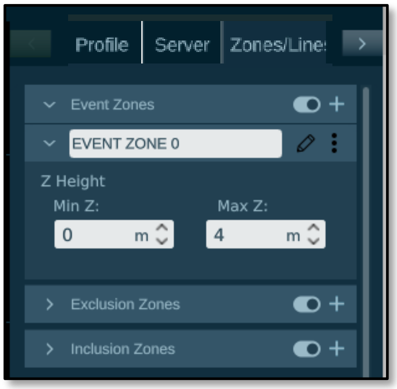
To cancel and exit zone/counter line Edit mode: Press **Escape** [esc] key.

Changes are not saved. All changes since you opened Edit mode, are lost.

Common Zone/Line Widgets

The **Event**, **Exclusion**, **Inclusion**, and **Counter Line** panels enable the user to control creation, visibility, and ability of up to 1000 zones. Each zone type or counter line and individual zone/line has management options. See *Table 12. Zone/Line Management Widgets*.

Table 12. Zone/Line Management Widgets

Management Option	Button	
	OFF/Collapsed	**ON/Expanded**
Change zone/line name: Click on name. Enter new name in field. Click out of field.	 Profile Server Zones/Lines Event Zones EVENT ZONE 0 Z Height Min Z: 0 m Max Z: 4 m Exclusion Zones Inclusion Zones	
Enable/Disable all zones/lines. Click button toggle.	○	●
Expand/Collapse zone/line panel. Click arrow toggle.	›	⌄
Enable/Disable specific zone/line. Click button toggle. Default ON.	□	☑
View/Hide specific zone/line in Visualizer.	👁	👁

Management Option	**Button**	
	OFF/Collapsed	**ON/Expanded**
Click eye icon.		
Delete specific zone/line. Click trashcan icon, click **Apply**, and click **Confirm.**	🗑	🗑

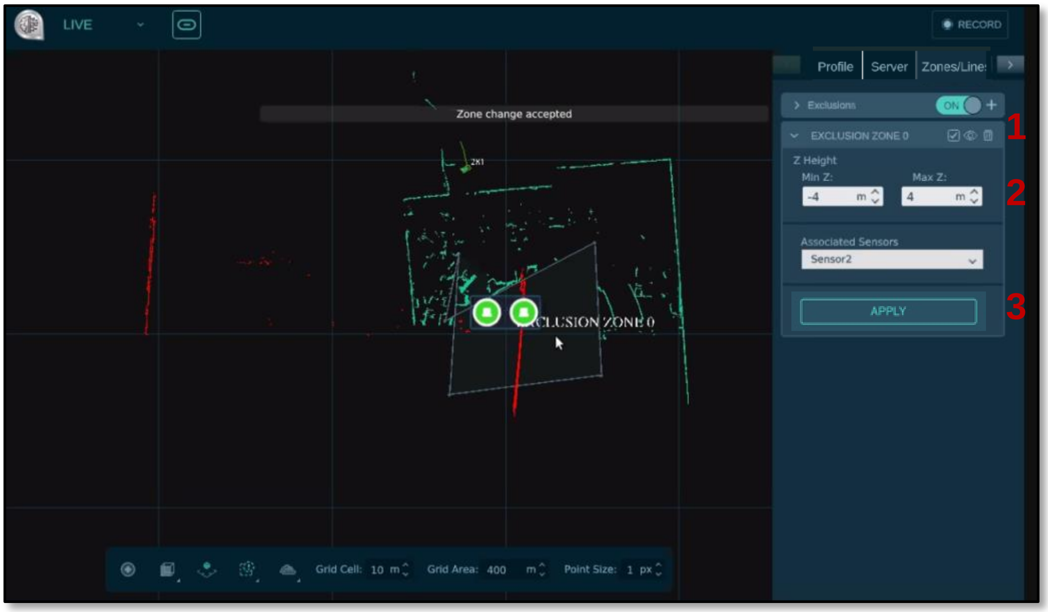
Adjust Vertical Distances of the Zone

In the **Event**, **Exclusion**, or **Inclusion** panels, **Min Z** is the minimum distance, defining the zone bottom plane relative to the world Origin. **Max Z** defines the zone top plane relative to Min Z and world origin. Between the Min Z and Max Z is the range where points are considered meaningful. Any points that are detected outside of that range are discarded.

Default values (Min Z at 0 meters and Max Z at 4 meters) are likely to need adjustment. For example, a sensor mounted on top of a fence or strapped to a pole, needs a negative number for the Min Z and possibly a higher number for Max Z. It can also be useful to adjust Min Z if excluding fluttering branches and leaves, for example, or tracking a person climbing up over a fence while ignoring a person standing next to the fence. See *Figure 59. Adjusting Vertical Height in Zones*.

Adjust a zone Min Z and/or Max Z:

1. In the Visualizer, click the **Reset View** menu, then select **Side View** to more easily visualize relative zone heights within the scene.
2. In the zone tab, select a zone.
3. In the **Z Height** panel, adjust the zone **Min Z** and/or **Max Z** height values in whole numbers by one of these methods:
   - Click the up / down arrows of the user interface or the keyboard to change a value.
   - Click the value, backspace, and type a new value.
4. Click **APPLY** to save the values, update the configuration file on the server, and visualize the results of the choices made.
5. Revise and re-APPLY as many times as needed.



Exclusions ON

EXCLUSION ZONE 0

Z Height

Min Z: -4 m

Max Z: 4 m

Associated Sensors: Sensor2

Figure 59. Adjusting Vertical Height in Zones

Enable Exclusion Zones by Sensor

Some monitored environments have areas that contain reflections, such as bodies of water or mirrored windows. Reflections of objects can sometimes be mistaken for actual objects. One approach to reducing reflections as objects is to create Exclusion zones around the reflecting area. Sometimes an area generates reflections only from a certain direction, in which case, the reflections only affect data collected from selected sensors. See *Figure 60. Sample Reflections in Visualizer*.

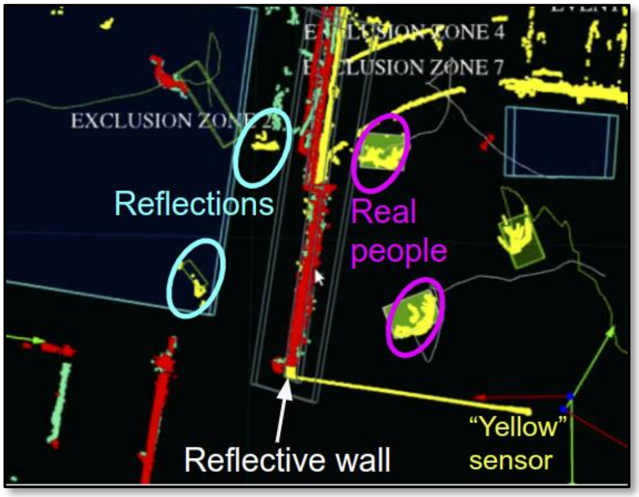


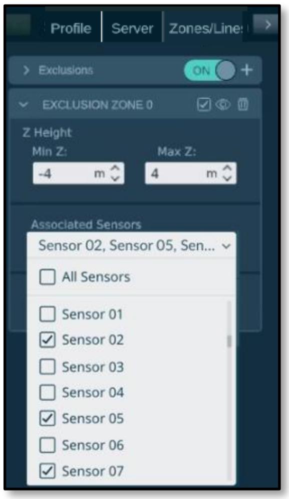
Figure 60. Sample Reflections in Visualizer

For additional methods to adjust for reflections, such as intensity and noise-based settings.xml edits, contact support@quanergy.com.

Use the **Associated Sensor** option to enable Exclusion zones on specific sensors. See *Figure 61. Exclusion Zones by Sensor*.

1. In the Visualizer, click the **Zones/Lines** tab, and expand the **Exclusions** panel.
2. To apply specific Exclusion zones to specific sensors:
   a. Expand the ***zone_name*** **> Associated Sensor** menu.
   b. Click the checkboxes to apply ***zone_name*** to each sensor.
   c. Alternatively, check **All Sensors**.
   d. Click outside the menu.

      The selected `sensor_names` are listed in the **Associated Sensor** box.
3. Click **APPLY**.



Min Z: -4 m

Max Z: 4 m

Associated Sensors: Sensor 02, Sensor 05, Sen...

□ All Sensors

□ Sensor 01

☑ Sensor 02

□ Sensor 03

□ Sensor 04

☑ Sensor 05

□ Sensor 06

☑ Sensor 07

Figure 61. Exclusion Zones by Sensor

13. Visualizing Data

The process of setting up data visualization involves both QORTEX DTC 2.3 server and client, as explained in the following sections.

Start the Operation

Both the server and client must be in operation to visualize LIVE or PLAYBACK (recorded) data:

1. On the Ubuntu machine hosting the QORTEX DTC 2.3 server, confirm the server is running by checking its status. See *Control the Server Daemon Service* (page 57).
2. On the Windows or Ubuntu machine hosting the QORTEX DTC 2.3 client, start the client. See *Start the Client on a Windows Host* (page 58) or *Start the Client on an Ubuntu Host* (page 59).

Monitor Mode

Monitor mode supports multiple monitoring sensors simultaneously. Multiple distributed clients can monitor the same server published data. See *Figure 62. Bottom Menu Bar*.

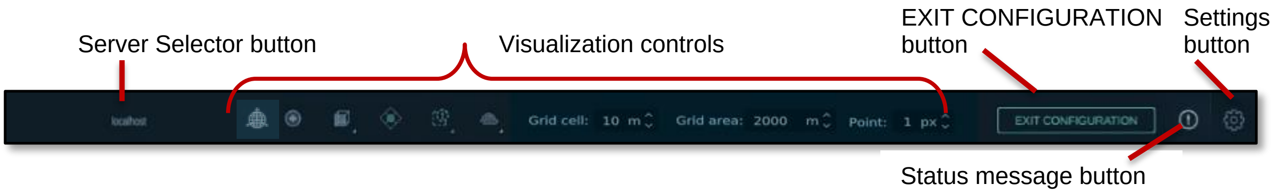


Figure 62. Bottom Menu Bar

To use monitor mode:

1. By default, the client interface is in **Monitor** mode when the following events occur.
   - As soon as the client first starts running.
   - Whenever the interface is not explicitly in **Configuration** mode.
   - After the user has clicked the **EXIT CONFIGURATION** button to exit **Configuration** mode.
2. As soon as a server is selected and before the server has been configured, **Monitor** mode visualizes the server data output without the point cloud (for reduced bandwidth consumption).
3. Monitor objects as they move and interact with any zones/counter lines that are defined.

   Use the Visualizer controls to adjust the view. See the following sections for descriptions of each of the Visualizer control options.

Visualizer Controls

In the bottom of the client screen are the Visualizer controls. The following sections describe the actions for each of the options.

World Coordinate Point Cloud Alignment

The World Origin button toggles adding or removing the World Coordinate object in the visual display. When the World Coordinate object is displayed, you can align the point cloud to a physical plane. For example, the ground or a wall.

To use the World Coordinate object to align the point cloud:

1. Set the World Origin in Q-View for each sensor. See the *Q-View User Guide*.
2. Click the World Origin button to toggle on the World Coordinate object.

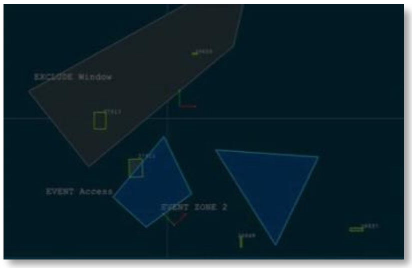
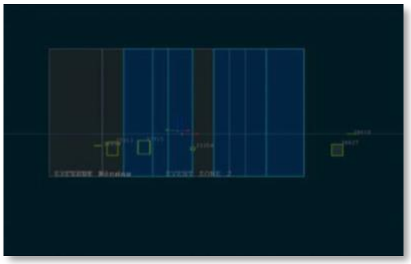
Reset View Toggle

The **Reset view** button toggles the display between Top View or Side View perspective relative to the sensor. See *Table 13. Viewing Controls: Reset View Snaps to a Flat View*.

To use reset view:

1. Click the button to open a pop-up menu to see options for snapping to Top View or Side View.
2. Hover over an icon in the menu to reveal its function.
3. Click the icon for a particular view. The Visualizer and background grid snaps to the selected view.

Table 13. Viewing Controls: Reset View Snaps to a Flat View

Name	Widget	Visual	Example
Top View		Grid and Visualizer snaps to the birds-eye-view as though looking straight down on the top of the sensor, as shown on the right (with Orthographic projection toggled on).	
Side View		Grid and Visualizer snaps to the worms-eye-view as though looking straight into the side of the sensor, as shown on the right (with Orthographic projection toggled on).	

Perspective View through 3D Controls

Use the following mouse (or trackball) controls to fine-tune the display perspective as. If multiple sensors are displayed, then instructions that refer to a specific axis pertain to the sensor specified to be the World Lidar. See *Add or Update Multiple Sensors at a Location* (page 89) and Appendix 4: *Cartesian Coordinate System* (page 292).

To zoom the display in and out, rotate the mouse wheel.

To reposition the display within the visible area:

1. Hold down the right mouse button and move the mouse or mouse ball until the area of interest is located.
2. Release the mouse button.

To spin the display around the red X-axis.

1. Reset the display to the top view.
2. Hold down the left mouse button, then move the mouse or mouse ball up or down along a vertical line.
3. Moving along the line, the view passes from the top view (looking down on the sensor) to a side view (looking at the side of the sensor) to an upside-down view of the sensor.
4. Release the mouse button.

To spin the display around the green Y-axis.

1. Reset the display to the top view.
2. Hold down the left mouse button, and Shift+Drag the mouse or mouse ball left-to-right or right-to-left along a horizontal line.
3. Release the mouse button.

To spin the display around the blue Z-axis.

1. Reset the display to the top view or side view.
2. Hold down the Control+Drag the mouse or mouse ball left-to-right or right-to-left along a horizontal line.
3. Release the mouse button.

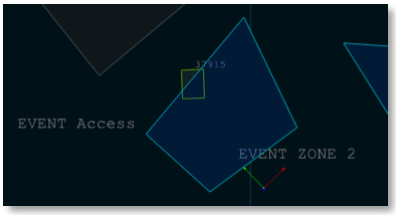
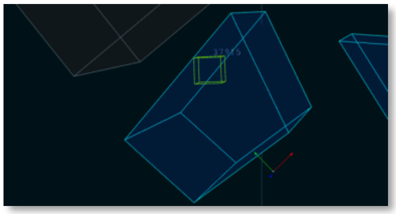
Orthographic Projection Toggle

The **Orthographic projection** toggles between on and off mode. ON switches to 3D view. Off switches to 2D view.

Orthographic projection is most useful in Top View and when trying to determine whether objects are in or near zones/counter lines. See *Table 14. Viewing Controls: Orthographic Projection Toggles On and Off*.

Click the **Orthographic projection** button to toggle between on and off mode. When the Orthographic projection is on, the Visualizer and background grid snaps to the selected mode. This makes it easier to see whether an object is in or out of the zone/counter line boundary.

Table 14. Viewing Controls: Orthographic Projection Toggles On and Off

Name	Widget	Visual	Example
Orthographic Projection is ON (click to turn off)	◈	ON represents 3D view in 2D, so the view of a location is along parallel lines that are perpendicular to the plane of the floor, which clarifies where an object is in relationship with a zone/counter line boundary, as shown in the example image.	
Orthographic Projection is OFF (click to turn on)	●	OFF makes it harder to determine whether an object is inside or outside of a zone/counter line boundary. In the example image, notice how the objects appear to be simultaneously inside and outside of the zone/counter line boundary.	

Visibility: Zones/Lines and Object ID

The **Visibility** menu sets whether to include Exclusions, Events, and/or Trackable Object (Track) IDs in the display. See *Table 15. Viewing Controls: Zones/Lines Hide and Show* Example and *Table 16. Viewing Controls: Track IDs Enabled and Disabled*.

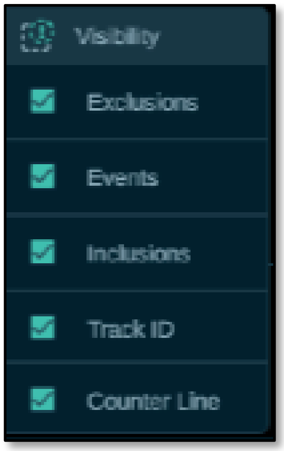


Figure 63. Visibility Menu for Zones and Counter Lines

To use visibility Zones/Lines:

1. Click to open a pop-up menu, which displays options for hiding/unhiding Zones/Lines and track ID.
2. Hover over an item in the menu to reveal its function.
3. Click the box for the preferred option.

   When checked, the Visualizer displays the selected zones/lines and/or track ID labels. When unchecked, the Visualizer hides the selected zone/line and/or track ID label.

   If the **Sensor > Advanced Settings >** `use_timestamp_in_id` option is set to `True`, then the object ID includes a timestamp. The setting for the option is listed in the `settings.xml` file.

Table 15. Viewing Controls: Zones/Lines Hide and Show Example

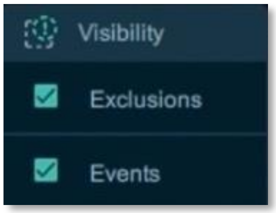
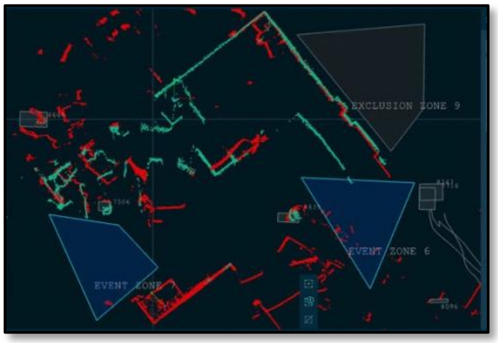
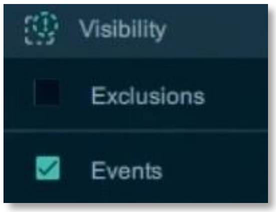
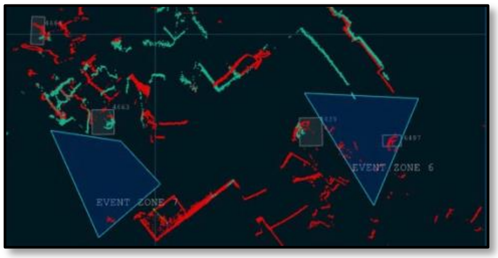
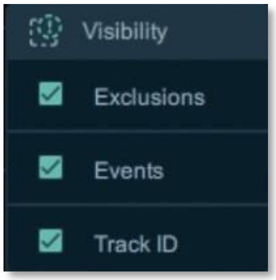
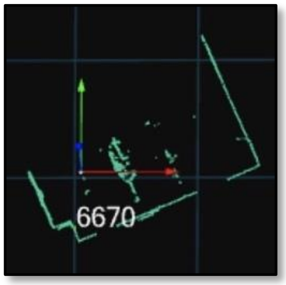
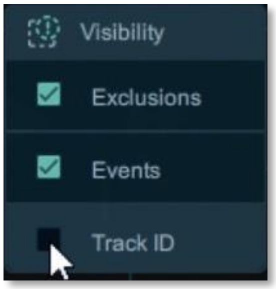
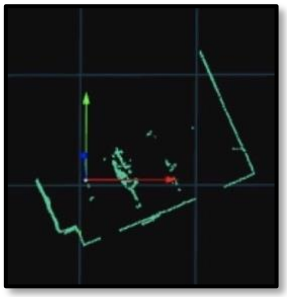
Status	Apply	Visual	Example
Checked Exclusions and Events	 Visibility ☑ Exclusions ☑ Events	All Event zones and all Exclusion zones are shown in the Visualizer pane.	
Checked Events Only	 Visibility ☐ Exclusions ☑ Events	All Event zones are shown in the Visualizer pane.	

Table 16. Viewing Controls: Track IDs Enabled and Disabled

Status	Apply	Visual	Example
Checked Track ID	 Visibility ☑ Exclusions ☑ Events ☑ Track ID	All Track IDs are shown in the Visualizer window.	

Status	Apply	Visual	Example
Unchecked Track ID		All Track IDs are hidden in the Visualizer window.	

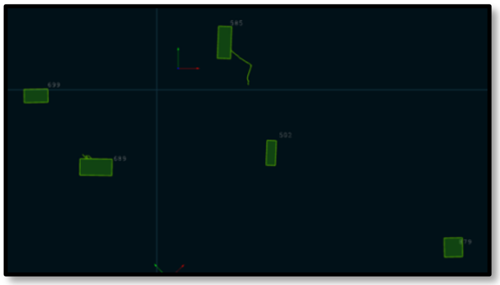
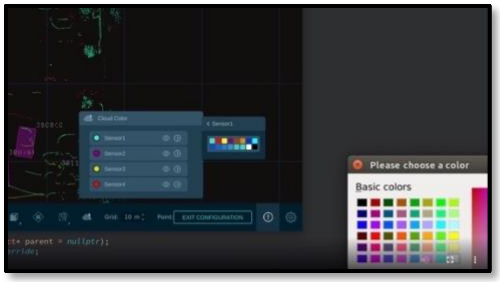
Point Cloud Display

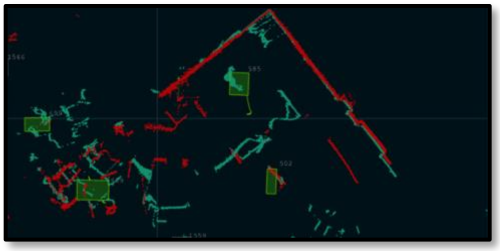
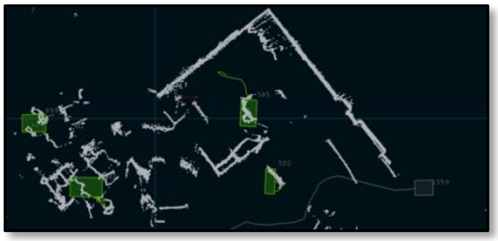
The **Point Cloud Display** toggles between multi-sensor viewing options: **No Cloud**, **Color Picker**, **Split**, and **Combined/Fused**. This menu has no effect in **Monitor** mode. See *Table 17. Viewing Controls: Point Cloud Visualization*.

To use point cloud display:

1. Click it to open a pop-up menu of options for hiding/unhiding point clouds.
2. Hover over an icon in the menu to reveal its function.
3. Click the icon for the preferred option. The Visualizer conforms to the request.

Table 17. Viewing Controls: Point Cloud Visualization

Name	Widget	Visual	Example
No Cloud	☁	The point clouds are not visualized in the Visualizer window.	
Color Picker	☁	In the **Cloud Color** panel: Hide/show 👁 a sensor point cloud. Click arrow ▶ to select a different color swatch. Click black swatch ■ to pick/define new color.	

Name	Widget	Visual	Example
Split	☁	Each sensor point cloud is visualized in a different color in the Visualizer window. This is useful for troubleshooting.	
Combined/ Fused	☁	All sensors' point clouds are combined into a single fused point cloud that is visualized in a single color in the Visualizer window.	

Grid Cell

The **Grid Cell** menu Grid Cell: 10 m adjusts the background grid spacing in the display. Measured in meters. There are always at least 4 grid cells, each with the connecting point from the center of the area. Grid cell square minimum size is 1 meter, maximum is 100 meters. See *Figure 64. Viewing Controls: Grid Cell 50 m (left), 10 m (middle), 1 m (right)*.

To use grid cell settings:

> **Note:** If the monitor resolution is 1440x900, 1280x800, 1024x768, or 1280x1024, exit **Configuration** mode before proceeding.

1. Click (or click and hold) an up or down arrow on Grid Cell selection menu to adjust the number increment.
2. Click the number and backspace to remove a digit, or type to add a digit. When the number is correct, click the **Enter** button.

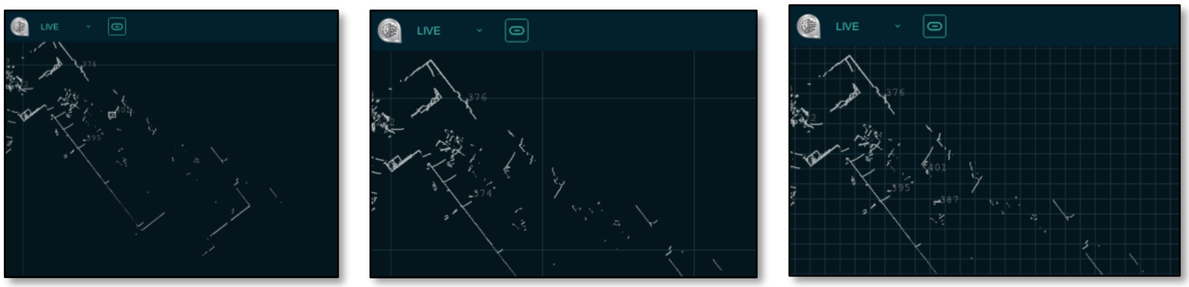


Figure 64. Viewing Controls: Grid Cell 50 m (left), 10 m (middle), 1 m (right) Grid Area

Grid Area

The **Grid Area** menu Grid Area: 400 m adjusts the size of the monitored area. The measurement applies to all sides of the monitored area. This option adjusts the viewable area that is displayed in the Visualizer. It does not change the view in the Visualizer. Grid Area along each side is minimum 10 meters, maximum 50,000 meters. See *Figure 65. Viewing Controls: Grid Area 10 m (left), 25000 m (middle), 50000 m (right)*.

To use grid area settings:

> **Note:** If the monitor resolution is 1440x900, 1280x800, 1024x768, or 1280x1024, exit **Configuration** mode before proceeding.

1. Click (or click and hold) an up or down arrow on Grid Area selection menu to adjust the number increment.
2. Click the number and backspace to remove a digit, or type to add a digit. When the number is correct, click the **Enter** button.

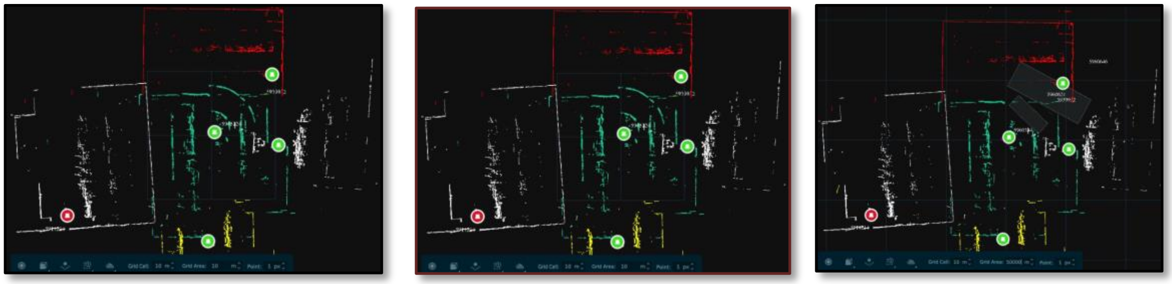


Figure 65. Viewing Controls: Grid Area 10 m (left), 25000 m (middle), 50000 m (right)

Point Size

The **Point** menu Point: 1 px adjusts the size of the points in the display. Measured in pixels, adjustable in increments of 1 pixel, within a range of 1 pixel for the finest size (default) to 5 pixels for the coarsest size. See *Figure 66. Viewing Controls: Point Size of 5 px (left), 3 px (middle), 1 px (right)*.

> **Note:** With client installations, depending upon the computer graphic card, the point cloud size might not change when the value is changed in the **Point Size** setting.

To use point size settings:

> **Note:** If the monitor resolution is 1440x900, 1280x800, 1024x768, or 1280x1024, exit **Configuration** mode before proceeding.

1. Click (or click and hold) an up or down arrow to adjust the number increment.
2. Click the number and backspace to remove a digit, or type to add a digit, then click the **Enter** button when the number is satisfactory.

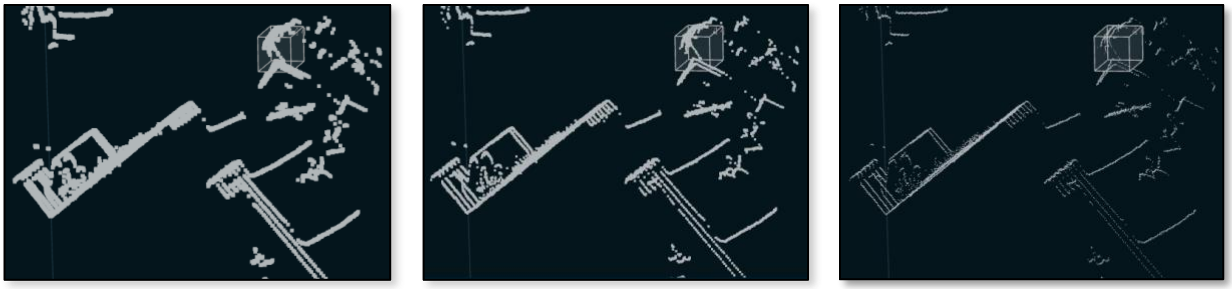


Figure 66. Viewing Controls: Point Size of 5 px (left), 3 px (middle), 1 px (right)

Visualize Live Data

LIVE data is visualized according to the configuration saved in the settings file. See *View PTZ Camera Details* (page 140).

1. If the top left corner of the interface indicates **Disconnected** status, click that button to reconnect sensor(s) to the server and visualize the collected data. When connected, the button status is .

   - For a sample visualization of single-sensor `settings.xml` file output, see *Figure 67. Visualize Single-Sensor LIVE Data*.
   - For a sample visualization of multiple-sensor `transform_alignment.xml` file output (no zones) from Q-View, see *Figure 68. Visualization based on transform_alignment.xml File without Zones*.
   - For a sample visualization of the `settings.ini` file output (with zones) from QORTEX DTC 1.x, see *Figure 69. Visualization based on settings.ini File with Preconfigured Zones*.

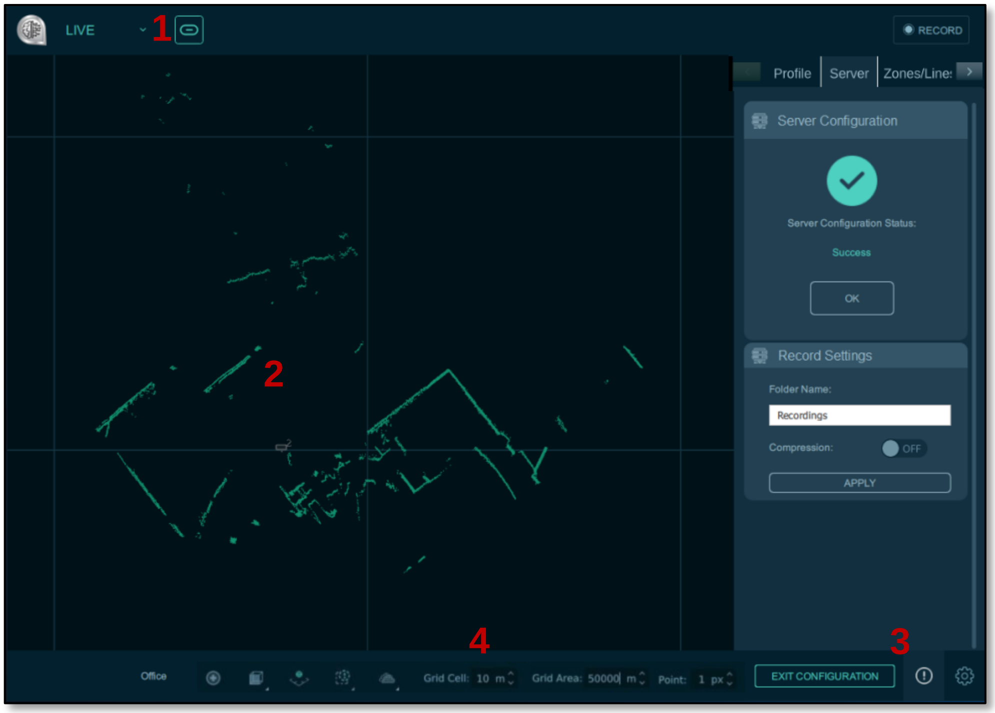
   It may take up to 30 seconds for the sensors to spin up and data to appear in the Visualizer window. By default, each sensor point cloud appears in a different color.

2. In the **Visualizer** window of the client interface, visually confirm the point cloud, objects, and tracks, if any are moving in the sensors' or PTZ camera area. See *Table 6. Visualizer Elements and Colors*, *Table 7. Visualizer Zone Types, Counter Line, and Colors,* and *Table 8. Visualizer Tracked Objects and Colors*.

3. In the bottom right corner of the client interface, click the **Settings** button ⚙ to close the **Configuration** panel and see more of the Visualizer window.

4. To select options for the point cloud view, click the icons in the visualizer bar.

5. To turn off the point cloud (to conserve bandwidth), click the **EXIT CONFIGURATION** button in the same corner of the window. Visualization of objects, tracks, zones/counter lines, and so on continues without the point cloud.



Server Configuration Status:
Success

Folder Name: Recordings

Compression: OFF

Grid Cell: 10 m Grid Area: 50000 m Point: 1 px

Figure 67. Visualize Single-Sensor LIVE Data

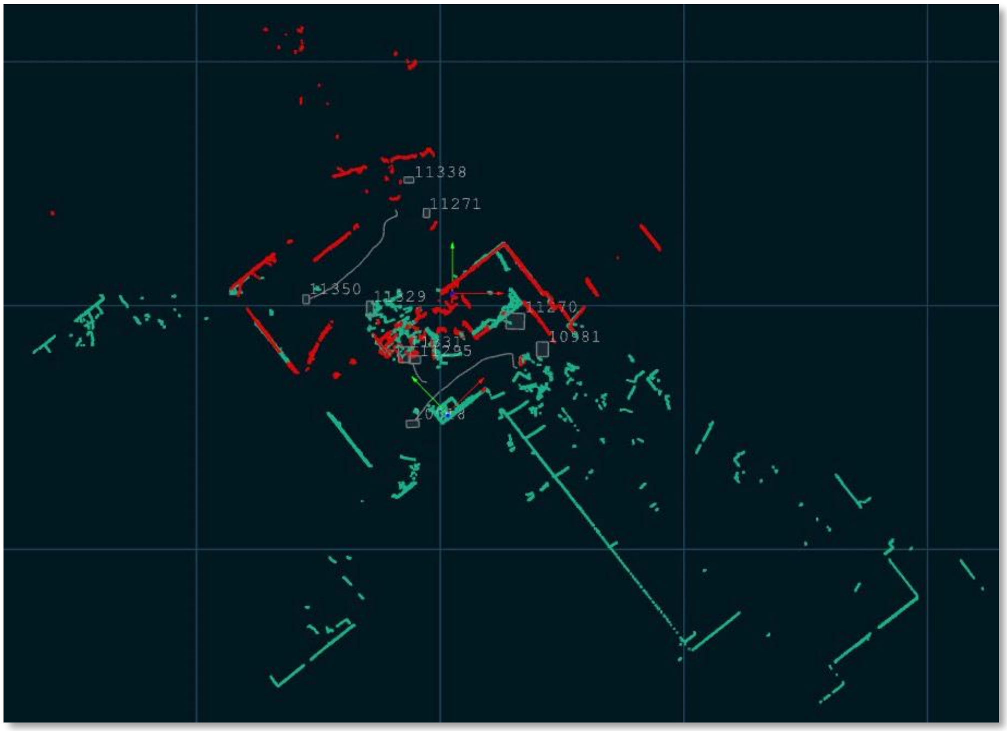


Figure 68. Visualization based on `transform_alignment.xml` File without Zones

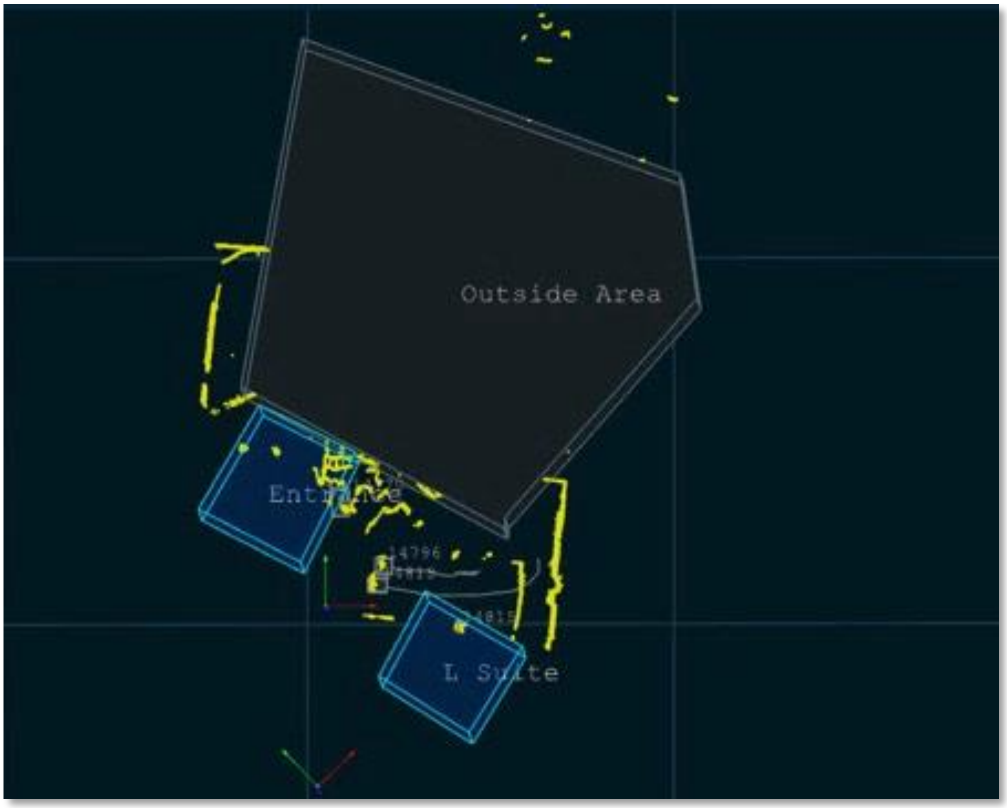


Figure 69. Visualization based on `settings.ini` File with Preconfigured Zones

14. Adding PTZ Cameras

A server can be configured to control and visualize data from up to five PTZ cameras per server. Configure the PTZ camera, so QORTEX DTC server can collect data on location and objects. Adding PTZ cameras also enables optionally recording video.

The button indicating **Connected** or **Disconnected** status in the upper left corner of the interface window is a toggle mechanism that connects the server to, or disconnects the server from, the sensors and PTZ cameras.

Summary of steps to use PTZ cameras:

1. Verify the specific PTZ camera license.
2. Enter **Configuration** mode. See *Entering Configuration Mode* (84).
3. Add PTZ camera to the QORTEX DTC server.
4. Calibrate PTZ cameras to align with the sensors to track objects.
5. Configure PTZ camera Field of View (FOV) to help QORTEX DTC determine which objects are within view of the PTZ camera.
6. Define the PTZ camera Home Location, to which it returns when idle. When idle, PTZ cameras can be used by other applications.
7. Track objects already detected by QORTEX DTC server that are in the PTZ Camera FOV. Automatically track objects using Rules. Manually track objects through the PTZ camera icon in the Visualizer.

Supported PTZ Cameras

Quanergy supports Pan-Tilt-Zoom (PTZ) cameras that are compliant with the global standards set by the Open Network Video Interface Forum (OnVIF), Profile S (streaming video). Such compliance ensures the effective interoperability of security products operating under the Internet Protocol (IP). The following PTZ cameras have been tested with QORTEX DTC:

- AXIS Q6115-E PTZ Network Camera
- Bosch AUTODOME IP starlight 5000i
- Bosch MIC IP starlight 7100i

> **Note:** When using Bosch MIC IP starlight 7100i, ensure the camera is on the latest Bosch firmware update. See https://downloadstore.boschsecurity.com/FILES/Bosch_Releaseletter_CPP7.3_FW_7.60.0118.pdf.

- FLIR Systems Saros DM-Series
- GeoVision [Beta]

- Hanwha / Samsung [Beta]
- Hikvision DS-2DF8336IV-AEL
- Hikvision DS-2DE5225IW-AE

PTZ Camera Terms

- **Pan** — a sideways rotation to scan horizontally.
- **Tilt** — an up/down movement to scan vertically.
- **Zoom** — an in/out magnification of the observed frame.

Prepare the PTZ Camera

It is outside the scope of this user guide to explain exactly how to set up your camera, which is affected by the model's capabilities, specific environment, and intended usage.

The general steps involved in setting up any QORTEX DTC-compliant camera are:

1. **Understand**. Check the camera datasheet or published specifications to understand its particular abilities, limitations, and requirements.
2. **Define**. In the web settings, according to camera instructions, define an OnVIF user.
3. **Mount**. Securely mount the camera according to the camera's installation instructions. Some general, common sense mounting tips include placing the camera in a permanent, static location where:
    - The camera will not be moved or bumped.
    - Both camera and LiDAR sensors share common lines of sight.
    - The camera and LiDAR sensors don't block each other's view.
    - No other static objects block the camera's view.
4. **Network**. Connect the camera to the same subnet that the QORTEX DTC server host computer Ethernet is on.
5. **Stream**. Follow the camera's instructions for assigning an IP address, accessing the video stream through a web browser, and performing any other setup procedures.
6. **View**. Place the QORTEX DTC client window next to the web browser displaying the camera output for maximum situational awareness and interactivity.
7. **Routine**. Follow all of the camera's safety, cleaning, and maintenance instructions.

Add a PTZ Camera

Add a PTZ camera through the QORTEX DTC client Configuration panel. Alternatively, you can edit the settings file. The steps below describe the client option. See *Figure 70. Add PTZ Camera Steps*.

You can add up to five PTZ cameras to your QORTEX DTC server. These provide object location for automated security.

1. Prepare to add a PTZ camera.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From **Configuration** panel, select the **PTZ** tab and **Unlock PTZ camera** configuration.

   Click the lock icon on the PTZ camera tab to switch to the unlocked icon.

   If the PTZ tab is grayed out, verify and update licenses. Licensing must state support for 1 or more PTZ cameras. There is no hard limit in terms of the number of PTZ cameras that can be supported per server, but Quanergy has tested and verified support for up to 5 PTZ cameras.

3. Click the **ADD CAMERA** button.

4. Add PTZ camera details, then click **Next**.

   - **Name** field—assign a name for the PTZ camera.
   - **User** and **Password** fields—enter the credentials required to access the PTZ camera.
   - **IP** field—enter the IP address for access to the PTZ camera. Every PTZ camera must have its own IP address. QORTEX DTC supports both IPv4 and IPv6 IP addresses.
   - **Model** menu—select the manufacturer for the PTZ camera displayed in the list. Select **UNKNOWN**, if your PTZ camera is not listed.
   - **Activation URL**—is not required. Used for enabling the camera to handle positioning requests with OnVIF.
   - **Video URL**—is automatically assigned. Used for the Real Time Streaming Protocol (RTSP) video stream.

     If you select **UNKNOWN** for the **Model**, manually enter the URL for your PTZ camera.

   When the activated PTZ camera is activated, the Visualizer displays the activated PTZ camera icon ●.

   If the busy icon does not go away, the PTZ camera is not added. Correct the activation data and click **Apply**.

If the PTZ camera icon has a hash line across it, it means the PTZ camera is not activated. Correct the activation data and click **Apply**.

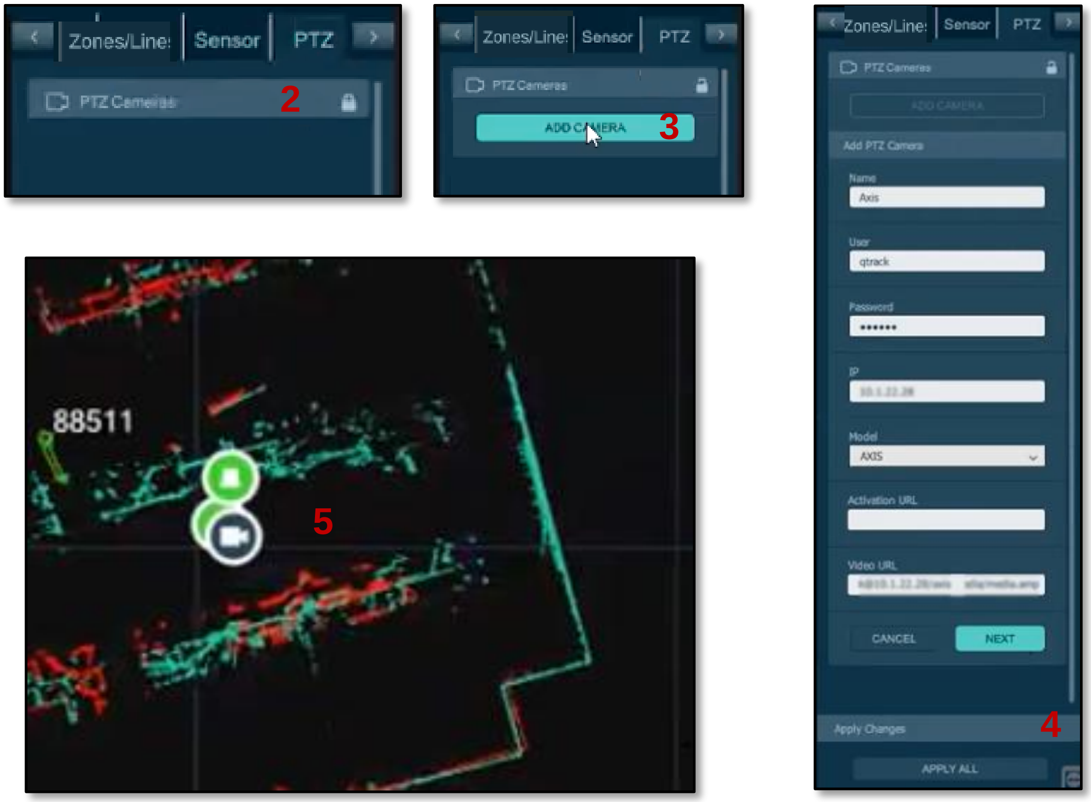
Each PTZ camera must have its own IP address and must be on the same network as the sensors and QORTEX DTC.

5. When the PTZ camera is added. Click **Apply Changes.**

   While the server adds the PTZ camera, a busy icon is displayed in the PTZ panel. When completed:

   - The settings file is updated with the PTZ camera settings.
   - The PTZ camera appears in the Visualizer.

6. Repeat these steps for each PTZ camera you want to connect. Provide a unique name and IP address for each PTZ camera.



Name: Axis
User: qtrack
Password: ******
IP
Model: AXIS
Activation URL
Video URL

Figure 70. Add PTZ Camera Steps

Calibrate a PTZ Camera

Calibrate PTZ cameras to align with the sensors to track objects.

Calibration is a semi-automatic process. Collect a minimum three pairs of sample points. Each of three pairs creates an association between the location of an object in the Visualizer (such as the LiDARs frame of reference) and the location of the object in the PTZ camera video.

When selecting locations for the point pairs, try to cover, as completely as possible, the physical area in which objects are expected to be tracked by QORTEX DTC. This ensures accurate camera tracking throughout the area.

1. Prepare to calibrate the PTZ camera.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From **Configuration** panel, go to the **PTZ** tab. Unlock PTZ camera configuration, if locked.

3. Expand the tabs: ***PTZ camera name*** > **Calibration > Calibrate Camera.** Click the **Start** button.

   In the example, the PTZ camera name is `Axis_work`. See *Figure 71. Start PTZ Camera Calibration*.

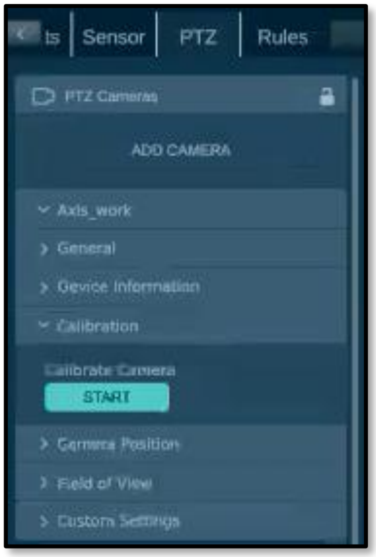


Figure 71. Start PTZ Camera Calibration

PTZ Camera video panel opens. Calibrate PTZ Camera wizard starts in Configuration panel.

4. Set first point of first pair. **Position A.**

   a. While standing at least 2 meters in front of the camera, click on the track box you are associated with in the Visualizer.

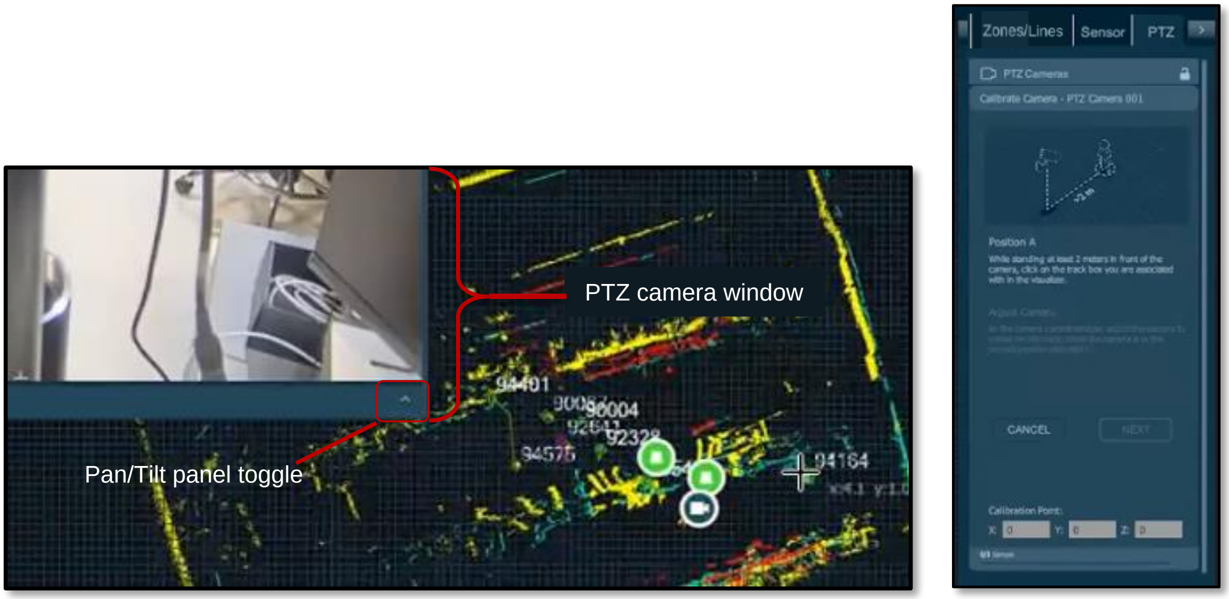
   b. In the Visualizer, place the cross-hair (**+)** on an object with a track ID and click. See *Figure 72. PTZ Camera Calibration: Position A*.

   c. Adjust PTZ Camera.

   In PTZ camera video view, adjust PTZ camera to locate object using:

   - Click and drag the entire PTZ camera video window to move it around the Vizualizer window.
   - Select the PTZ camera video window, then using the keyboard up-down and left-right arrows to move the PTZ camera focus.
   - Use the mouse wheel to move the PTZ camera focus in or out.
   - Adjust the pan and tilt axis of the PTZ camera. At the bottom right of the PTZ camera video, click the ^/v toggle to expand/close the Pan and Tilt fields panel. Adjust the values using the sliders.

   As slider shifts, values also change in top bar of PTZ camera video view.



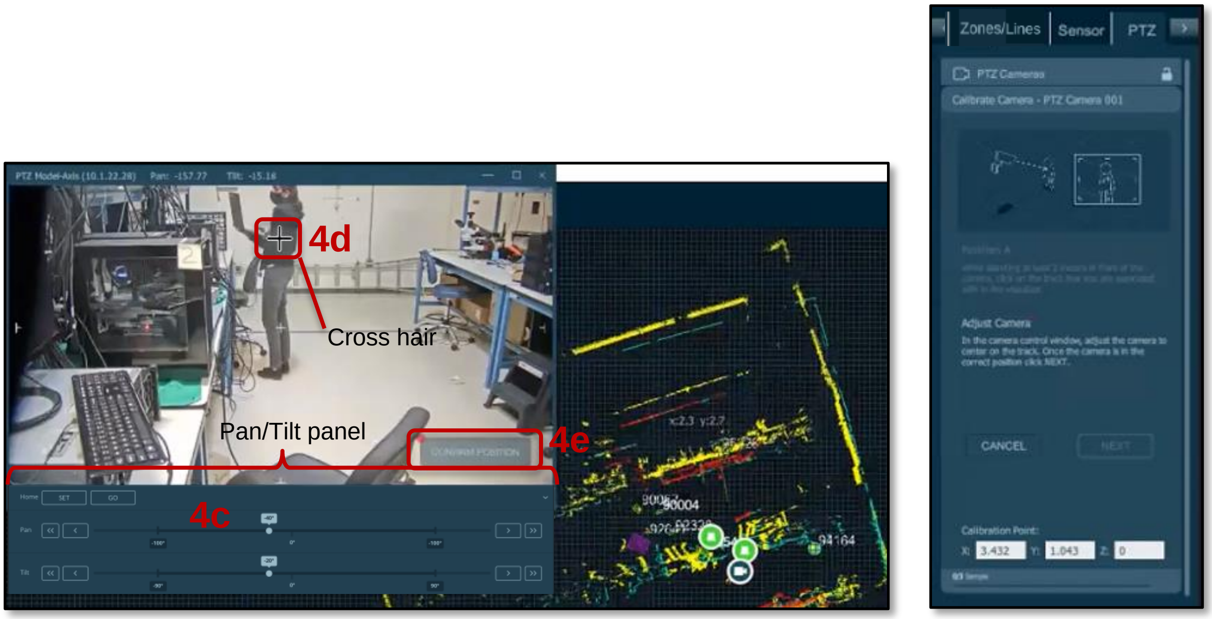
Position A

While standing at least 2 meters in front of the camera, click on the track box you are associated with in the visualizer.

Calibration Point:
X: 0 Y: 0 Z: 0

Figure 72. PTZ Camera Calibration: Position A

d. In the PTZ camera video, center and place the cross hair (**+)** symbol to be centered on the object. For a human mid-chest height is good.



Adjust Camera

In the camera control window, adjust the camera to center on the track. Once the camera is in the correct position click NEXT.

Calibration Point:

X: 3.432 Y: 1.043 Z: 0

Figure 73. PTZ Camera Calibration: Set Video Position

e. In the PTZ camera tab, click **CONFIRM POSITION** button. See *Figure 73. PTZ Camera Calibration: Set Video Position*.

f. In the **Configuration** panel, click **Next**.

The cross-hair in the Visualizer changes to a colored sphere.

5. Set second point of first pair. **Position B.** See *Figure 74. PTZ Camera Calibration: Position B Visualizer and Video*.

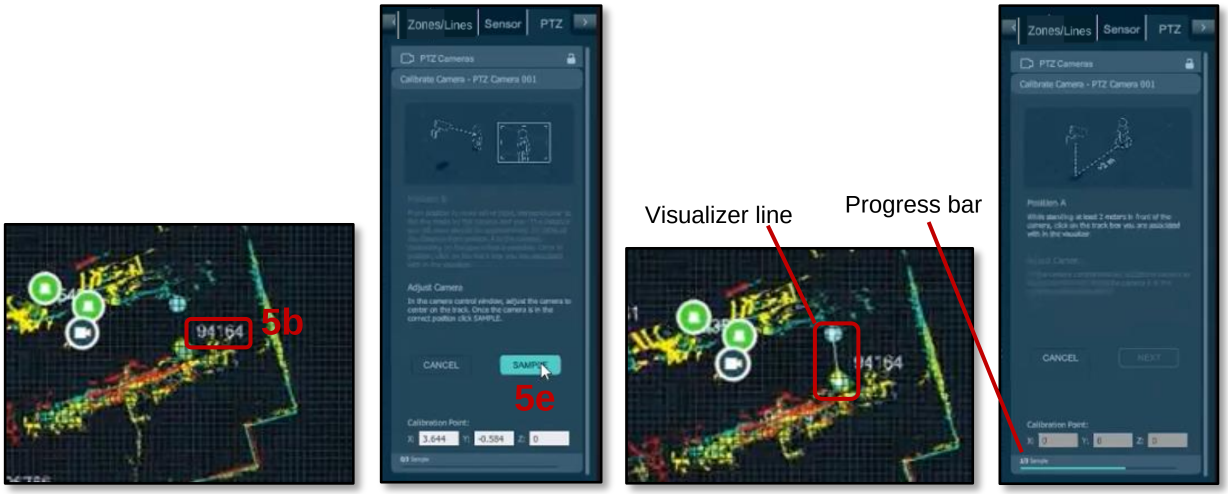
a. Physically move the object to a different place within the sensor defined location.

b. In Visualizer, click the cross-hair (**+)** on the object ID used in Position A. This can be a different object than was used for Position A.

c. In PTZ camera video, adjust PTZ Camera using keyboard, mouse, or expanded Pan/Tilt panel. See Step 4c.

d. In PTZ camera video, place cross-hair (**+)** on the object. The cross-hair in the Visualizer changes to a colored point.

e. In PTZ Camera tab, click **SAMPLE**.



Adjust Camera
In the camera control window, adjust the camera to center on the track. Once the camera is in the correct position click SAMPLE.

Calibration Point: X: 3.644 Y: -0.584 Z: 0

Position A
While standing at least 2 meters in front of the camera, click on the track box you are associated with in the visualizer

Calibration Point: X: 0 Y: 0 Z: 0

Figure 74. PTZ Camera Calibration: Position B Visualizer and Video

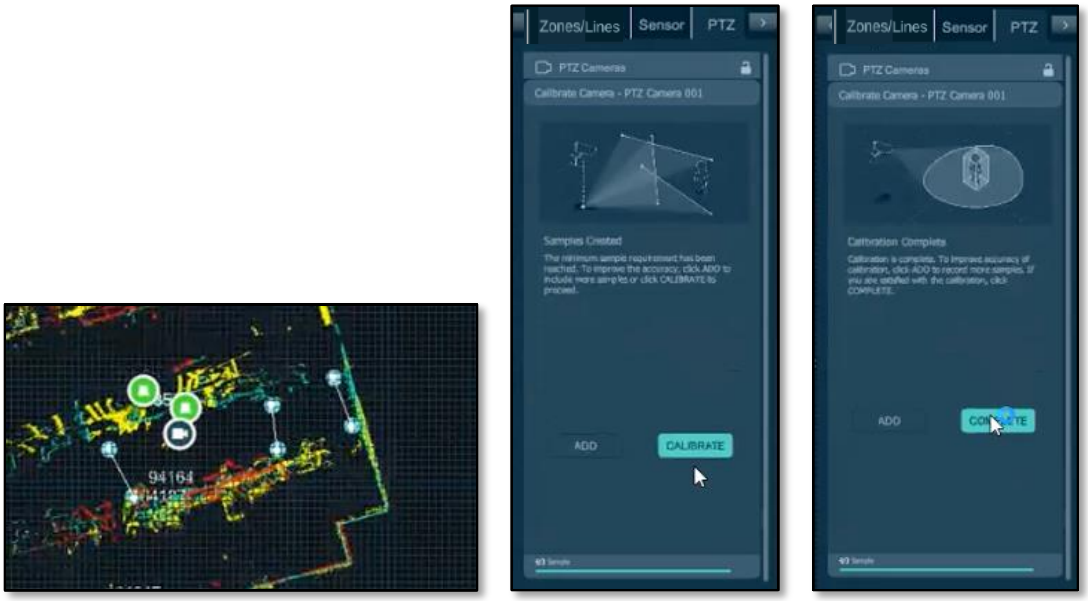
Progress bar indicates, one pair is complete. In Visualizer a line is drawn between Position A and Adjusted PTZ Camera point.

Notice the Calibration Point values in the Configuration panel have changed from all zeros.

6. Set second and third pair. Repeat a Position A and Position B.

   Click **Add** for more samples, if needed.

7. In Configuration panel, click **CALIBRATE** button. Click **COMPLETE**. See *Figure 75. PTZ Camera Calibration: Calibrate All Points*.



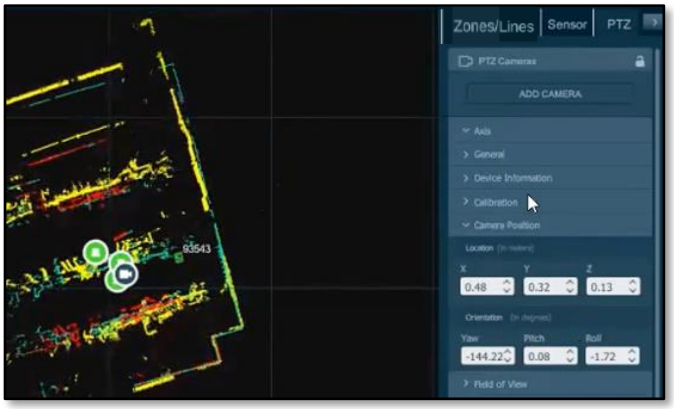
Samples Created
The minimum sample requirement has been reached. To improve the accuracy, click ADD to include more samples or click CALIBRATE to proceed.

Calibration Complete
Calibration is complete. To improve accuracy of calibration, click ADD to record more samples. If you are satisfied with the calibration, click COMPLETE.

Figure 75. PTZ Camera Calibration: Calibrate All Points

8. View the PTZ calibrated values. Expand the **Camera Position** tab. See *Figure 76. PTZ Camera Calibration: Verify Camera Position*.

If calibration fails, the calibration data is not populated, and an error message is displayed in the bottom status bar.



Location (in meters)

X 0.48 Y 0.32 Z 0.13

Orientation (in degrees)

Yaw -144.22 Pitch 0.08 Roll -1.72

Figure 76. PTZ Camera Calibration: Verify Camera Position

9. Continue to edit the PTZ camera Field of View. See *Add PTZ Camera Field of View* (page 133).

Add PTZ Camera Field of View

Configure PTZ camera Field of View (FOV) to designate the camera coverage area and help determine which objects are visible to the camera. Camera following distance is 3 to 50 meters. Each PTZ camera has its own FOV.

Note that zones/lines are the areas created within a sensor field of view. Zones/lines can only be polygons or lines. PTZ camera field of view can be circle, arc, or polygon, but they are not to be confused with zones/counter lines.

1. Prepare to define the PTZ camera Field of View.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From **Configuration** panel, go to the **PTZ** tab. Unlock PTZ camera configuration, if locked.

3. Select **PTZ Camera > PTZ Camera name tab > Field of View:** CIRCLE or POLYGON.

   Maximum distance away from the PTZ camera is 500 meters.

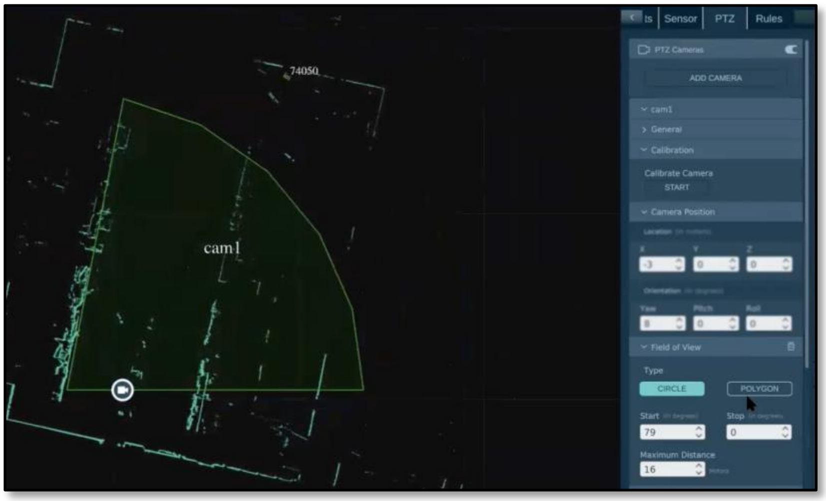
4. **Circle FOV**. See *Figure 77. PTZ Camera Circle FOV*.

   a. Click **CIRCLE** button.

   b. In the **Field of View** panel, the values entered center around PTZ camera.

      - **Start**—measured in degrees from the center of the camera. Range is -180 to 180.

- **Stop**—measured in degrees from the Start position, creates the width of the arc of the circle. Range is -180 to 180 degrees.
- **Maximum Distance—**measured in meters, the length of radius from center point. Range is 0 to 200 meters.



X -3 Y 0 Z 0

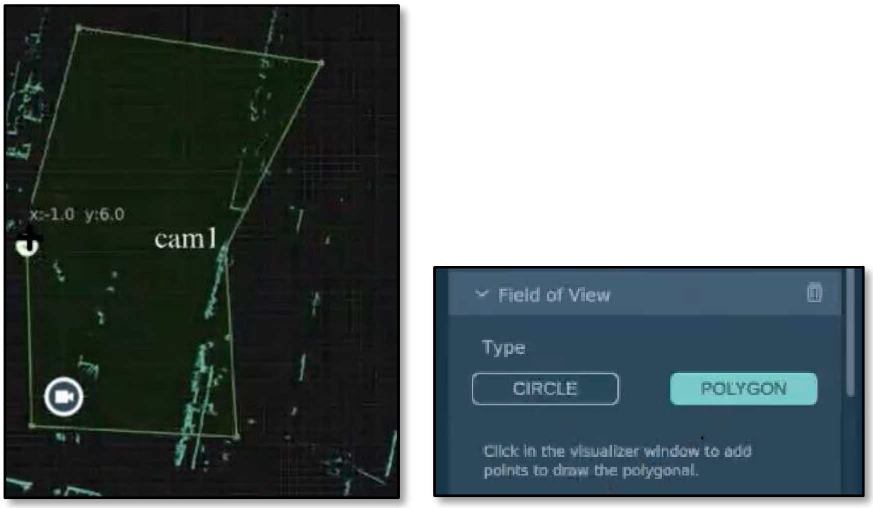
Yaw 8 Pitch 0 Roll 0

Start 79 Stop 0

Maximum Distance 16

Figure 77. PTZ Camera Circle FOV

5. **Polygon FOV.** See *Figure 78. PTZ Camera Polygon FOV*.
   a. Click **POLYGON** button.
   b. Click in Visualizer minimum of three points to create a defined space.



Click in the visualizer window to add points to draw the polygonal.

Figure 78. PTZ Camera Polygon FOV

6. Click **APPLY ALL**.
7. Continue to add a Home Location. See *Set PTZ Camera Home Location* (page 135).

Set PTZ Camera Home Location

When idle, PTZ cameras can be used by other applications. This is where the PTZ camera returns its point of view location when in resting or idle state, and when in lock mode.

The Home Location is important to enable the PTZ camera to efficiently capture an intruder (person, vehicle) as reported by the QORTEX DTC system Home Location can help the camera readiness to turn the minimal angle for responsiveness. Use a position where you might expect a zone intrusion to occur.

Recommended use: Set the home location so that the PTZ camera can cover the most probable route of objects or provide an immediate view of objects entering an Event zone.

Define the PTZ camera Home Location. See *Figure 79. PTZ Camera Set Home Location*.

1. Prepare to define the PTZ camera home location.

   Connect your QORTEX DTC client to a QORTEX DTC server. See *Connect the Client to the Server* (page 61).

   Setting the PTZ camera home location does not require **Configuration** mode.

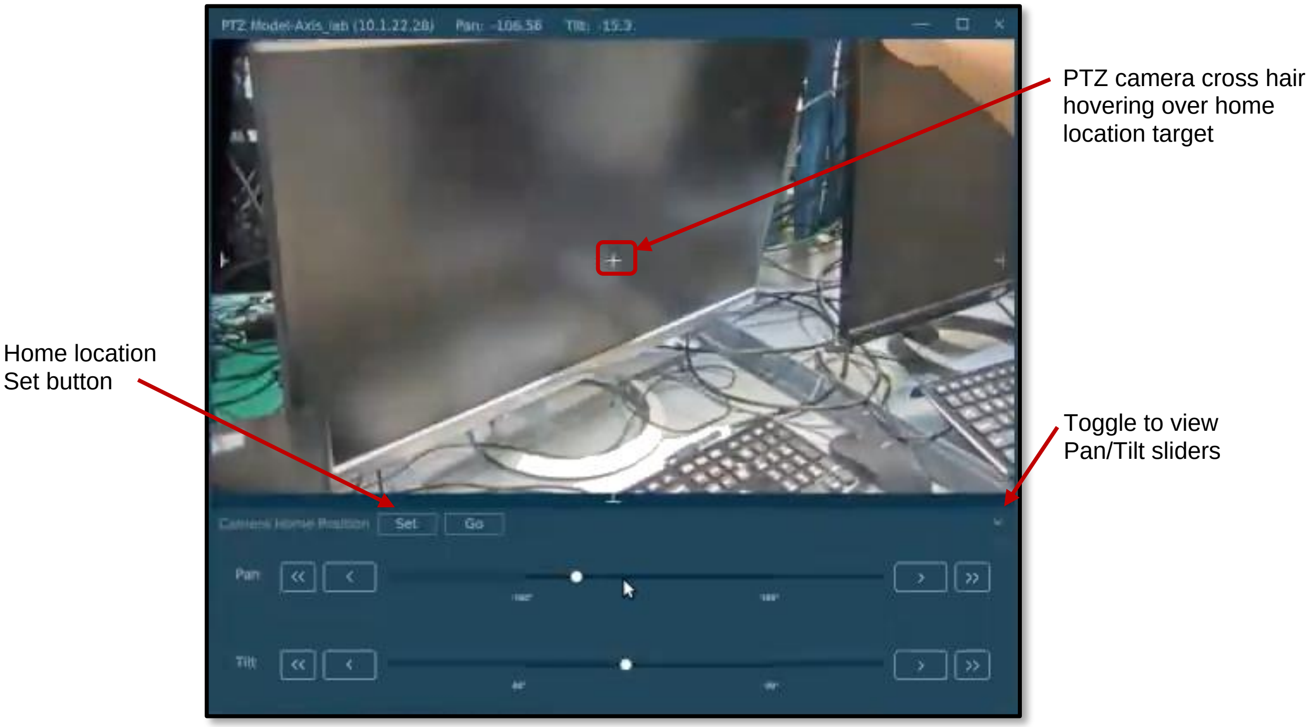
2. Open either a PTZ camera Calibration window or PTZ camera video window. See *Calibrate a PTZ Camera* (page 129) or *View the PTZ Camera Video Feed* (page 137)

3. Click the toggle to open the **Pan and Tilt slider** panel. Optionally, use the keyboard directional arrows to shift the position.

4. Position the PTZ camera, adjusting the Pan and Tilt settings until the cross hair ✛ is pointing to your preferred home location.

5. Click the **Set** button at the bottom of the PTZ camera window.

6. When PTZ camera calibration is complete, lock it. On the PTZ Camera tab, click the lock toggle to close the **Lock** 🔒.



PTZ camera cross hair hovering over home location target

Home location Set button

Toggle to view Pan/Tilt sliders

Figure 79. PTZ Camera Set Home Location

Edit PTZ Camera Custom Settings

Keep the default settings unless you are using a non-standard PTZ camera. If your PTZ camera has different values, as determined by the manufacturer, enter them here. The content updates the PTZ camera library file. See *Figure 80. PTZ Camera Custom Settings*.

1. Prepare to edit the PTZ camera settings

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From **Configuration** panel, scroll to the **PTZ** tab. Unlock PTZ camera configuration, if locked.

3. From the **Configuration** panel, expand the tabs to **PTZ > PTZ Camera > *PTZ Camera name* > Custom Settings**.

4. Click the **^/v** arrows to increase or decrease the values for each field. Sets position space in degrees.

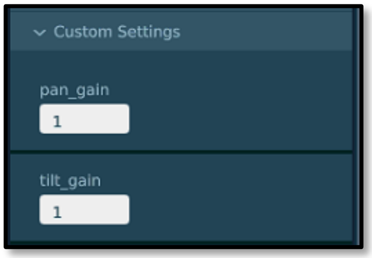
   For example, these values are adjusted to the values reported by an Axis model PTZ camera. The specific fields that are display vary depending upon the PTZ camera model. If you selected **UNKNOWN** camera type, all the possible fields are listed for you to edit.

- **pan_gain**—defines the relative PTZ camera pan left-right move request during calibration. Default is 1.0.

  reported_pan_range—defines the range before customizing. Default is [0, 360]. Use [360,0] to reverse the range.

- **tilt_pain**—defines the relative PTZ camera tilt up-down move request during calibration. Default is 1.0.

  reported_tilt_range—defines the range before customizing. Default is [-90, 90]. Use [90, -90] to reverse the range.



Custom Settings

pan_gain: 1

tilt_gain: 1

Figure 80. PTZ Camera Custom Settings

5. When your edits are complete, click **Apply Changes** or **APPLY ALL**, as highlighted.

View the PTZ Camera Video Feed

When a PTZ camera is added to a server, a PTZ camera icon and an option to view the live PTZ camera feed, is added to the Visualizer. The pop-out PTZ camera video uses the Video URL assigned when the PTZ camera is configured.

View the Nested PTZ Camera Feed

To view the PTZ camera video, from the Visualizer, click the camera icon.

Adjust the Nested PTZ Camera Feed

1. From the Visualizer, expand the **Sensor Health** icon in the Visualizer Controls bar and set PTZ camera to **Axis** mode.
2. Select the PTZ camera to view. Left-click the feed for a specific PTZ camera.
3. Select viewing options.

   From the menu:

   - **Minimize**—Shrinks the PTZ camera feed to a reference point.
   - **Maximize**—Expands the PTZ camera feed to fill the space of the Visualizer.
   - **Restore**—Returns the PTZ camera feed to the thumbnail size.

   Using a mouse click and drag:

    - **Close Focus**—Shifts PTZ camera focus closer in.
    - **Distant Focus**—Shift PTZ camera focus further out.

Move the Nested PTZ Camera Feed

Click the PTZ camera video windows and drag it to any area on your monitor. This includes outside the Visualizer.

Track Objects through PTZ Camera

There are multiple setting options for tracking objects with your PTZ Cameras.

- **Rule based PTZ Camera** control configured for track type in a zone with specific start and stop actions. See *Add PTZ Camera Automatic Tracking Rule* (148).
    - **Rule based PTZ Time Switching** configured to switch between tracking of objects over time. This rule overrides Rule-based Zone priorities. See *Add PTZ Camera Time Switching Rule* (150).
- **Manual tracking** – Only available in LIVE mode, allows you to select an object to monitor as long as it remains in the QORTEX DTC server coverage area. This overrides Rule-based tracking. See *Manually Track Objects through the PTZ Camera* (138).

Manually Track Objects through the PTZ Camera

Use supported PTZ cameras to identify and track manually-selected objects through a zone. See *Establishing Zones and* Counter Lines (page 105).

Manually tracking objects overrides the automatic object tracking. See *Add PTZ Camera* (page 148).

To manually track an object from the PTZ camera:

1. Before you can manually track objects using the PTZ camera:
    - Switch to **LIVE** mode. See *Monitor Mode* (page 114)
    - Ensure the PTZ camera configuration is **Locked** 🔒.

      From **Configuration** panel **> PTZ** tab, click the unlocked icon in the **PTZ Camera** tab to toggle it to locked, if needed.
2. Select an object to track. See *Figure 81. PTZ Camera Select Object*.

   In the Visualizer, right-click the PTZ camera icon. A list of trackable objects is displayed. Select the checkbox for the track ID number of the object to track.

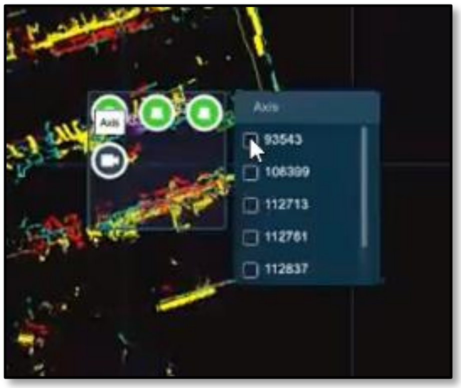


Figure 81. PTZ Camera Select Object

The PTZ camera video panel opens and displays the object. The Visualizer displays a line attached to the object that traces the movement of the numbered object. See *Figure 82. PTZ Camera Track Object*.

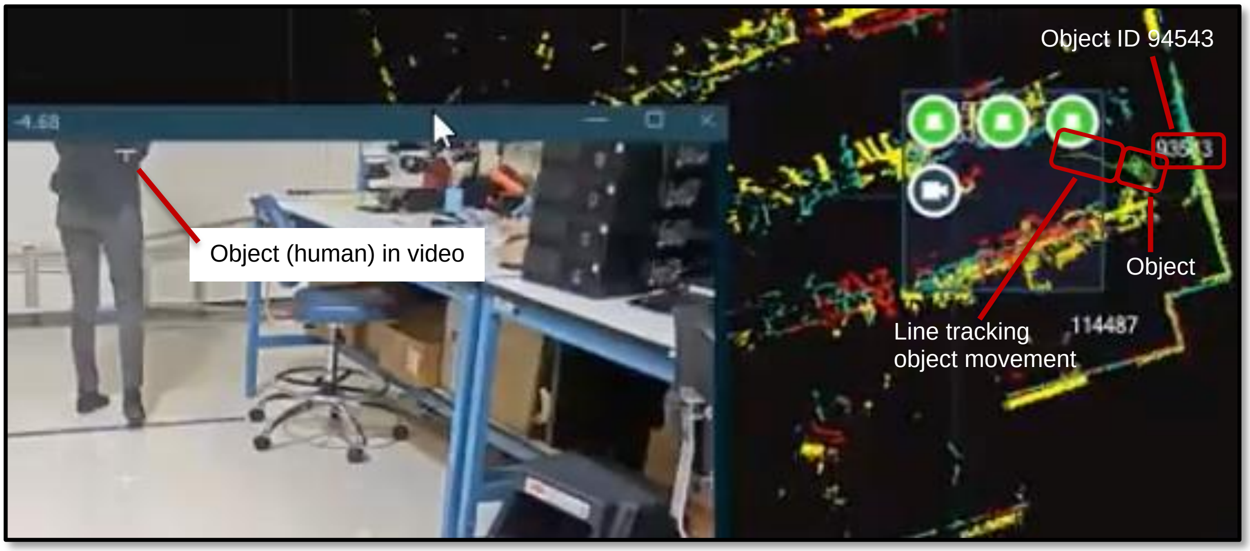


Figure 82. PTZ Camera Track Object

- The PTZ camera continuously tracks the object through the defined zone until the object disappears or the PTZ camera is manually released.
- When the object disappears, it loses its track ID number.
- If the object returns and becomes visible again, the QORTEX DTC server assigns it a new track ID. The PTZ camera does not continue to follow the object because it now has a different track ID.

3. Optionally, record the tracking of the object. See *Manually Start a Recording* (page 155).

   Recordings do not include the PTZ camera video tracking.

View PTZ Camera Details

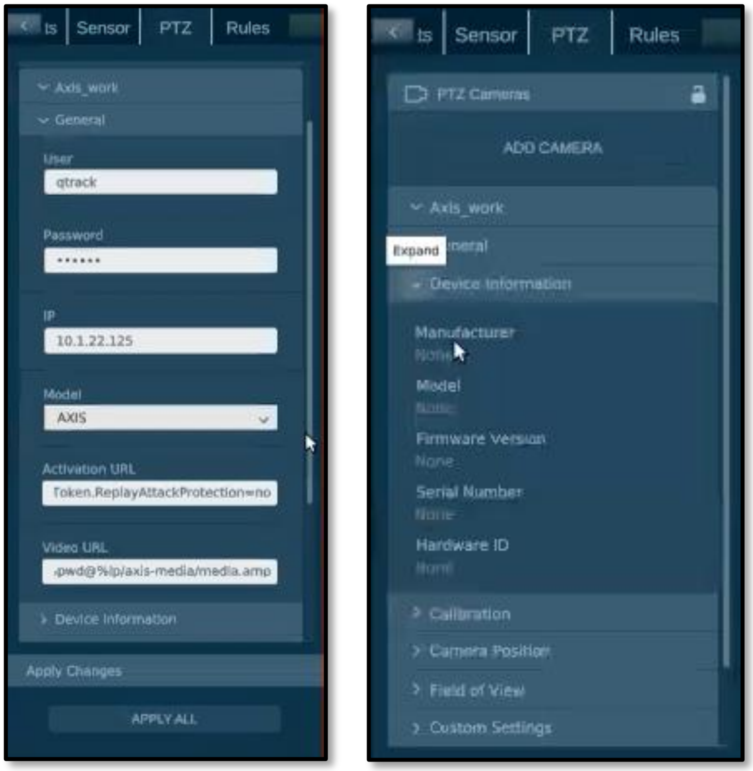
For each configured PTZ camera data through the **PTZ >** ***camera_name*** **> General** and **Device Information** tabs. See *Figure 83. PTZ Camera General and Device Information Panels*.

The **General** information panel lists the information provided when you added the PTZ camera to the QORTEX DTC server configuration. The fields can be edited. The fields are:

- **Name** field—assign a name for the PTZ camera.
- **User** and **Password** fields—enter the credentials required to access the PTZ camera.
- **IP** field—enter the IP address for access to the PTZ camera. Every PTZ camera must have its own IP address. QORTEX DTC supports both IPv4 and IPv6 IP addresses.
- **Model** menu—select the manufacturer for the PTZ camera displayed in the list. Select **UNKNOWN**, if your PTZ camera is not listed.
- **Activation URL**—is not required. Used for enabling the camera to handle positioning requests with OnVIF.
- **Video URL**—is automatically assigned. Used for the Real Time Streaming Protocol (RTSP) video stream. If the **Model** is listed as **UNKNOWN**, the URL is what you manually entered for your PTZ camera.

The **Device Information** is specific to the PTZ camera manufacturer. The **Device Information** panel is populated automatically when the PTZ camera is activated. To modify this data, use the gRPC API. See Appendix 2. *API for Remote Procedure Calls* (page 184). The self-explanatory **Device Information** fields are:

- Manufacturer
- Model
- Firmware Version
- Serial Number
- Hardware ID



User
qtrack

Password
••••••

IP
10.1.22.125

Model
AXIS

Activation URL
Token.ReplayAttackProtection=no

Video URL
pwd@%ip/axis-media/media.amp

Manufacturer
None

Model
None

Firmware Version
None

Serial Number
None

Hardware ID
None

Figure 83. PTZ Camera General and Device Information Panels

15. Adding Rules for Event Zones

Rules are applied to Event zones. Rules are not applied to Exclusion or Inclusion Zones/Lines. Rules are triggered by trackable objects (trackables) entering or leaving zones.

To create a rule, first create a zone. Each zone can have multiple rules with different rule types. See *Establishing Zones and* Counter Lines (page 105).

Event Zone Rules – Provide a method for processing objects in Event zones. Only one rule of each type can be applied to each Event zone. The Event Zone Rule Types are: **PTZ Camera**, **Network Action**, **Recording**, and **Object Stitching**. See *Figure 84. Add Rules Panel Options*.

- **PTZ Camera rules** – For automatic rule-based tracking with the PTZ camera, define zone stop and start triggers. These include objects as they enter, leave, or currently exist in an Event zone. See *Add PTZ Camera Automatic Tracking Rule* (page 148).
  - **Time Switching** – Enabling time switching between objects is an optional configuration. As of QORTEX DTC 2.3, PTZ Time Switching rules define how many objects to track, rotating tracked object every specified number of seconds. When this option is applied, the default PTZ camera assignment based on rules priority order is not followed. See *Add PTZ Camera Time Switching Rule* (page 150).
- **Network Action rules** – Define a network command to run when objects enter and exit a zone. See *Add Network Actions Rule* (page 143).
- **Recording rules** – Automatically record collected data on an object while it is in a zone. See *Add Recording Rule* (page 146).

> **Note:** In QORTEX DTC 2.1 or later automatic recordings are configured through the **Rules** tab. In QORTEX DTC 2.0 or older, automatic recording is configured through the **Events** tab.

- **Object Stitching rules** – Detect and merge duplicate trackable IDs. See *Add Object Stitching Rule* (page 145).

If the **Rules** tab is grayed out, verify and update licenses. Licensing **Add-Ons** field must state **Rules**.

Start to Add Rules

1. From the QORTEX DTC Client, select the **Settings** button.
2. In the Configuration panel, scroll through the top tab to the **Rules** tab.
3. Click the **ADD** button and select a **Rule Type**.
4. Proceed to the section for the Rule Type: **PTZ Camera**, **Network Action**, or **Recording**.

Note: For any time-based rule, make sure the computer system clock is correctly configured. QORTEX DTC references the system clock for day and time information.

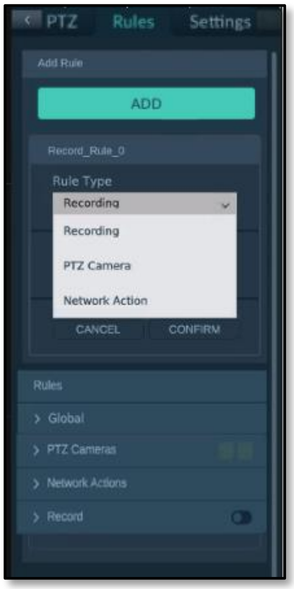


Figure 84. Add Rules Panel Options

Add Network Actions Rule

Network actions rules send notification messages to designated URL locations.

Prepare for Network Actions

To view the results of a Network Action, set up a server to listen for messages from QORTEX DTC. These messages can be simply displayed in a server terminal, or depending upon your server configuration, they can be added to a log or emailed.

Add Network Action Rules

1. From the **Configuration** mode panel > **Rules** tab, click **Add**. See *Figure 85. Network Action Add Rule Panel, Track Type, Schedule, and Log Response*.

   Rules trigger network commands when an object enters and exits a zone.

2. From the **Rule Type** menu, select **Network Action**.

   Rules trigger messages sent to specified URL when objects enter or exit a zone.

3. From the **Track type** menu, select one:

   - Any
   - Human
   - Vehicle

- Human and Vehicle
- Unknown

4. From the **Zone** menu, select from the list of Events zones you defined for this server.

   Create Event zones before adding a rule.

5. Specify when to activate the rule.

   For the Network Action, define an **Enter Trigger**, by typing one or more HTTP commands or HTTP command with digestive authentication in the text box. For example:

   Turn a light switch on when an object enters. Type `http://` with the string:

```
    <network-switch-ip>/outlet?<number>=ON
```

   Send a `$TRACK-ID` to log when object enters. Type `http://` with the string:

```
     <webserver-ip>:3000/$TRACK_ID-ENTRY
```

6. Specify when to stop the rule.

   For the Network Action, define an **Exit Trigger**, by typing one or more HTTP commands in the text box. For example:

   Turn a light switch off when an object enters. Type `http://` with the string:

```
    <network-switch-ip>/outlet?<number>=OFF
```

   Send a `$TRACK-ID` to log when object exits. Type `http://` with the string:

```
     <webserver-ip>:3000/$TRACK_ID-EXIT
```

7. If **Security** mode is setup, add the following lines to your HTTP command to include the digestive authentication information.

   Sample HTTP command with digestive authentication when Security mode is `OFF/Disabled`. Type `http://` with the string:

```
    <user>:<pwd>@<ip>/rcp.xml?command=0x01c1&type=F_FLAG&direction=WRITE&num=1&payl
oad=0x0
```

   Sample HTTP command with digestive authentication when Security mode is `ON/Enabled`. Type `http://` with the string:

```
    <user>:<pwd>@<ip>/rcp.xml?command=0x01c1&type=F_FLAG&direction=WRITE&num=1&payl
oad=0x0
```

8. Optionally, set a rule activation schedule. By default, rules are always on. You have the option to set a rule to run at a specific time of day for selected days of the week.

   a. Enable **Rule Schedule**, click the toggle ● to `ON`.

   b. Expand the **Schedule** panel, click the expand icon ❯.

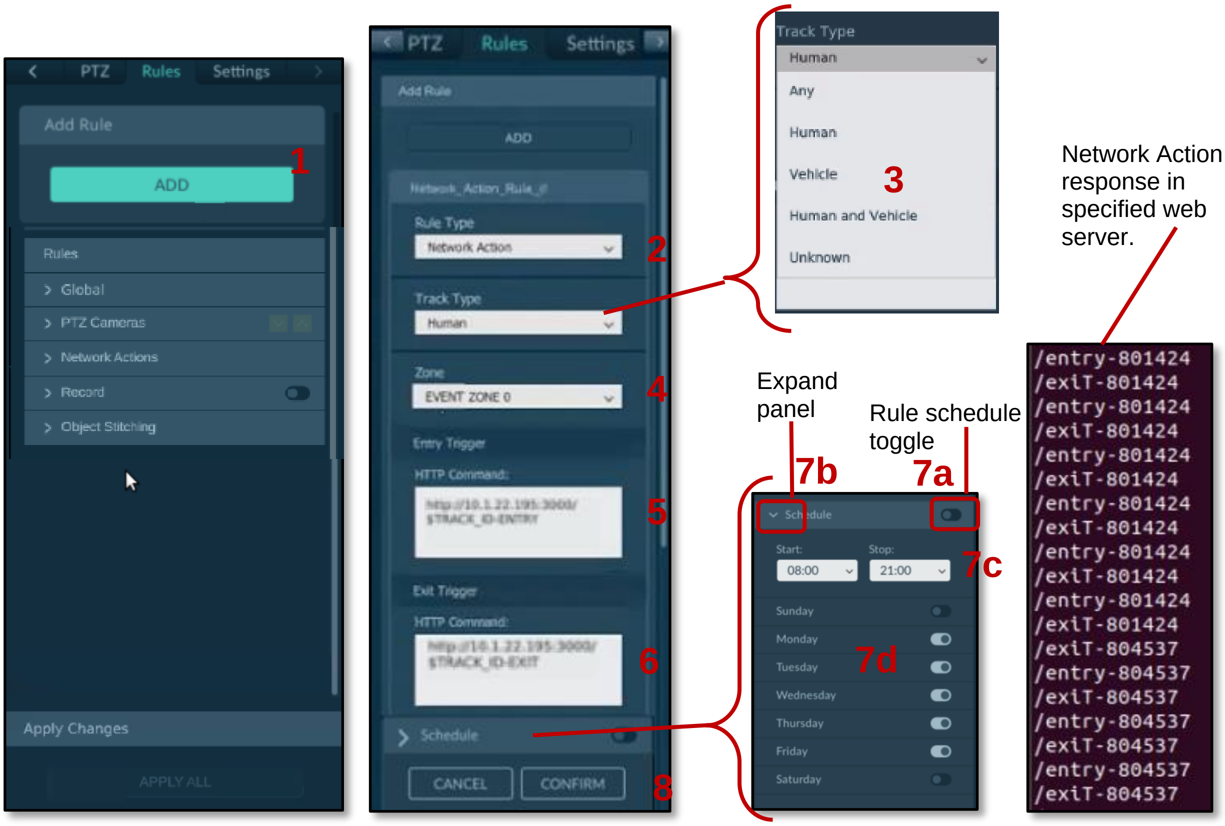
c. Select a time range for each day the rule is active, click the expand icon ⌄ and scroll through to select a time. Select both a **Start** and **Stop** time.

d. Select the days of the week the rule is active, click each day toggle ●, as needed.

9. Click **CONFIRM**.

   The new rule is added to the **Rule** tab panel under its **Rule Type**.

10. Click **Apply Changes** and if done with CONFIGURATION mode, click **EXIT CONFIGURATION**.



Rule Type: Network Action

Track Type: Human

Zone: EVENT ZONE 0

HTTP Command: http://10.1.22.195:3000/ $TRACK_ID-ENTRY

HTTP Command: http://10.1.22.195:3000/ $TRACK_ID-EXIT

Start: 08:00

Stop: 21:00

```
/entry-801424
/exiT-801424
/entry-801424
/exiT-801424
/entry-801424
/exiT-801424
/entry-801424
/exiT-801424
/entry-801424
/exiT-801424
/entry-801424
/exiT-801424
/exiT-804537
/entry-804537
/exiT-804537
/entry-804537
/exiT-804537
/entry-804537
/exiT-804537
```

Figure 85. Network Action Add Rule Panel, Track Type, Schedule, and Log Response

Add Object Stitching Rule

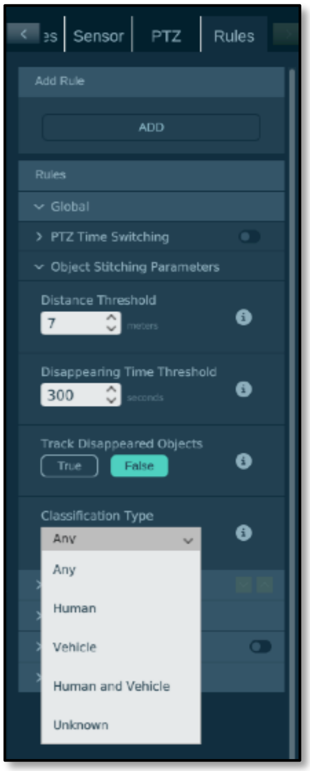
Object stitching detects and merges duplicate trackable IDs (multiple IDs for the same object). When an object disappears from a zone, then reappears, it is evaluated to determine if it is the same object. If it is the same object, it is assigned the same object ID. Create a zone to apply object stitching.

1. From the **Configuration** panel > **Rules** tab. Expand **Global > Object Stitching Parameters** panel. See *Figure 86. Rules > Global > Object Stitching Parameters Panel*.

2. Select the parameters.

- **Distance Threshold –** Set the distance within which QORTEX DTC should search for the duplicate or disappeared object.
- **Disappearing Time Threshold –** Set the time threshold to remember a duplicate or disappeared object.
- **Classification Type –** Select the class of object to track and stitch. The options are Any, Human, Vehicle, Human and Vehicle, Unknown. If you have licensing for vehicle subcategories, those are selectable in the pull-down menu.

3. Turn on Object Stitching. Click **True** in the **Track Disappeared Objects** field.



Distance Threshold: 7 meters

Disappearing Time Threshold: 300 seconds

Track Disappeared Objects: True [✓ False]

Classification Type: Any

Figure 86. Rules > Global > Object Stitching Parameters Panel

Object Stitching Settings Parameters

Object Stitching parameters can be edited through the Qortex DTC client or through gRCP messaging.

Add Recording Rule

1. From the **Configuration** panel > **Rules** tab, click **Add**. See *Figure 87. Record Add Rule Panel*.

   Rules trigger tracking objects when an object enters a zone.

2. From the **Rule Type** menu, select **Recording**.

Record is a toggle switch to enable automatically recoding the collected data per defined rules applied to zones.

3. From the **Zone** menu, select from the list of zones you defined for this server.

   Create zones before adding a rule.

4. Optionally, set a rule activation schedule. By default, rules are always on. You have the option to set a rule to run at a specific time of day for selected days of the week.

   a. Enable **Rule Schedule**, click the toggle ● to `ON`.

   b. Expand the **Schedule** panel, click the expand icon >.

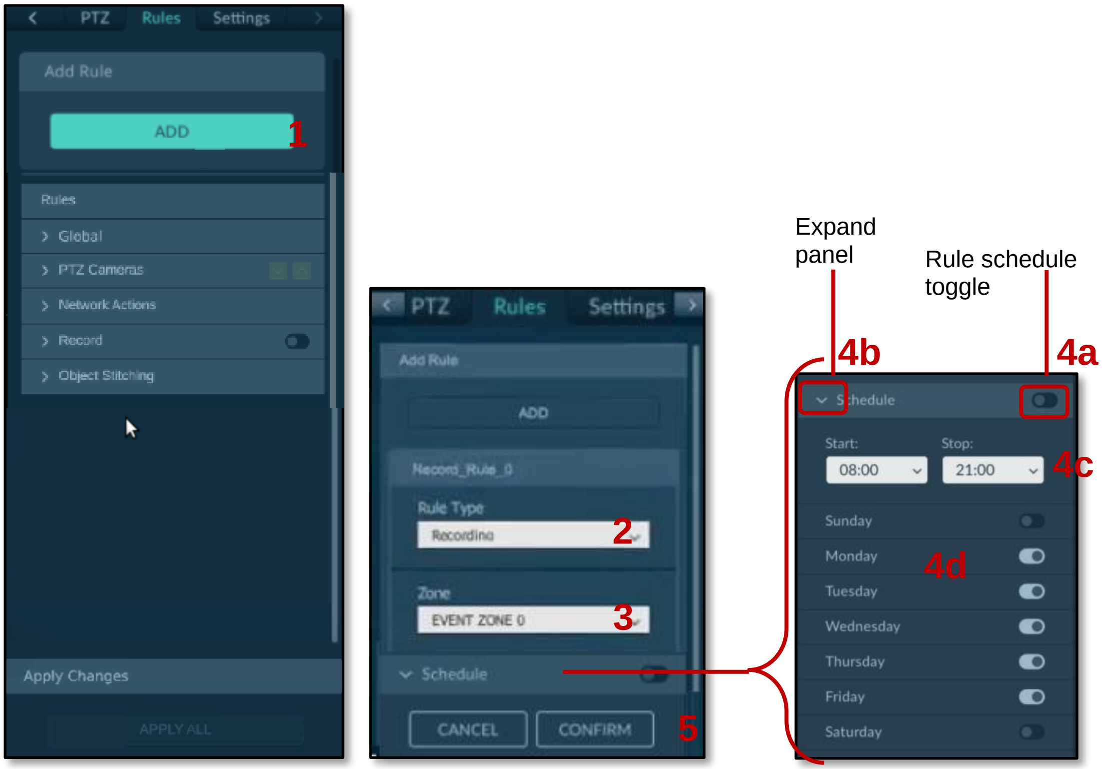
   c. Select a time range for each day the rule is active, click the expand icon ⌄ and scroll through to select a time. Select both a **Start** and **Stop** time.

   d. Select the days of the week the rule is active, click each day toggle ●, as needed.

5. Click **CONFIRM**.

   The new rule is added to the **Rule** tab panel under its **Rule Type**.

6. Click **Apply Changes**. If done with CONFIGURATION, click **EXIT CONFIGURATION**.



Rule Type: Recording

Zone: EVENT ZONE 0

Start: 08:00

Stop: 21:00

Figure 87. Record Add Rule Panel

Add PTZ Camera Automatic Tracking Rule

Use supported PTZ cameras to identify and track objects through a zone. See *Establishing Zones and* Counter Lines (page 105).

If there are multiple tracked objects QORTEX DTC defaults to automatic tracking, when the following criteria are met:

- PTZ Time Switching is OFF.
- Each tracked object is followed by one PTZ camera.
- PTZ cameras are assigned to track the objects based on:
    - **Rule priorities.** You can automatically track objects using Rules with your PTZ camera in a selected Event zone. Rules apply to all PTZ Cameras. See *Sort Rule Priority* (page 151).
    - **Proximity**. The PTZ camera with the FOV that is closest to each tracked object.

You can manually track an object during LIVE mode. See *Manually Track Objects through the PTZ Camera* (page 138).

To setup automatically tracking objects using Rules with your PTZ camera in a selected Event zone:

1. From the **Configuration** mode panel > **Rules** tab, click **Add**. See *Figure 88. PTZ Camera Add Rules Panel*.

   Rules trigger tracking objects when an object enters a zone.

2. From the **Rule Type** menu, select PTZ Camera.

   PTZ rules do not apply for Playback mode, they only apply to Live mode.

3. From the **Track type** menu, select one:
    - Any
    - Human
    - Vehicle
    - Human and Vehicle
    - Unknown

4. From the **Zone** menu, select from the list of Event zones you defined for this server.

   Create Event zones before adding a rule.

5. Select when to activate the rule. Currently, there is only one option.

   Enter/Occupied

6. Select when to stop the rule. Select from the **Stop** menu:
    - **Object disappears**
    - **Exits the zone**

- ○ **Triggered by a new object**

7. Optionally, set a rule activation schedule. By default, rules are always on. You have the option to set a rule to run at a specific time of day for selected days of the week
   a. Enable **Rule Schedule**, click the toggle to ON.
   b. Expand the **Schedule** panel, click the expand icon ›.
   c. Select a time range for each day the rule is active, click the expand icon ⌄ and scroll through to select a time. Select both a **Start** and **Stop** time.
   d. Select the days of the week the rule is active, click each day toggle, as needed.
8. Click **CONFIRM**.

   The new rule is added to the **Rule** tab panel under its **Rule Type**.

   During LIVE mode, when an object of the selected type enters the designated Event zone, and if the PTZ camera is available, then QORTEX DTC automatically tracks the object. This tracking is included in the collected data.

   When the object leaves the Event zone, and there are no other objects to track, the PTZ camera returns to the configured Home location. See *Set PTZ Camera Home Location* (page 135).
9. Click **Apply All** and if done with CONFIGURATION mode, click **EXIT CONFIGURATION**.

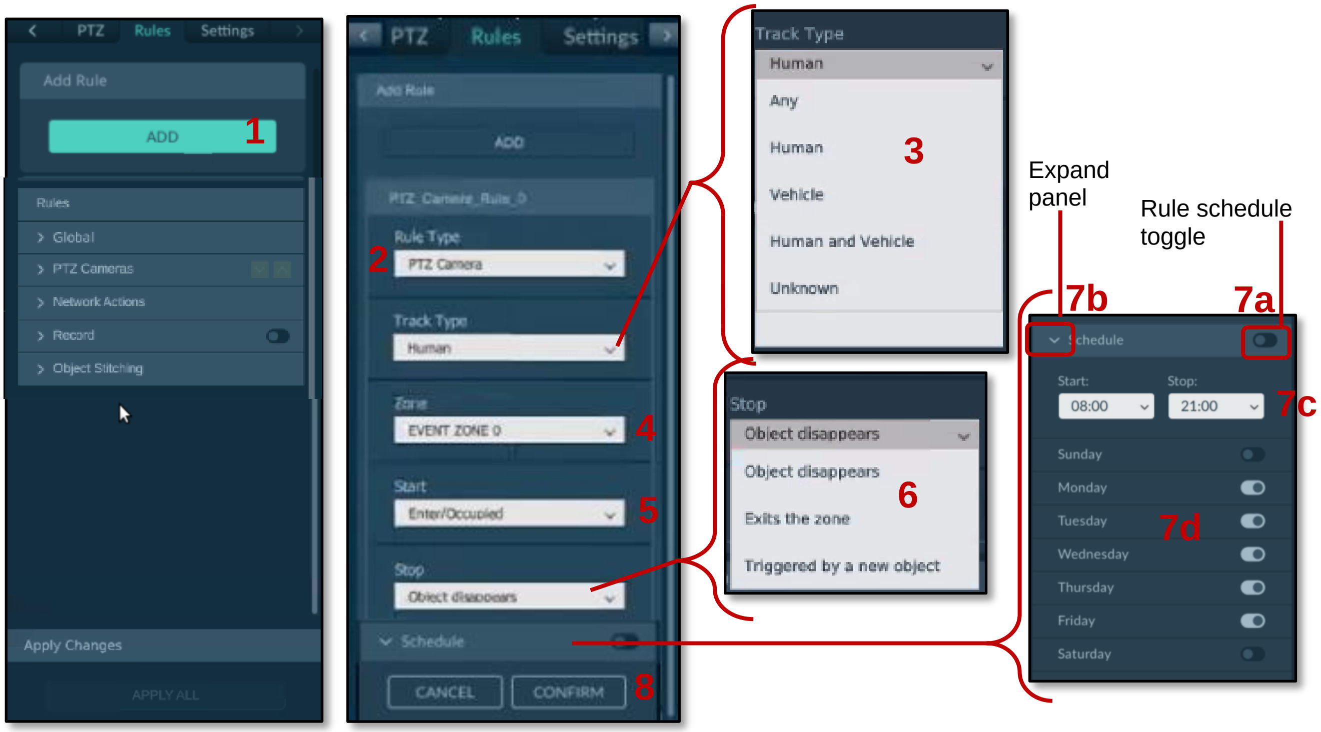


Figure 88. PTZ Camera Add Rules Panel

Add PTZ Camera Time Switching Rule

When more than one trackable object is in a PTZ camera view, the object being tracked is typically the most recent object. To continue tracking older objects, enable PTZ Time Switching to alternate between tracking the Y number of triggered objects every few X number of seconds. These values are configurable, see the steps below.

For example. If **Alternate Between Last** (value Y) is 3 (Object 1, Object 2, Object 3) and **Follow Time Per Object** (value X) is 10. Then the PTZ camera video window tracks Object 3 for 10 seconds, switches and the tracks Object 2 for 10 seconds, switches and the tracks Object 1 for 10 seconds, and returns to Object 3 to start the cycle again. This repeats until one or more of the tracked objects leaves the monitored area or new trackable object enters the monitored area.

To set the number and frequency of switching between tracked objects. See *Figure 89. PTZ Time Switching Settings*.

1. Prepare to define the PTZ camera Field of View (FOV).

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).

2. From the **Configuration** panel > **Rules** tab, expand the **Global** panel.

3. Enable the **PTZ Time Switching** toggle and expand the **PTZ Time Switching** panel.

   OFF (black): PTZ camera tracking behavior is the same a QORTEX DTC 2.1, where the PTZ camera continues to track one object at a time.

   ON (green): PTZ camera tracking switches between objects based on the settings in Step 4.

4. Select the field values:

   - **Alternate Between Last.** Sets the number of objects to track in alternating rotation. The default is one object minimum. The maximum is 10 objects.
   - **Follow Time Per Object.** Sets the length of time to track an object before switching to the next object in rotation. The default is 10 seconds minimum. The maximum is 120 seconds.

   Click the ^/v arrows and make your selection.

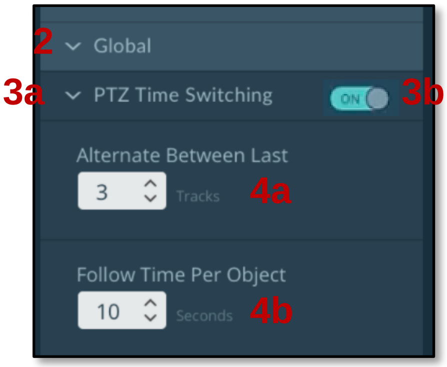


Figure 89. PTZ Time Switching Settings

Sort Rule Priority

When there are multiple PTZ Camera rules, the sort icon **^/v** is displayed on the PTZ Camera tab. Click the icon to sort the rules numerically, lowest to highest or highest to lowest. The numerical value is the incremented portion of the rule name. See *Figure 90. Sort Rule Priority*.

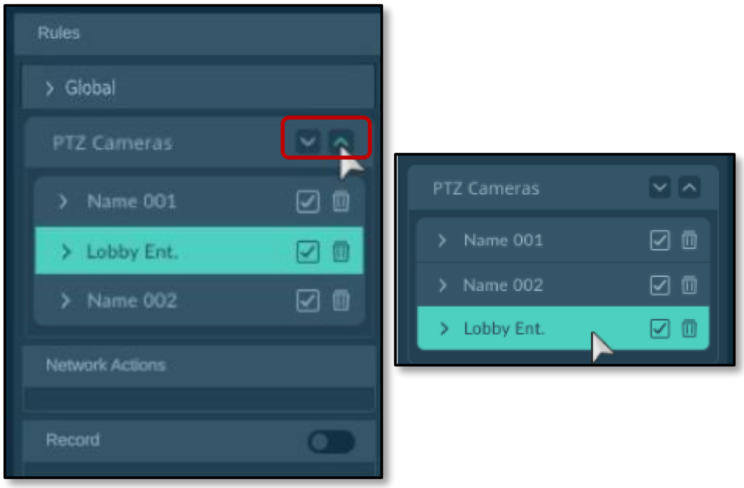


Figure 90. Sort Rule Priority

Disable a Rule

1. From the **Configuration** mode panel > **Rules** tab, expand › the **Rules** panel.

   Rules are enabled by default, as indicated by the checkbox icon ☑.

2. Click the **checkbox** icon off □ for the rule you want to disable.
3. Click **Apply All**.
4. Click **Apply Changes** and if done with CONFIGURATION mode, click **EXIT CONFIGURATION**.

Delete a Rule

1. From the **Configuration** mode panel > **Rules** tab, expand › the **Rules** panel.
2. Click the **trashcan** 🗑 icon for the rule you want to delete.

> **Note:** There is no confirmation for deleting a rule. If you click the trashcan icon, the rule is deleted.

16. Recording Data

There are two modes for recording the data collected at a location: manual and rule based. Both modes save recordings in Quanergy proprietary Qlog format. Recordings can be replayed in the QORTEX DTC client (or other Quanergy application). This section discusses how to record in both modes and play back the recording.

Stored Recording Data

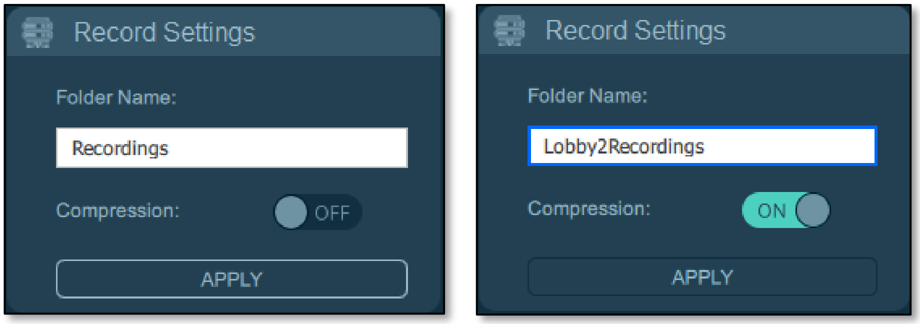
Recordings are stored as follows:

- In the datasets directory. See *Start the Server Daemon Service the First Time* (page 56).
- In the specified directory. See *Specify Recording Settings* (page 153).

Specify Recording Settings

Recording settings have default values and do not need to be adjusted. However, you can make adjustments to optimize for a particular computing environment. There are two simple adjustment options, which affect both modes of recording.

- **Compression**—Used when either bandwidth or storage space in limited.
- **Folder Name**—Use to change the default recording folder name. The folder contains:
  - A Recordings folder name is created for each recording. To ensure a unique folder name a Universal Time Coordinated (UTC) time is part of the Recordings folder name. The time used aligns with when the logging it started.
  - At least one sequentially numbered index file, such as `SensorName.i00`, for every sensor at the location.
  - At least one sequentially numbered QLog file, such as `SensorName.q00`, for every sensor at the location.
  - A copy of the `settings.xml` file that was in force during the recording.



Record Settings

Folder Name: Recordings

Compression: OFF

APPLY

Record Settings

Folder Name: Lobby2Recordings

Compression: ON

APPLY

Figure 91. Client Record Settings Fields: Default (left), Optimized (right)

1. From the **Configuration** panel > **Server** tab, expand the **Record Settings** tab.
2. In the **Folder Name** field, the default name (**Recordings**) can be edited. See *Figure 91. Client Record Settings Fields: Default (left), Optimized (right)*.

   If you want to change the name, the new name must be made up of numbers, letters, and underscores.

   Recordings are saved in the data sets directory. Recording folder names are automatically created for each recording. To ensure a unique folder name a UTC time is part of the Recordings folder name. The time used aligns with when the logging it started. See *Start the Server Daemon Service the First Time* (page 56).
3. The **Compression** switch (enabled only in **LIVE mode**) can be switched on or off, depending on how much bandwidth and storage space is available.
4. If any changes are made, click the **APPLY** button to save them, update the configuration file on the server, and implement them.
5. Continue the setup process with one or both of the following procedures:
   - *Manually Start a Recording* (page 155).
   - *Add Recording Rule* (page 146).

The recording is automatically saved in the server host computer directory created when the server software was first installed. The default path is:

```
/home/quanergy/quanergy/qortex/datasets/Recordings/YYYY-MM-DD_HH-MM-
SS.xxx.
```

Add a Rule-Based Recording

Event zones (not Exclusion or Inclusion zones) have the option of triggering a recording event based on the actions of a moving object that is tracked, that is, a trackable. If preferred, take the following steps to enable rule-based recording. These options are not changeable when a recording or playback event is in progress. See *Figure 92. Record Toggle on Rules Tab*.

1. Prepare to add a rule-based recording.

   Connect your QORTEX DTC client to a QORTEX DTC server and enter **Configuration** mode. See *Connect the Client to the Server* (page 61) and *Enter Configuration Mode* (page 85).
2. From the **Configuration** panel > the **Rules** tab, expand the **Rules** panel to view the **Record** tab. Toggle the **Record** switch so that it is green and indicates **ON**.

   ON disables the **Record** button in the top right corner of the client window. This allows **Event zone** rule-based recording settings to execute.

   OFF disables the automatic recording function for all **Event zones** and enables the manual **Record** button. See *Manually Start a Recording* (page 155).

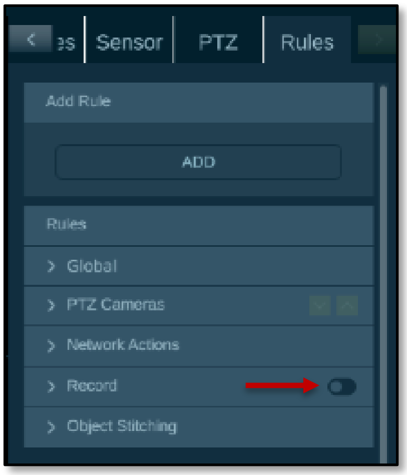


Figure 92. Record Toggle on Rules Tab

3. A trigger, defined by the rules created, start a recording.

   During a rule-based recording event, the **Record** button (top right of the client) is disabled and the recording is complete.

   When the recording event ends, the **Record** button returns to grayed out, the **timer** disappears, and the event is saved as a separate QLog.

   By default, the recording has a buffer time of 60 seconds. This buffer is the amount of time that is retroactively recorded before a **start_record** action, as well as the amount of time that continues recording after a **stop_record** action. So, if a trackable object triggers a zone for 30 seconds, the recording will be 150 (60+30+60) seconds long.

4. To replay rule-based recorded data, see *View Recorded (PLAYBACK) Data* (page 156).

Manually Start a Recording

When LIVE data is visualized, a gray (inactive) **Record** button appears (top right of the client), enabling the start and stop of data recording with a mouse click.

Recordings do not include PTZ camera video tracking.

1. From the **Configuration** panel **> Rules** tab **> Rules** panel, select **Recording** switch.

2. Set the **Rules Recording** switch to `OFF`. See *Figure 92. Record Toggle on Rules Tab*.

   `OFF` disables the automatic recording function for all Event zones and enables the manual Record button in the top right corner of the client window

3. Click the **Record** button to start recording.

   The Record button turns **red** and the time elapsed time counter appears. Leave the client open while recording is in progress if an accurate indication of elapsed time is needed. See *Figure 93. RECORD Button: Inactive and Active with Elapsed Time Counter*.

> **Note:** The elapsed time counter is accurate only if the client remains open while recording is in progress.

4. When satisfied with the data recording length, click the **Record** button to stop recording.

   The **Record** button returns to the gray (inactive) state, and the recording is saved as a QLog file.

5. Play back or download the recordings. See *Visualize Live Data* (page 122).

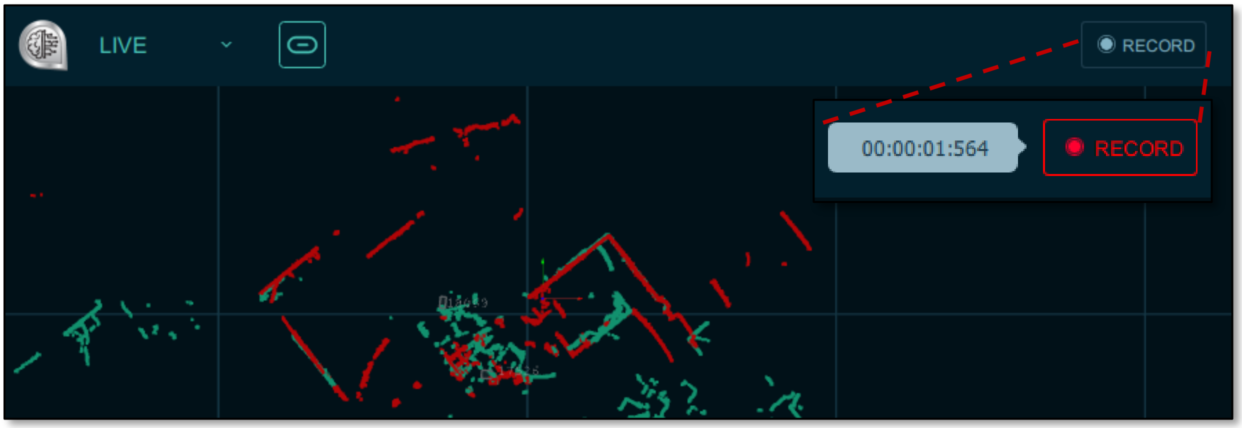


Figure 93. RECORD Button: Inactive and Active with Elapsed Time Counter

View Recorded (PLAYBACK) Data

All recording data is captured in recorded QLog files. To visualize (view) this data:

1. From **Configuration** mode, **Disconnect** from the sensors.

   Click **Connected** button to disconnect the server from the sensors' streams of LIVE data, if needed.

   PLAYBACK is available only when the sensors are disconnected.

2. In the same corner, click the **Data Selector** arrow, and select the **PLAYBACK** item.

3. Click the **Select Recording** button that appears, then select a recording from the menu. See *Figure 94. Top Navigation Bar: Selecting and Playing a Recorded Dataset*.

   The available options change, depending on what action is taken:

   - Click the **Play** arrow to see the recording. All the viewing controls still allow customization of the visualization.
   - Click the **Replay** arrow to see the same recording again. Viewing control choices made during the previous play persist upon replay.
   - Click the **Download** arrow to download the recording from the server to the local client host computer for sharing. An animated green circle indicates a busy status, and red indicates error status.
   - Click the circling **Download** arrow to cancel the downloading process.

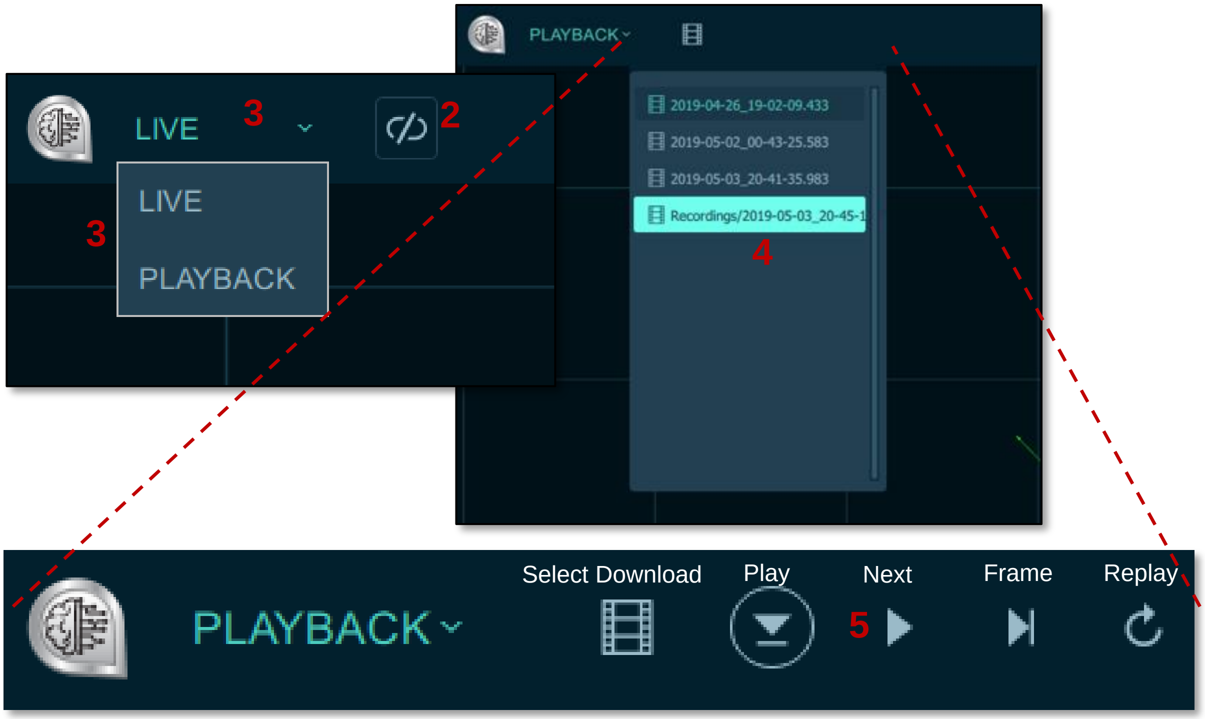


Figure 94. Top Navigation Bar: Selecting and Playing a Recorded Dataset

4. Click the **Play** button arrow. See *Figure 95. Top Navigation Bar: During Play in Playback Mode*.

   - The **Play** button turns blue.
   - The name of the selected dataset appears.
   - The elapsed time appears as a counter, which starts at 00:00:00:00 and increments up to the maximum time length of the recording.

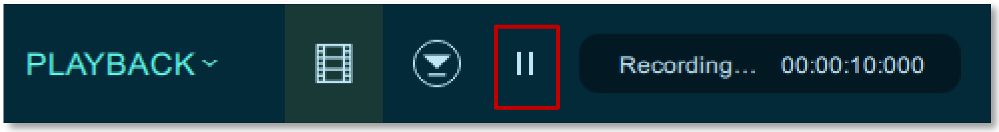


Figure 95. Top Navigation Bar: During Play in Playback Mode

5. Optionally, click to Pause the recording. See *Figure 96. Top Navigation Bar: During Pause in Playback Mode*.

   When playback is paused, the Play arrow becomes a Pause symbol, and the Next Frame and Replay arrows disappear.

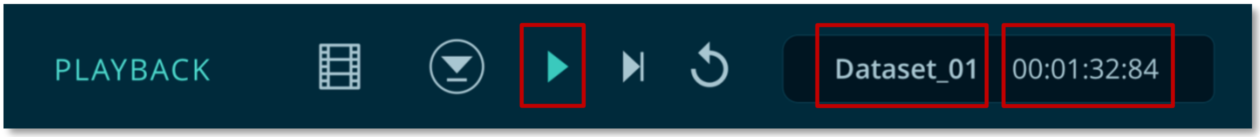


Figure 96. Top Navigation Bar: During Pause in Playback Mode

Visualize Playback Data in Loopback Mode

Loop the play of a dataset through the command line (not the user interface), as follows.

1. On the QORTEX DTC server host computer, open a terminal (or `ssh`) and log in. The username and password for the host machine may be required.
2. Execute the command to play a dataset in **Loopback** mode (continuous looping play), substituting the brown portions with the pathway and name of the settings file and recorded dataset:

```
$ cd /opt/quanergy/qortex-server
$ sudo ./Qortex-server -s /pathway/to/settings/file.xml
   --replay-path /pathway/to/dataset/folder -|
```

- Press the **spacebar** to pause the looping.
- Press the **Esc** key to stop the playback.

> **Note:** Help is available at the terminal to recall commands and options, as follows: `./Qortex-server -h`

Refresh Tracks

Use the **Sensor** tab to refresh the tracks. Refresh tracks when a lag in the display of new tracks is detected, or continuing noise is displayed as old track artifacts. See *Figure 97. Refresh Tracks and Optimize Sensor Settings with Sensor Tab*.

1. **Disconnect** from the sensors. Click **Connected** button to disconnect the server from the sensors' streams of LIVE data, if needed.
2. In the bottom right corner of the interface, click the **Settings** icon ⚙ to open the **Configuration** panel.
3. Enter **Configuration** mode. See *Entering Configuration Mode* (84).
4. From the **Configuration** panel, select the **Sensor** tab.
5. At the top of **Sensor** tab, click the **REFRESH** button.

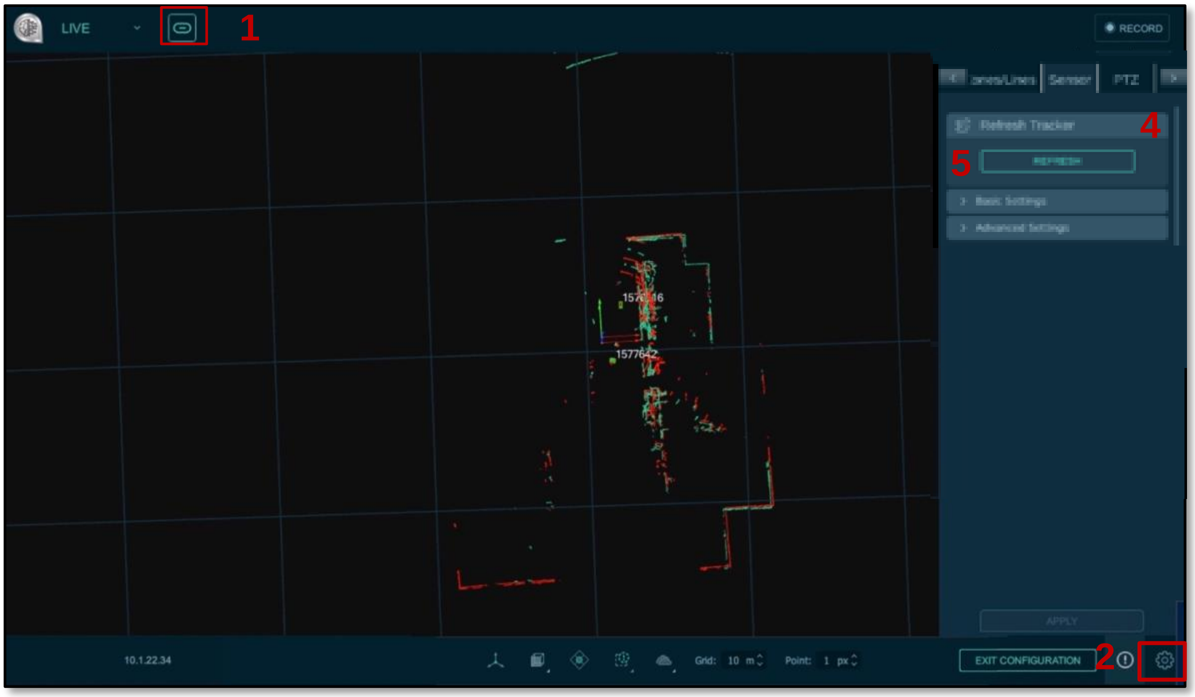


Figure 97. Refresh Tracks and Optimize Sensor Settings with Sensor Tab

Delete a Recording

When a dataset is recorded by running a daemon service, the owner of the recordings is Quanergy. When the user-based or trigger-based recordings on the hard drive are no longer needed, use `sudo` permission to overrule the restrictions and delete the files from the server host computer, as follows:

1. In a terminal, navigate to the directory enclosing the folder to be deleted.
2. Execute the data removal command, updating the folder name and providing the login password:

```
/home/quanergy/quanergy/qortex/datasets
$ sudo rm -rf Test_Recordings
```

17. Troubleshooting

Most problems have simple solutions that can be resolved as explained below.

General Help

For the server, the QORTEX DTC API provides some specific help commands and parameters that are useful. See *API Help Command* (page 182) and Appendix 2: *API for Remote Procedure Calls* (page 184).

Version Discovery

To identify which version of QORTEX DTC software is currently running:

For the server, on the Ubuntu terminal, look for the package by executing the following:

```
$ sudo dpkg -l qortex-server
```

Or, in the Ubuntu terminal, use the QORTEX DTC API command, as follows:

```
$ cd /opt/quanergy/qortex-server
$ ./Qortex-Server --version
```

For the client, when the QORTEX DTC client icon is double-clicked on the Windows desktop, the launch window and GUI window both list the version at the top of their respective windows.

Previous Software Removal

For the server, if previously installed software needs to be removed to install a new version or reinstall the same version:

1. Open a new Ubuntu terminal window on the computer hosting that software.
2. Execute the following command to remove the old software:

```
    $ sudo systemctl stop qortex-server.service
    $ sudo dpkg -r qortex-server
```

For the client, if previously installed software needs to be removed, right-click it in a file browser, select the Uninstall item, and select Yes to “allow this app to make changes.”

Quanergy Folder

For the server, the QORTEX DTC server automatically creates a `~/quanergy/` folder the first time it starts up. If the folder is deleted and does not exist, all attempts to select a filepath or save settings returns errors. To fix, restore the missing folder on the computer hosting the QORTEX DTC server, as follows:

1. In the Ubuntu terminal window, execute the following:

```
    $ sudo userdel quanergy
```

2. Reinstall the server. See *Install or Update QORTEX DTC with a Debian Package* (page 40).

Licensing Errors

For the server, the license management process may trigger errors, as discussed below.

Activation Error 5008

After logging in to the QORTEX DTC server and activating the license key. See *Managing the License* (page 45), the following appears:

```
Status: Activated
```

But if a wrong License ID or Activation Password is provided, the following appears:

```
Error 5008: Invalid Activation Data
```

This means that the License ID and/or Activation Password were not validated by the License Portal server. Make sure a valid License ID and Activation Password is available and are entered correctly for activation. See *Activate a License* (page 47).

Deactivation Error 9015

After deactivating a license, the following appears if a user tries to deactivate the previously deactivated license:

```
Return Code = 9015 Some (or all) of the arguments are invalid
```

Ignore the error; QORTEX DTC is already non-functional. When ready to reactivate the license and resume activity. See *Activate a License* (page 47).

Settings File Error

When you select an area (**LIVE** mode) or dataset (**PLAYBACK** mode), the QORTEX DTC server checks the relevant `settings.xml` file against the current license. The server stops and displays an error in the status bar if the number of sensor configuration sections in the `settings.xml` file is more than the number of sensors permitted by the license. If this error occurs, try the following:

- Refresh the license. See *Refresh a License* (page 49).
- Contact a support representative. See *Any Other Issue* (page 164).

Status and Error Messages

If the QORTEX DTC system has a suspected issue, check the following:

For the system **Status** panel, see *Status Button* (page 82). Status and error messages are listed below in *Table 18. System Status and Error Messages*.

For the sensor **Diagnostics** panel, see *Sensor Health Status Options* (page 103). A list of the sensor state messages is provided in *Table 19. Sensor State Messages*. For possible causes and solutions for the error reported, see *Troubleshooting Issues* in your sensor-specific user guide. Request the sensor user guide from support@quanergy.com.

The sensor health is also shown on the Q-View **Diagnostics** panel, discussed in the *Q-View User Guide* (available at https://downloads.quanergy.com/).

Table 18. System Status and Error Messages

Category	Severity	Error
Server State Error	**Red**	Request server state error
Sensor State Error	**Red**	Request sensor state error
Live Mode Error	**Red**	Stop live mode error
Playback Mode Error	**Red**	Start playback error
		Stop playback error
		Slow down playback error
		Step playback error
		Replay playback error
Zone Editing Error	**Red**	Cannot set existing name for the zone
QLog Error	**Red**	QLog download error
		QLog file list request error
Location Settings Error	**Red**	Sensor location settings request error
Location Error	**Red**	Start location error
Zone Error	**Red**	Adding a zone error
		Editing a zone error

Category	Severity	Error
		Removing a zone error
		Requesting the Event zone violation enabled state error
		Event zone violation enabled state error
		Changing specific zone violation trigger error
		While requesting specific zone violation trigger actions error
		While requesting specific zone violation trigger error
		While requesting zone enabled status for the zone type error
		While setting zone enabled status for the zone type error
		Changing specific zone violation trigger error
License Expiring	**Yellow**	Subscription license has 30 day(s) remaining
		Subscription license has 10 day(s) remaining
	Red	Subscription license has expired
		Subscription does not support the number of sensors connected
		Sensor disconnect info error

Table 19. Sensor State Messages

Severity	Error
Colorless	None
Red	Motor Initialization Error
Red	High Sensor Temperature
Red	Insufficient Motor Velocity
Red	Field Of View Is Blocked (MQ-8 sensors only)
Red	Version Mismatch
Red	Watchdog Violation

Software Crash

The QORTEX DTC application is resilient by design, but software crashes can occur.

For the server, the client will detect the server crash and send a notification. The server is designed to run persistently, so it will restart its service automatically.

1. If it fails to restart, see *Control the Server Daemon Service* (page 57). Check the status of the service, and if a process number is not given, re-start the service. Check status again to confirm that the return includes a process number.
2. If necessary, notify a support representative.

For the client, do the following:

1. Update all of the available graphics drivers on the host machine to rule out that potential issue.
2. Restart the client. See *Start the Client on a Windows Host* (page 58) or *Start the Client on an Ubuntu Host* (page 59). The client remembers the previously saved configuration and resumes in **Monitor** mode.
3. If necessary, notify a support representative.

Any Other Issue

Contact a support representative to create an automatic support ticket to receive the specific help needed. Provide the sensor serial number, if required.

- If the hardware and software was purchased from Quanergy, send an email to support@quanergy.com with feedback, questions, or concerns.
- If the hardware and software was purchased from a value-added reseller (VAR) or system integrator (SI), contact them for support.

18. Logs and Configuration Files

User Activity Log

The user log file is useful for auditing purposes. The user log file contains QORTEX DTC client IP, username, and login and logout times. The login time is when the user is successfully authenticated with the QORTEX DTC server. Logout time is when the user disconnects from the QORTEX DTC server. See *Figure 98. Security: User Activity Log* and *Table 20. User Log Statements*.

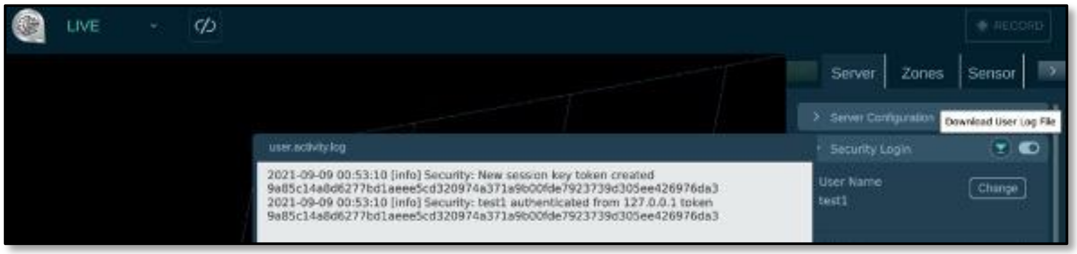
Data is stored in the user log whether the QORTEX DTC client is in LIVE or PLAYBACK mode. The option to download the log is only available in LIVE mode. The user log continues to grow as long as the QORTEX DTC server is running. Each time the QORTEX DTC server restarts, a new user log is created and the old user log is overwritten.

Log content includes: the user log file accessible from the QORTEX DTC client contains user's IP address, username, permission level, login and logout times. In addition, the file contains security configuration updates/changes such as adding new users, changing user permissions, resetting passwords, when password expiry is enabled/disabled, error when adding an existing user, and admin username change.

Download User Log Files

1. Select login option:
   - To modify the log, login as `Admin` or `root` user.
   - To only view the log, login as `Viewer` or `Editor` user.
2. Select the **Configuration** mode panel > **Server** tab > **Security Login** panel header.
3. Click the download icon.

   The current log is displayed in a new terminal and a userLogs.zip of the current files are written to: `/home/quanergy/quanergy/qortex/userLogs/user.activity.log`.



```
2021-09-09 00:53:10 [info] Security: New session key token created
9a85c14a8d6277bd1aeee5cd320974a371a9b00fde7923739d305ee426976da3
2021-09-09 00:53:10 [info] Security: test1 authenticated from 127.0.0.1 token
9a85c14a8d6277bd1aeee5cd320974a371a9b00fde7923739d305ee426976da3
```

User Name
test1

Figure 98. Security: User Activity Log

QORTEX DTC Server implements User Activity Log to track user-initiated actions explicitly related to security changes or approved/denied by Cyber Security. Download the security user activity log. This file is read only except for Quanergy or root users.

Table 20. User Log Statements

Log level	Log statement type	Text format example
Info	Security Token Created	2022-06-24 22:58:57 [info] Security: New session key token created
info	Client authenticated	2022-06-24 22:58:57 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 authenticated
info	Client authentication rejected	2022-06-24 23:12:24 [info] Security: ::ffff:127.0.0.1 @Alex434 auth rejected
info	Client session started	2022-06-24 22:58:57 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 started new session
info	Client session finished	2022-06-24 23:04:45 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 finished session
info	Set Credentials	2022-06-24 23:03:23 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 set user name @Alex435 and password new user type Viewer
info	TCP port connected	2022-06-24 23:17:54 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 ::ffff:127.0.0.1 connected to port 17175 secure mode
info	TCP port N unsecured	2022-06-24 23:17:54 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 TCP 17175 set unsecured
info	TCP port N secured	2022-06-24 23:17:54 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 TCP 17175 set secured: must reconnect all
Info	Setting edited	2022-06-24 23:00:39 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 edited settings set expiryEnabled to false set secureTcp to true Message can list different settings changed.
Info	Set user name/ password/ user type	2022-06-24 23:01:59 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 set user name @Alex434 and password new user type Viewer
Info	Deleted user	2022-06-24 23:04:10 [info] Security: Admin user @Alex433 ::ffff:127.0.0.1 deleted user @Alex434

Log File Rotation

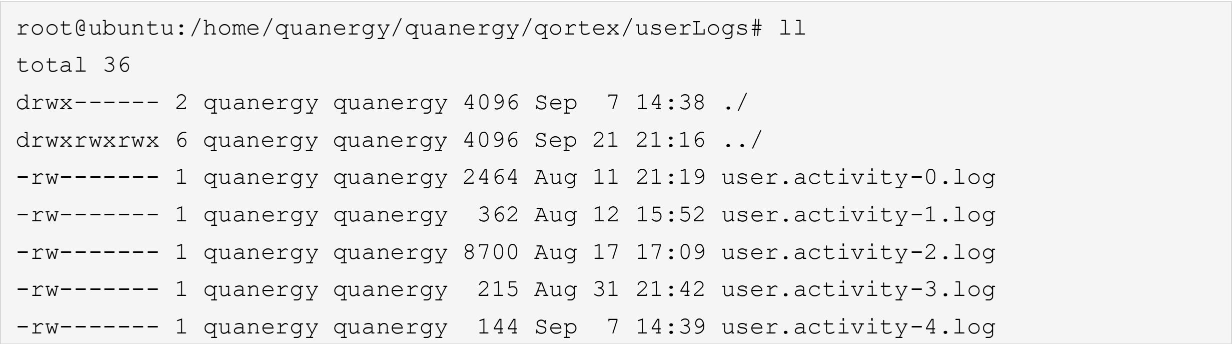
By default message logging is sent to the QORTEX DTC console when the QORTEX DTC server is unable to find a properties file, `qortex.server.logging.properties`. Default location for the properties file is `/home/quanergy/quanergy/qortex/`. The properties file provides multiple options for logging messages to a log file.

Log file rotation has two primary features:

- After a configured state is reached, the primary log file content is transferred to a new, incremented, file. The primary log file continues to receive messages.
- After a configured state is reached, the oldest log file is removed to make room for the newest incremented log file.

Define the log file rotation conditions through the properties file.

- The rolling log files directory is: `/home/quanergy/quanergy/qortex/userLogs/`
- Rotation time: every 24 hours
- Max log file count allowed: 100



```
root@ubuntu:/home/quanergy/quanergy/qortex/userLogs# ll
total 36
drwx------ 2 quanergy quanergy 4096 Sep  7 14:38 ./
drwxrwxrwx 6 quanergy quanergy 4096 Sep 21 21:16 ../
-rw------- 1 quanergy quanergy 2464 Aug 11 21:19 user.activity-0.log
-rw------- 1 quanergy quanergy  362 Aug 12 15:52 user.activity-1.log
-rw------- 1 quanergy quanergy 8700 Aug 17 17:09 user.activity-2.log
-rw------- 1 quanergy quanergy  215 Aug 31 21:42 user.activity-3.log
-rw------- 1 quanergy quanergy  144 Sep  7 14:39 user.activity-4.log
```

Figure 99. Sample Rolling Log File Directory

System File

The `settings.xml` file contains the list of configuration settings applied to each sensor. Select and change settings through the QORTEX DTC client options listed on tabs in the **CONFIGURATION** mode panel. Alternatively, you can edit the `settings.xml` file in a text editor.

Properties File

The properties file is a text file with the name `qortex.server.logging.properties`. The default location is in the `/home/quanergy/quanergy/qortex/` directory. If the file is not in this locatio when the QORTEX DTC server is launched, logging configuration is not uploaded.

Logging Properties File Parameters

logToFile

Set whether to log message to a file. This is a flag. Default is `False`.

- `True` Logs messages to a file.
- `False` Logs message only to the Qortex DTC console.

logFile

Define the path to the messages log file. Action is dependent on `logStrategy` setting. If the `logStrategy` setting is:

- `Rolling` The log file directory cannot contain any other files.
- `syslog` The `logFile` is redundant because messages are logged to `/var/log/syslog`.
- `logger` The `logFile` path already exists and that the QORTEX DTC server user has permission to edit the file.

fileLogLevel

Sets log levels for messages logged to the file. Default is `info`. Available log message types.

- `error`
- `warn`
- `info`
- `debug`
- `trace`
- `notset`

Only messages that are >= the log level defined here are logged to the file.

logToConsole

Set if message display in the QORTEX DTC console. This is a flag. Default value is `true`.

- `True` Messages display in the QORTEX DTC console.
- `False` Messages do not display in the QORTEX DTC console.

consoleLogLevel

Sets log levels for messages logged to the QORTEX DTC console. Default is `info`. Available log message types.

- `error`
- `warn`
- `info`
- `debug`
- `trace`

- notset

Only messages that are >= the log level defined here are logged to the file.

logStrategy

Set the method for handling messages sent to the log file. Default is `logger`. Options:

- `logger` Default. Log messages to the file using `ostream`. No file rotation or rolling options available.
- `syslog` Log messages to the file using `syslog handler`, which provides its own log file rotation/rolling. These rotation properties are defined in `/etc/logrotate.d/rsyslog`.
- `rolling` Log messages using customized properties with `boost` library. This provides log file rotation/rolling. The properties are defined by the parameters in the `properties` file, as explained below.

default

Set the default logging level for both the console and the log file. Default is `info`. Available log message types.

- `error`
- `warn`
- `info`
- `debug`
- `trace`
- `notset`

<parent_name>.<child_name>

Optionally, set log levels for sub-loggers. Both parameters must have values. This is available for all loggers and sub-loggers.

- `parent_name` The main logger. The log level is defined.
- `child_name` The sub-logger. By default, the sub-logger uses the same log level as the main logger. Use this parameter to set a different log level.

For example: `QortexServer.GRPCServer = warn`

– where `QortexServer` is a main logger and `GRPCServer` is a sublogger

rotationTime

Set the time lapse, in hours, to role the current log file. Requires `logStrategy` set to `rolling`. Default is 24 hours.

When the log file roles:

- The content in the current log file is transferred to a file, with an incremented filename, `qortex.server--<file_index>.log`. This transferred file is saved in the `logFile` directory.
- New log content continues to be added to the current, now empty, log file.

maxFileCount

Set the maximum number of files created without rotation. Requires `logStrategy` set to `rolling`. Default is 100 files.

The next file created after `maxFileCount` is reached, triggers rotation. Rotation is removing and replacing the oldest log file with the new empty log file ready to receive messages.

For example: If this parameter is set to 10 and the rotation time is set to 24 hours. A a new log file is created after every 24 hours. By the beginning of the 11th day, there are 10 logs files in the folder. One log for each of the last 10 days. At the end of the 11th day, the log file from day 1 is removed and replaced by the log file of the 11th day. The oldest log file is replaced when the `maxFileCount` parameter is reached.

maxLogFileSize

Set the maximum combined file size of the log files created without rotation. Requires `logStrategy` set to `rolling`. The value is in bytes. By default, this is not used to trigger rotation.

When the combined file size is reached, additional log entries trigger rotation. Rotation is removing and replacing the oldest log file with the new empty log file ready to receive messages. This is similar to maxFileCount, except file size is used to trigger rotation, rather than count.

minFolderSize

Set the minimum about of free space left in the log file directory without rotation. Requires `logStrategy` set to `rolling`. The value is in bytes. By default, this is not used to trigger rotation.

When the log file directory total free space drops to this value, additional log entries trigger rotation. Rotation is removing and replacing the oldest log file with the new empty log file ready to receive messages. This is similar to `maxFileCount`, except total free space is used to trigger rotation, rather than count.

Sample qortex.server.logging.properties File

Stored in `/home/quanergy/quanergy/qortex/`

```
logToFile   = true
logFile     = /home/user/qortex_log/qortex.server.log
fileLogLevel= warn
logToConsole= true
consoleLogLevel = warn
```

```
logStrategy = rolling
default = warn
QortexServer.GRPCServer = warn
rotationTime = 24
maxFileCount = 100
minFolderSize = 1000000000  # 1Gb
```

Appendix 1. QORTEX API

The QORTEX DTC 2.3 server publishes the object/trackable list, zone list, and state list as streams of data output from its processing action in a serialized format on the Ethernet network.

The QORTEX DTC 2.3 client consumes and visualizes these streams of published data, but any number of potential third-party host infrastructure applications can subscribe to them for surveillance or visualization purposes.

> **Note:** TCP port 17177 is reserved for gRPC communication between client and server. Other port numbers (17161, 17168, 17175, and 17178) can be defined in the `settings.xml` file.
>
> **Note:** If security is enabled, user authentication is required to enable connection between the QORTEX DTC server and QORTEX DTC client. See Appendix 3. *Cybersecurity* (page 255).

Outputs to Consume

Adjust the `settings.xml` file `<Publisher>` sections to determine which outputs are published, and in what format. See *Figure 100. Configuration of Generated settings.xml File Publisher Parameters*. The output data is available in multiple formats:

- **Protobuf** — delivers in binary packages; recommended for most efficient use of bandwidth.
- **JSON** — delivers in human-readable serial format.
- **XML** — delivers in human-readable serial format.
- **NDJSON** — delivers in human-readable serial format.

If the output format is changed, restart the server. See *Restart the Server Daemon Service* (page 57).

> **Note:** In the `settings.xml` file, QORTEX 1 refers to QORTEX DTC 1.x, and QTRACK refers to QORTEX DTC 2.x.

```
<TCPPublishers>
<!-- TCPPublishers section lists all publishing APIs, each publisher section has minimum:
Fixed Name, this is a string referenced in code to identify specific API, it could be:
        QORTEX1_OBJECT_LIST,
        QORTEX1_ZONE_LIST,  . . .
Publishing Format ‘json’, ‘xml’, ‘protofuf’, ‘ndjson’, or ‘none’: Optional parameter. Default is ‘none’
Publishing TCP Port: Optional parameter. Default is hard-coded value
-->
<Publisher>
```

```
   <Name>QORTEX1_OBJECT_LIST</Name>
   <Format>none</Format>
   <Port>17171</Port>
   <AddDataSize>false</AddDataSize>
   <NetworkByteOrder>false</NetworkByteOrder>
</Publisher>
<Publisher>
   <Name>QORTEX1_ZONE_LIST</Name>
   <Format>none</Format>
   <Port>17172</Port>
   <AddDataSize>false</AddDataSize>
   <NetworkByteOrder>false</NetworkByteOrder>
</Publisher>
<Publisher>
   <Name>QORTEX1_OBJECT_LIST</Name>
   <Format>none</Format>
   <Port>17161</Port>
   <AddDataSize>false</AddDataSize>
   <NetworkByteOrder>true</NetworkByteOrder>
</Publisher>
<Publisher>
   <Name>SENSOR_HEALTH_STATE</Name>
   <Format>none</Format>
   <Port>17168</Port>
   <AddDataSize>false</AddDataSize>
   <NetworkByteOrder>false</NetworkByteOrder>
</Publisher>
<Publisher>
   <Name>QORTEX1_STATE</Name>
   <Format>none</Format>
   <Port>17178</Port>
   <AddDataSize>false</AddDataSize>
   <NetworkByteOrder>false</NetworkByteOrder>
</Publisher>
<!-- Data intended for QORTEX DTC Client -->
<Publisher>
   <Name>CONFIGURATION_CLIENT_DATA</Name>
   <Port>17175</Port>
</Publisher>
<Publisher>
   <Name>MONITOR_CLIENT_DATA </Name>
   <Port>17176</Port>
</Publisher>
</TCPPublishers>
```

Figure 100. Configuration of Generated `settings.xml` File Publisher Parameters

Object Lists

QORTEX DTC 2.3 outputs two object lists:

- Object List
- Trackable List

Both the Object List and the Trackable List share the following characteristics:

Purpose: An existing system may subscribe to this data stream to gain detailed information and long-range movements of people and/or vehicles in the area.

Format: Default is none, which disables publication. To enable publication, specify protobuf (preferred), json, ndjson, or xml.

Frequency: The `<frequency>` parameter in the `<TrackerPipeline>` section of the `settings.xml` file determines the maximum rate, in Hz, at which the object list is published.

For a single LiDAR area: The output object list rate cannot exceed the rate at which the LiDAR is spinning, but the `<frequency>` parameter can reduce the rate at which the object list is published.

For a multi-LiDAR area: The multi-LiDAR pipeline performs several mergers and calculations before publishing the result. If this time-intensive step cannot be done at the rate described by the `<frequency>` parameter, a log message is published to state that the multi-LiDAR pipeline cannot keep up.

> **Note:** For the multi-LiDAR pipeline, the rate at which the object list is published does not necessarily match the timestamps of objects in those messages.

Object List (Port 17171)

The Object List is compatible with QORTEX DTC 1.x. See *Figure 101. Protobuf Output Format for QORTEX DTC 1.x Object List*. It is distinguished by these characteristics.

Available: On TCP port 17171.

Publishes: Specific data about each detected moving object (with representations of each trackable object), including:

- Unique track ID (64-bit length).
- Timestamp of object creation epoch time in milliseconds.
- 3D position (X, Y, Z) in meters.
- Size in each axis (height, width, depth) in meters.
- Velocity (X, Y, Z) in meters per second.
- Classification (unknown, human, vehicle, ignored).

```
Message QObjectArray {                     Object List Compatible with QORTEX DTC 1.x
    Header header = 1;
    repeated QObject object = 2;
}
Message Header (
    uint64 timestamp = 1;    // Epoch time in milliseconds
    string frame_id = 2:     // ’quanergy’
    uint32 sequence = 3;     // A counter that increases with each publish, regardless if any
                                 listener is listening
)
Message QObject {
    uint64 is = 1:               // Unique ID
    uint64 timestamp = 2;        // Object creation time: Epoch time in milliseconds
    Vector3 position = 3;        // Position (x, y, z) unit: meter
    Vector3 size = 4;            // Size in each axis (width, depth, height)
    Vector3 velocity = 5;        // Velocity (x, y, z) unit: meter/second
    enum ObjectClass {
        UNIDENTIFIED = 0;
        HUMAN = 1;
        VEHICLE = 2;
        IGNORED = 3;
    }
    ObjectClass object_class = 6;  // object class (UNIDENTIFIED, HUMAN, VEHICLE,
                                       IGNORED)
```

Figure 101. Protobuf Output Format for QORTEX DTC 1.x Object List

Trackable List (Port 17161)

The Trackable List is compatible with QORTEX DTC 2.0. See *Figure 102. Protobuf Output Format for QORTEX DTC 2.x Trackable List*. This trackable list is distinguished by these characteristics.

> **Note:** In QORTEX DTC 1.x, the coordinate system applied a left-hand rule. This means, the objects x and y coordinates were reversed in QORTEX DTC 1.x object lists.

Classification:

- `PERSON [mobile, movable], (trackableClassFlag 322)`
- `VEHICLE [mobile, movable], (trackableClassFlag 194)`
- `UNKNOWN, [0]`

The following additional classification options require `SubVehicle` license.

- `TWOWHEELER [mobile, movable], (trackableClassFlag 1218)`
- `PASSENGER_VEHICLE [mobile, movable], (trackableClassFlag 2242)`
- `COMMERCIAL_VEHICLE [mobile, movable], (trackableClassFlag 4290)`

Location: On TCP port 17161

Publishes: Compound data including timestamp, state, and decorators for each detected merged trackable, including:

- Unique track ID (64-bit length).
- Timestamp epoch time in milliseconds.
- Classification (person [mobile, movable], vehicle [mobile, movable], twowheeler [vehicle, mobile, movable], passenger [vehicle, mobile, movable], commercial [vehicle, mobile, movable], unknown)
- 3D position (X, Y, Z) in meters.
- Size in each axis (height, width, depth) in meters.
- Velocity (X, Y, Z) in meters per second.
- Centroid (X, Y, Z geometric center of the 3D shape) in meters.
- Acceleration (X, Y, Z) in meters per second squared.
- Float heading within a range of –π to π in radians.
- Float heading rate: rate of change of the heading, applicable only for vehicles, otherwise value is 0.

Trackable List of QORTEX DTC 2.x

```
Message QTrackableArray {
    uint64 timestamp = 1;       //Epoch time in milliseconds
    Repeated QTrackable trackable = 2;
}
// Trackable Classification (PERSON, TWOWHEELER_VEHICLE, PASSENGER_VEHICLE, COMMERCIAL_VEHICLE, VEHICLE and UNKNOWN is available and PERSON, TWOWHEELER_VEHICLE, PASSENGER_VEHICLE, COMMERCIAL_VEHICLE, VEHICLE are classified as MOBILE and MOVABLE as well)
// Note that TWOWHEELER_VEHICLE, PASSENGER_VEHICLE, COMMERCIAL_VEHICLE are only populated when a valid subclassification license add-on is present
Message TrackableClass {
    enum TrackableClassFlags {
        UNKNOWN                 0x00;
        IMMOVABLE               0x01;
        MOVABLE                 0x02;
        GROUND                  0x03;
        WALL                    0x04;
        CLUTTER                 0x10;
        STATIONARY              0x20;
        MOBILE                  0x40;
        VEHICLE                 0x80;
        PERSON                  0x100;
        BICYCLE                 0x200;
        TWOWHEELER_VEHICLE      0x400; //Requires SubVehicle license
        PASSENGER_VEHICLE       0x800; //Requires SubVehicle license
        COMMERCIAL_VEHICLE      0x1000;//Requires SubVehicle license
    }
    //Bitwise-OR of TrackableClassFlags.
    uint32 trackableClassFlags = 1;  // Bit-wise or of the TrackableClassFlags
}
Message QTrackable {
    uint64 id = 1;                       // Unique ID
    uint64 timestamp = 2;                // Epoch time in milliseconds
    TrackableClass trackable_class = 3;
    Vector3 position = 4;        // Position (x, y, z) unit: meter
    Vector3 size = 5;            // Size in each axis (width, depth, height)
    Vector3 velocity = 6;        // Velocity (x, y, z) unit: meter/second
    Vector3 centroid = 7;        // unit: meter
    Vector3 acceleration = 8;    // Acceleration (x, y, z) unit: meter/second
    float heading = 9;           // unit: radians
                                 // range -u to u
    float heading_rate = 10;     // Rate of change of the heading
                                 // This is applicable only for vehicles, otherwise value is 0
}
```

Figure 102. Protobuf Output Format for QORTEX DTC 2.x Trackable List

Object Movement Filter parameters

Object Movement Filter parameters are in the `settings.xml` file.

```
<OutputFilter>
<useMovementThresholdFilter>false</useMovementThresholdFilter>
<!-- Enable or disable movement threshold filter. This can be used to filter out
occasionally moving static objects (e.g. grass, foliage) which cause false objects.
-->
<minMovementThreshold>0.3</minMovementThreshold>
<!-- Trackables whose movement is less than this threshold (in meters) are filtered
out.
Movement is measured by the distance of trackable's current position from its
initial, i.e., first detected position. -->
</OutputFilter>
```

Zone List

Event zones are created and configured. See *Establishing Zones and* Counter Lines (page 105). If Event zones are in use, the Zone List is compatible with QORTEX DTC 1.x. See *Figure 103. Protobuf Output Format for QORTEX DTC 1.x and QORTEX DTC 2.x Zone List*. It has these characteristics:

Available: On TCP port 17172.

Level 0: Publishes zones with a list of the objects inside.

Level 1: Publish zones based on changing state of trackables inside [The Q-Track violation zone has a "Tigger state" field which specify one of the current state for trackables: "Enter", "Exit", "Appear", "Disappear"]

Level 2: Publish zones based on changing state of trackables and event configuration of zone. The format of this level is same as Level 1

Publishes at a regular frequency, typically 10 Hz:

- Unique IDs (match the object/trackable list IDs) of trackables inside a zone. Unique IDs are 128 bit length.
- Timestamp.
- String name.
- Polygon 2D shape and coordinates.
- Specific data to detect if an object track enters or exits a defined zone.
- Count the objects entered by type. This includes sub-classifications when licensed.
- Optionally defined data related to alerting the host environment.

Purpose: An existing system may wish to subscribe to this data stream to gain detailed information on people breaching pre-defined Event zones, including their long-range movements, and to determine the handling of potential threats, (use a third-party system to turn on a visual PTZ camera or alarm, for example).

Format: Default is none, which disables publication. To enable publication, the user specifies `protobuf` (preferred), `json`, `ndjson`, or `xml`.

Frequency: Published per frame to maintain the heartbeat for the server to sensor connection. The zone list is repeated in each message as long as the zone includes a trackable.

Zone List Compatible with QORTEX DTC 1.x and QORTEX DTC 2.x

```
message QZoneArray {
   Header header = 1;
   repeated QZone zones = 2;
)
message Header {
   uint64 timestamp = 1;  //Epoch time in milliseconds
   string frame_id = 2;
   uint32 sequence = 3;   // A counter that increases with each publish,
                          // regardless if any listener is listening
}
message QZone (
   string uuid = 1;             // string representation of a 128-bit Universally Unique ID
   uint64 timestamp = 2;
   string name = 3;
   Polygon2D shape = 4;         // supported for polygon/rectangle zones only
   uint32 object_count = 7;
```

```
    repeated int64 object_ids = 5;
    double z_min = 8;
    double z_max = 9;

// Zone Classification
    enum ZoneClass
    {
        FENCE = 0;
        EXCLUSION_ZONE = 1;
    }
    ZoneClass zone_class = 6;   // object list only for FENCE
         repeated ObjectCountByType object_counts = 10;
}
message ObjectCountByType {
  string object_class = 1;
  uint32 object_count = 2;
  repeated int64 object_ids = 3;
}
```

Figure 103. Protobuf Output Format for QORTEX DTC 1.x and QORTEX DTC 2.x Zone List

State List

The State List is distinguished by these characteristics:

Purpose: Administrators may subscribe to this data stream to monitor and resolve a Missing sensor issue quickly, or to check a Connected sensor health.

Publishes: During data collection or recording, reports a specific sensor current:

- **state of connection:**
    - **Connected**. The sensor is connected and sending point cloud information to the QORTEX DTC server. If the sensor loses its connection, it is automatically re-connected to QORTEX DTC.
    - **Missing**. The sensor point cloud is missing, which could indicate a sensor malfunction or other network congestion issue.
- **state of health:**

    For the M8 sensor, health status is available only for Rev D5 and up, or Rev D4 with a patch updating Base firmware to 7.03.

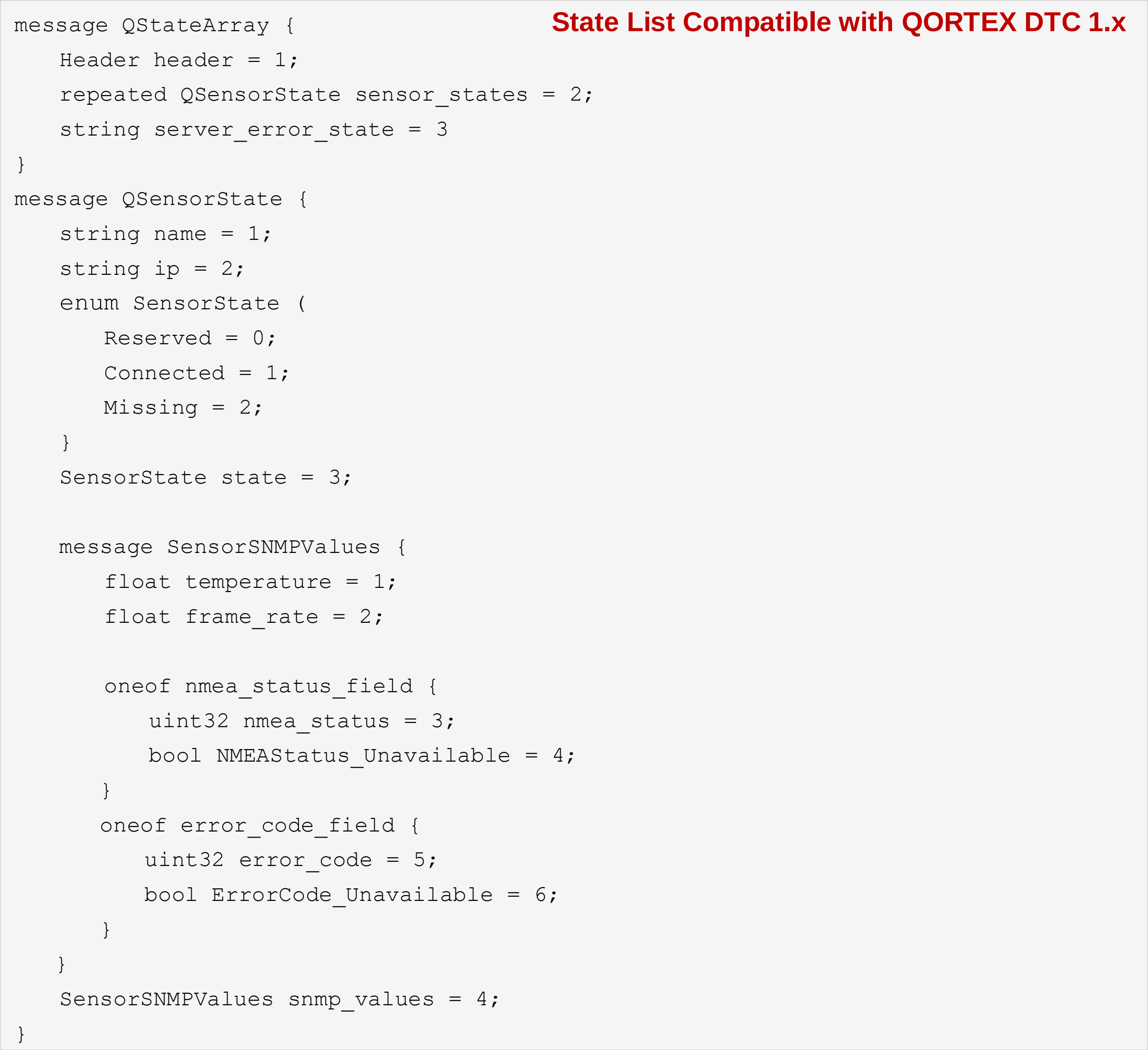
    - Temperature
    - Frame rate
    - Triggered error states

        Refer to the *Troubleshooting Issues* section in the specific sensor User Guide, which is available upon request from support@quanergy.com.

Format: User selects Protobuf (preferred), JSON, NDJSON, or XML.

QORTEX DTC 2.3 outputs two state lists:

- **QORTEX DTC 1.x compatible State List is** available on TCP port 17178. See *Figure 104. Protobuf Output Format for QORTEX DTC 1.x State List*. Sensor Health outputs information at 0.2 to 0.05 Hz, or every 5 to 20 seconds. Generally, the more sensors are polled, the slower the publishing rate becomes.
- **QORTEX DTC 2.x compatible State List** is available on TCP port 17168. See *Figure 105. Protobuf Output Format for QORTEX DTC 2.x State List*. Sensor Health outputs information at 0.2 to 0.05 Hz, or every 5 to 20 seconds. Generally, the more sensors are polled, the slower the publishing rate becomes. It is useful to maintain different frameIDs to identify which SNMP data reflects which sensor update. The output uses the frameID in the settings file, which may be different for each sensor. Whenever any sensor SNMP data is ready, all sensors' status is output, along with the particular sensor frameID (whose SNMP data is ready) under the **frameID** field.



State List Compatible with QORTEX DTC 1.x

```
message QStateArray {
    Header header = 1;
    repeated QSensorState sensor_states = 2;
    string server_error_state = 3
}
message QSensorState {
    string name = 1;
    string ip = 2;
    enum SensorState (
        Reserved = 0;
        Connected = 1;
        Missing = 2;
    }
    SensorState state = 3;

    message SensorSNMPValues {
        float temperature = 1;
        float frame_rate = 2;

        oneof nmea_status_field {
            uint32 nmea_status = 3;
            bool NMEAStatus_Unavailable = 4;
        }
        oneof error_code_field {
            uint32 error_code = 5;
            bool ErrorCode_Unavailable = 6;
        }
    }
    SensorSNMPValues snmp_values = 4;
}
```

Figure 104. Protobuf Output Format for QORTEX DTC 1.x State List

State List Compatible with QORTEX DTC 2.x

```
message SensorState {
    enum ConnectionStatus {
        CONNECTED = 0;
        DISCONNECTED = 1;
}
    enum SensorMaskingStatus {
     UNKNOWN = 0;
     NOT_DETECTED = 1;
     DETECTED = 2;
}
    enum SensorType {
        UNSPECIFIED = 0;
        LIDAR_PHYSICAL_M8 = 1;
        LIDAR_PHYSICAL_S3 = 2;
        CAMERA_PTZ = 3;
}
    message SNMP_M8 {
        double temperature = 1;
        double frame_rate = 2;
        // double hfov = 3;
        int32 nmea_status = 4;
        int32 error_code = 5;
}
    message SNMP_S3 {
        double temperature = 1;
        double frame_rate = 2;
        double min_range = 3;
        double max_range = 4;
        int32 start_angle = 5;
        int32 stop_angle = 6;
}
    message ONVIF_PTZ {
        int32 state =1;
        string message =2;
        string json_cam_model_data =3;
}
    one of SNMPValue {
        SNMP_M8 snmp_m8 = 1;
        SNMP_S3 snmp_s3 = 2;
        ONVIG_PTZ onvif_ptz = 8;
}
    ConnectionStatus connection = 3;
    SensorType sensor_type = 4;
    string sensor_name = 5;
    string sensor_ip = 6;
    string sensor_model = 7;
    SensorMaskingStatus masking_status = 9;
}
message SensorStateList {
    uint64 timestamp = 1;
    uint64 sequence = 2;
    string sensor_id = 3;
    repeated SensorState sensor_state_list = 4;
}
```

Figure 105. Protobuf Output Format for QORTEX DTC 2.x State List

Publication-Confirmation Utility

The netcat utility has a similar function as the Listener in bypassing the client to verify that the server is publishing serial streams. However, it only works for human readable formats (JSON, NDJSON, and XML), not for binary (Protobuf). For Windows 10, use PuTTY.

Use the netcat utility as follows:

1. Open an Ubuntu terminal window.
2. To monitor the TCP port where serial data is being published, edit this `netcat` (`nc`) command to include the IP address of the server host computer and the desired port number, then execute it. See *Server TCP Ports to Monitor* (page 92).

```
    $ nc x.x.x.x 171xx
```

   If serial data is published on that port, it prints directly to the terminal window.

3. After data starts flowing (and assuming the zones and objects are active), the Ctrl+C command can be executed to terminate the process.

API Help Command

The API help command is available. In an Ubuntu terminal window, execute the following to return a comprehensive list of arguments and parameters. See *Figure 106. API Help Commands*.

```
$ cd /opt/quanergy/qortex-server
$ ./Qortex-server --help
Qortex Server Application:
    -h [ --help]                        Display this help message
    -s [ --settings-file] arg           Object tracking settings file
    -o [ --csv-file] arg                Output a CSV file with tracking results
    --output-trackable-points           Output CSVs representing point data for
                                        each trackage (only effective when
                                        --cvs-file is used)
```

```
--record-path arg                    Path to create qlog directory for
                                     recording. When specified, all data is
                                     recorded unless --record-manually or
                                     --record-events is specified
                                     (incompatible with --replay-path)
--record-manually                    Recording starts/stops when spacebar is
                                     pressed (requires –record-path,
                                     incompatible with –record-events)
-C [ --compression-level] arg (=3)   If recording, compression level
                                     (0-9) to use, where 0 is no compression,
                                     9 is max compression
--replay-path arg                    Path to qlog file(s) or directory of qlog
                                     file(s) (incompatible with –record-path)
-t [ --start-time] arg (=0)          If replaying, start at this time into
                                     replay (seconds)
-r [ --replay-factor] arg (=1)       If replaying, multiply playback speed by
                                     this factor
-p [ --start-paused]                 If replaying, start paused
--license arg                        For detail information about license
                                     options use –license 'usage'
--loglevel arg                       Set log level for console output
-v [ --version]                      Display version information
```

Figure 106. API Help Commands

Appendix 2. API for Remote Procedure Calls

Third-party client software can make use of open-source remote procedure calls (https://grpc.io/) through the QORTEX DTC 2.3 server API commands.

> **Note:** If security is enabled, user authentication is required to enable connection between the QORTEX DTC server and QORTEX DTC client. Any application reading the QORTEX API TCP ports must authenticate with the QORTEX Server and decrypt the API data. See Appendix 3. *Cybersecurity* (page 255) and *Server TCP Ports to Monitor* (page 92).

Creating a python gRPC Client

In gRPC, a client application can directly call a method on a server application on a different machine as if it were a local object, making it easier to create distributed applications and services. As in many RPC systems, gRPC is based on defining a service, specifying the methods that can be called remotely with their parameters and return types. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub (referred to as a client in some languages) that provides the same methods as the server.

Creating a python gRPC client includes two steps:

1. Generate client code using the protocol buffer compiler.
2. Use the client code with gRPC API to make an API call.

Generate Client Code Using the Protocol Buffer Compiler

The protocol buffer compiler creates a `.py` file for each `.proto` file input. The names of the output files are computed by taking the name of the `.proto` file. Basically, the compiler generates two class files for each proto file provided. This class file must be created only once. If there are any changes/update in the proto file, then the class files must be created again. The command to generate class files from proto file is:

```
python -m grpc_tools.protoc -I <path to proto file>
--python_out= <path to create pb2.py file>
--grpc_python_out= <path to create pb2_grpc.py file> <proto file>
```

For example:

```
python -m grpc_tools.protoc -I ../../
--python_out=.
--grpc_python_out=. ../../Qortex_service.proto
```

The above command generates two files in the folder where the command is executed:

- `Qortex_service_pb2.py` - contains message classes
- `Qortex_service_pb2_grpc.py` - contains server and client classes

The following example command refers to multiple `.proto` files, which are available upon request from support@quanergy.com:

```
python -m grpc_tools.protoc -I ../../
--python_out=.
--grpc_python_out=.
Qortex_geometric_types.proto sensor_state_list.proto qortex_server_modes.proto
qtrackable.proto zone_3d.proto qortex_service.proto server_message.proto
zone_requests.proto
```

Use the Client Code with gRPC API to Make an API Call

Once the class files (client code) are generated, a channel and stub need to be created to send an API request along with metadata.

A gRPC channel provides a connection to a gRPC server on a specified host and port. It is used when creating a client stub.

To call service methods, first create a stub. Create a stub by instantiating the `Qortex ServiceStub` class of the `qortex_service_pb2_grpc` module, generated from our `.proto` file:

```
channel = grpc.insecure_channel("192.168.0.1:17177")
# Creates an insecure Channel to a server with port 17177
stub = qortex_service_pb2_grpc.QortexServiceStub(channel)
# Creating stub. Client sends a request to the server using the stub
```

Metadata is information about a particular RPC call (such as client ID) in the form of a list of key-value pairs, where the keys are strings and the values are typically strings. Metadata lets the client provide information associated with the call to the server and vice versa. QORTEX DTC uses client ID as metadata. For example:

```
client_id = "82ef205c-ed14-4013-89cf-85e11b6827b1"
# An example client ID. String representation of a 128-bit Universally unique ID
metadata = (("client_id", client_id),)
# Metadata to be transmitted to the service-side of the RPC
```

The QORTEX DTC server distinguishes API calls from multiple clients with the help of metadata. Each client has unique client ID, resulting in unique metadata, so API calls originating from client should have metadata associated with it.

Issuing API Commands

Each API command has its own permission for interacting with the client.

- **Configure Mode permission APIs**, the client must call `SwitchToConfigMode` API first, or else a `Permission denied` error occurs.
- **Monitor Mode permission APIs**, the client can call them with or without calling the `SwitchToConfigMode` API.

When software was installed on the server host computer, the default folder `/home/quanergy/quanergy/Qortex` was assigned permission level `777`.

SwitchToConfigMode API

Set Client in configuration mode. Required for multiple configuration actions.

Send and response: `SwitchToConfigMode`

```
    rpc SwitchToConfigMode (stream Empty) returns (stream Empty) {}
```

General Enumerations

gRPC uses Google Protocol Buffer (Protobuf 3) as its default parameter type.

- Some messages are used for general requests, such as `Empty`, `StatusResult`, and `RequestResult`.
- An `Empty` message means there are no other specific parameters (similar to void type in C++).
- The `StatusResult` and `RequestResult` are used to get a running status or to indicate if a request was successful.

Definitions of the messages are:

```
message Empty {}
message StatusResult {
 enum Status {
 Running = 0;
 Stopped = 1;
 }
 Status current_status = 1;
}

message RequestResult {
 bool result = 1;
}
```

Some enumeration types of messages are also frequently used by those APIs, including:

```
enum PubSubTopic {
 UNKNOWN  = 0;          /** reserved, to avoid use default enum type value */
 CONNECTION_STATUS = 1; /** Streaming type, used only for C++ client, to notify
connection is established/interrupted */ (see note 1)
 SERVER_MESSAGE = 2;    /** General type message; like server error, replay status
etc */
 SERVER_STATE = 3;      /** Will be published when play status is changed via
start/stop/pause/resume in live/playback position */
 SENSOR_STATE = 4;      /** SensorState will be published when fatal error in snmp
field or sensor changed to disconnected */
 ZONE_STATE = 5;        /** Will be published when zone is created/updated/deleted
*/
}
enum PlaybackCommand {
 GET_QLOG_LIST = 0;
 SPEED_UP = 1;
 SLOW_DOWN = 2;
 PAUSE = 3;
 RESUME = 4;
 SEEK = 5;
 STEP = 6;
 REPLAY = 7;
}
enum ServerMode {
 LIVE   = 0;
 PLAYBACK  = 1;
}

(Note 1) CONNECTION_STATUS is only used for a C++ client. The client will only
receive meaningful messages when the server side publishes them. Otherwise, blank
messages are received, and in C++ those blank messages are ignored. The expected
client behavior (depending on the C++ implementation) is: (1) The client sets the
topic of interest and sends a request to the server. (2) The client gets the
expected response with a corresponding data field that is contained in one of the
response fields.
```

PubSub API

`PubSubTopic` defines types of the topics that the API supports for Pub/Sub.

`PubSub` is acronym for Publish-Subscribe. This API is used by client to Subscribe to server, once subscribed, the state change of server gets published to all the clients that are subscribed. State changes such as starting a location, stopping a location, sensor spun up, sensor disconnected, recording started, recording stopped, add/edit/removing a zone, playback started, playback completed, sensor status (temperature, error code, ...) license state, etc.. shall get published.

These state changes are published from server in below message categories.

```
CONNECTION_STATUS = 1;
    /** Streaming type, only used for notifying connection has established/interrupted */
SERVER_MESSAGE    = 2;
    /** General type message; like server error, replay status etc */
SERVER_STATE      = 3;
    /** Will be published when play status is changed, like start/stop/pause/resume live/playback position */
SENSOR_STATE      = 4;
    /** SensorState will be published when there's fatal error in snmp field or sensor changed to disconnected */
ZONE_STATE        = 5;
    /** Will be published when zone is created/updated/deleted */
JSON_PUSH         = 6;
    /** Will be used for "pushing" notification from server in JSON format */
```

1. `Server_message` shall get published to monitor client as well, where as `server_state`, `sensor_state` and `zone_state` gets published to config client only.
2. Below is the `PubSub` API call to subscribe to server with a topic. The below code can be put inside a function after assigning a topic, similarly for different topic, different function can be created. Threading or multiprocessing can be used to call all the functions with different topics

```
def stream():
yield qtrack_service_pb2.PubSubRequest(topic= <'SERVER_MESSAGE' or
'SERVER_STATE' or 'SENSOR_STATE' or 'ZONE_STATE' >)
# Provide any one
input_stream = stub.PubSub(stream(), metadata=metadata)
while 1:
for i in input_stream:
print i.WhichOneof("response")
print ("\n PubSub Output is \n %s" % i)
print i.ClearField("response")
```

Send and response: `PubSub`

```
rpc PubSub(stream PubSubRequest) returns (stream PubSubResponse) {}
```

Sample

```
// Topics
message PubSubResponse {
 PubSubTopic topic = 1;
 oneof response {
 ServerMessage server_message = 2;
 ServerState server_state  = 3;
 Empty connection_status  = 4;
 SensorState sensor_state  = 5;
 ZoneState zone_state   = 6;
 }}
```

Playback API

`Playback` lists numbers of commands supported on the server side for playing back recorded QLogs.

Send and response: `Playback`

```
rpc Playback(PlaybackRequest) returns (PlaybackResponse) {}
```

ServerMode API

`ServerMode` indicates which mode the server is running.

Sample Python Code

Python code for using a General Enumerations API sample, see *Figure 107. Sample Python Code: General Enumerations API*.

```
def getserverstate(metadata, stub):
    """
    API call to get server state
    :param stub: stub object
    :param metadata: metadata to be sent
    :return: the server current state info such as server mode, version, is
        capturing, is paused, etc. .
    """
    state_req = qtrack_service_pb2.Empty()   # Creating an empty request
    state_resp = stub.GetServerState(state_req, metadata=metadata)
                                                        # Sending the request to server via
                                                        stub and getting response
    print "Server state is :\n", state_resp
    return state_resp

def getzone(metadata, stub, zonetype):
    """
    API call to get zone data
    :param stub: stub object
    :param metadata: metadata to be sent
    :param zonetype: type of zone (EVENT or EXCLUSION)
    :return: zone data such as zone name, UUID, class, vertices, etc. . of all the zones
        (Event/Exclusion)
    """
    getzone_req = zone_requests_pb2.ZonesRequest(zone_class=zonetype)
        # Creating a request with zone_class=<required zone class>
    getzone_resp = stub.GetZones(getzone_req, metadata=metadata)
        # Sending the request to server via stub and getting response
    print "The zone info is :\n", getzone_resp
    return getzone_resp

if __name__ == ‘__main__’:
    channel = grpc.insecure_channel("192.168.0.1:17177")
         # Creates an insecure Channel to a server
    stub = qtrack_service_pb2_grpc.QortexServerStub(channel)
        # Creating stub. Client sends a request to the server using the stub and waits for a responseto come
back, just like a normal function call
    client_id = "82ef205c-ed14-4013-89cf-85e11b6827b"      # An example client ID
    metadata = (("client_id", client_id),)                # Optional metadata to
        be transmitted to the service-side. Here metadata with client ID is a must
```

Figure 107. Sample Python Code: General Enumerations API

Calling Service Methods

There are several ways to call a service method.

Simple RPCs

A simple RPC is a unary RPC where the client sends a single request to the server and gets a single response back, just like a normal function call. For example:

```
liveModReq = qtrack_service_pb2.StartPlayRequest(mode="LIVE",)
    # Construct start Play request with mode value as LIVE
liveModRespon = stub.StartPlay(liveModReq,metadata=metadata)
    # send the request to server and wait for response
print "The start play result response is: ", liveModResponse.result
    # print the bool response result value
```

A call to the unary `StartPlay` API is nearly as straightforward as calling a local method. The RPC call waits for the server to respond, then either returns a response (a bool value) or raise an exception.

List of QORTEX DTC Server Simple/Unary RPCs

QORTEX DTC server has 26 simple/unary RPCs, grouped into the following modules:

`Config`
`Settings`
`Play/Playback`
`Track`
`Status`

Send and response: `config` module RPCs

```
rpc GetZones(ZonesRequest) returns (QZone3DArray) {} (see note 1)
rpc AddZone (AddZoneRequest) returns (RequestResult) {}
rpc EditZone (QZone3D) returns (RequestResult) {}
rpc RemoveZone (RemoveZoneRequest) returns (RequestResult) {} (see note 2)
rpc SetZonesEnabled (EnableZonesRequest) returns (RequestResult) {}
rpc GetZonesEnabled (ZonesRequest) returns (ZonesEnableStatus) {}
rpc ChangeEventZoneViolationAction (ChangeViolationActionRequest) returns
(RequestResult) {}
rpc ChangeEventZoneViolationTrigger (ChangeViolationTriggerRequest) returns
(RequestResult) {}
rpc GetEventZoneViolationActions(GetEventZoneViolationRequest) returns
(GetEventZoneViolationResponse) {}
rpc SetEventZoneViolationRecordingEnabled (ViolationRecodingStatus) returns
(RequestResult) {}
rpc GetEventZoneViolationRecordingEnabled (Empty) returns (ViolationRecodingStatus)
{}
rpc GetEventZoneViolationHandlerEnabled (Empty) returns (ViolationHandlerStatus) {}
rpc GetSettingsTemplateList(Empty) returns (GetSettingsTemplateResponse) {}

(Note 1) The request parameter for this RPC is:
ZonesRequest::QZoneClass::EVENT
ZonesRequest::QZoneClass::EXCLUSION
The message response for this RPC is:
message ZonesRequest {
 QZoneClass zone_class = 1;
}
(Note 2) The message response for this RPC is:
message RemoveZoneRequest {
 string uuid = 1;
}
```

Send and response: `settings` module

```
rpc GetSettings(GetSettingsRequest) returns (SettingsSectionResponse) {}
rpc SetSettings(SettingsSectionResponse) returns (RequestResult){}
rpc Settings(SettingsRequest) returns (SettingsResponse) {}
// config/recording module
rpc SetRecordingParams(RecordingParams) returns(Empty) {}
rpc GetRecordingParams(Empty) returns(RecordingParams) {}
rpc SetRecordingStatus(RecordingStatus) returns(Empty) {}
rpc GetRecordingStatus(Empty) returns(RecordingStatus) {} (See note 1)

(Note 1) The response for this RPC is the general usage message type, and the
result will reflect whether the request was successful:
message RequestResult {
 bool result = 1;
}
```

Send and response: `play/playback` module

```
//Stop capturing data in LIVE/PLAYBACK modes
    rpc StopPlay(Empty) returns(RequestResult) {}
//Start location in LIVE/PLAYBACK mode
    rpc StartPlay(StartPlayRequest) returns(RequestResult) {}
    rpc Playback(PlaybackRequest) returns (PlaybackResponse) {}
```

Send and response: `track` module

```
rpc ResetTrack(Empty) returns(RequestResult) {}
```

Send and response: `status` module

```
rpc GetServerState(Empty) returns (ServerState) {}
rpc GetSensorState(GetSensorStateRequest) returns (SensorStateList) {}
```

Bidirectional Streaming RPCs

A bidirectional streaming RPC occurs where both sides send a sequence of messages using a read-write stream. The two streams operate independently, so clients and servers can read and write in whatever order they like.

For example, the server could wait to receive all the client messages before writing its responses, or it could read a message then write a message, or some other combination of reads and writes.

The order of messages in each stream is preserved. Specify this type of method by placing the stream keyword before both the request and the response.

Streaming sample

```
def stream():
 yield qtrack_service_pb2.Empty() # Empty Stream of data from client
input_stream = stub.SwitchToConfigMode(stream(), metadata=metadata,
timeout=400000000)
# Send Empty stream of data from client and receive empty stream of data from server
while 1:
 print('switched to Config Mode.. \n {}'.format(next(input_stream)))
# indefinite receiving of empty data from server
```

List of QORTEX DTC Server Bidirectional RPCs

QORTEX DTC server has four bidirectional streaming RPCs, grouped by module:

```
Auth
Streaming
```

Send and response: bidirectional `auth` API

```
rpc SwitchToConfigMode (stream Empty) returns (stream Empty) {}
```

Send and response: bidirectional `streaming` API

```
rpc GetFile(stream FileChunk) returns (stream FileChunk) {} (see Note 1)
rpc PutFile(stream FileChunk) returns (stream FileChunkIndex) {}
rpc PubSub(stream PubSubRequest) returns (stream PubSubResponse) {}

(Note 1) The request parameter for this RPC needs to set
FileChunk::FileInfo::FileType as RECORDING_QLOG_FILE.
The message definition for file transmission is:
enum FileType {
 CALIBRATION_FILE  = 0; /* can only upload from client to server */
 QGUARD_SETTINGS_FILE = 1; /* can only upload from client to server */
 LOG_FILE = 2; /* can only download from server to client */
 RECORDING_QLOG_FILE = 3; /* can only download from server to client */
 GENERAL_FILE   = 4; /* can download/upload from/to server to/from client*/
}

message FileChunk {
 message FileInfo {
 FileType file_type = 1;
 string file_name = 2;
 uint64 file_size = 3;
 uint32 chunk_count = 4;
 }
 message RawData {
 uint32 chunk_index = 1;
 bytes data   = 2;
 }
 oneof Chunk {
 FileInfo file_info = 1;
 RawData raw_data = 2;
 FileChunkIndex chunk_index = 3;
 }
}

message FileChunkIndex {
 uint32 chunk_index = 1;
}
```

Authorization Module—SwitchToConfigMode API

`SwitchToConfigMode` API is an Authorization Module that puts the client into configuration mode for the requesting client, so that only one client can configure settings. There can be only one client in configuration (config) mode, so any request coming from another client for configuration is rejected.

SwitchToConfigMode API is activated by the request/response stream. The client initiates an empty request and the server responds with an empty request.

Sample

```
def stream():
 yield qtrack_service_pb2.Empty()
# Empty Stream of data from client
 input_stream = stub.SwitchToConfigMode(stream(), metadata=metadata,
 timeout=400000000)
# Send and receive Empty stream of data
 while 1:
 print('switched to Config Mode.. \n {}'.format(next(input_stream)))
# indefinite receiving of empty data from server
```

Certain APIs work only after the server verifies that the client is in configuration mode. If the server checks and discovers that the client is not in configuration mode, those APIs instigate an UNAUTHENTICATED error. Therefore, when calling those APIs, the SwitchToConfigMode API stream should be running in parallel, so that the above code would run in a separate python thread.

Terminating the SwitchToConfigMode API call switches the client into monitor mode. Terminating is done by stopping the python thread that started.

APIs That Do Not Require SwitchToConfigMode Module

Below is the list of APIs that **do not** require the SwitchToConfigMode API stream to run in parallel. The APIs listed below work in **Monitor** mode, while the remaining APIs need the client to be in config mode. These APIs are grouped into the following modules:

Zone
Status
Streaming

Send and response: zone module

```
rpc GetZones(ZonesRequest) returns (QZone3DArray) {}
rpc GetEventZoneViolationActions(GetEventZoneViolationRequest) returns
(GetEventZoneViolationResponse) {}
rpc GetEventZoneViolationRecordingEnabled (Empty) returns (ViolationRecodingStatus)
{}
rpc GetEventZoneViolationHandlerEnabled (Empty) returns (ViolationHandlerStatus) {}
```

Send and response: status module

```
rpc GetServerState(Empty) returns (ServerState) {}
rpc GetSensorState(GetSensorStateRequest) returns (SensorStateList) {}
```

Send and response: streaming module

```
rpc GetFile(stream FileChunk) returns (stream FileChunk) {}
rpc PubSub(stream PubSubRequest) returns (stream PubSubResponse) {}
```

Visualization Module

The Visualization module includes the APIs for `StartPlay`, `StopPlay`, `ResetTrack`, `SetRecordingParams`, `GetRecordingParam`, `SetRecordingStatus`, `GetRecordingStatus`, and `Playback`. These APIs enable the visualization of real-time (Live) or recorded (Playback) point cloud data. Most of these are simple RPCs. Sample Python code for using a Visualization API, see *Figure 108. Sample Python Code: Visualization API*.

```
def startplay(metadata, stub):
      """
      API call to start a location in live mode
      :param stub: stub object
      :param metadata: metadata to be sent
      : return: Bool value
      """
      print("starting live mode")
      liveModeReq =
      qtrack_service_pb2.StartPlayRequest(mode="LIVE")
         # Constructing start play request with
ServerMode enum value
      liveModeResponse =
      stub.StartPlay(liveModeReq,metadata=metadata)
      print "The start play result response is: ",
         liveModeResponse.result
      return liveModResponse.result
if __name__ ** '__main__':
      channel =
      grpc.insecure_channel("192.168.0.1:17177")
         # Creates an insecure Channel to a server
      stub =
      qtrack_service_pb2_grpc.QortexServiceStub(channel)
         # Creating stub. Client sends a request to the server using the
         # stub and waits for a response to come back, just like a normal
         # function call
      client_id = "82ef285c-ed14-4813-89cf-85el1b6827b1"
         # An example client ID
      metadata = (("client_id", client_id),)
         # Optional metadata to be transmitted to the service-side of the
         # RPC. Here metadata to be transmitted to the service_side of the
         # RPC.
      startplay(metadata,stub)
```

```
enum ServerMode {
      LIVE=0;
      PLAYBACK=1;
}

/********************
 * StartPlay
 */
message
StartPlayRequest {
      SeverMode mode=1;
      string
qlog_name=2;
}
```

Figure 108. Sample Python Code: Visualization API

Each API is explained below, with an example of the call and response. These are:

`GetRecordingParam`
`GetRecordingStatus`
`Playback`
`ResetTrack`
`Set_Qlog`
`SetRecordingParam`
`SetRecordingStatus`
`StartPlay`
`StopPlay`

GetRecordingParam API

Fetches the recording parameters set in the server by using the `SetRecordingParams` API. It returns the record folder filepath and compression level set.

Send and response: `GetRecordingParams`

```
rpc GetRecordingParams(Empty) returns(RecordingParams) {}
```

Sample

```
getRecParm = qtrack_service_pb2.Empty()
    # Construct an empty request
getRecParmResp = stub.GetRecordingParams(getRecParm,metadata=metadata)
    # Send the Empty request to GetRecordingParams API and get response
print getRecParmResp
    # print recording folder name and compression level
```

GetRecordingStatus API

Fetches the recording status (whether a recording is started/in-progress or stopped/completed).

Send and response: `GetRecordingStatus`

```
rpc GetRecordingStatus(Empty) returns(RecordingStatus) {}
```

Sample

```
recsts_req = qtrack_service_pb2.Empty()
    # Construct an empty request
recSts_resp = stub.GetRecordingStatus(recsts_req,metadata=metadata)
    # Send the Empty request to GetRecordingStatus API and get response
print "Recording status is (0=STOP,1=START): ", recSts_resp.status
    # print the response value 0 (STOP) or 1 (START)
```

Playback API

Used to play a recorded dataset. This API can pause, resume, or replay a dataset, step a duration, seek to a duration, and speed up or slow down a playing dataset. Playing a recorded dataset is a two-step process. First, the client must send a request to the server to get all available datasets from the `datasets` folder with `Playback` API having `cmd=GET_QLOG_LIST`.

Second, switch to playback mode to play the dataset with `StartPlay` API with `mode=Playback` and `qlog_name = <required qlog name>` fetched from first step.

Send and response: `Playback`

```
rpc Playback(PlaybackRequest) returns (PlaybackResponse) {}
```

Sample

Step 1

```
playbck_req = qtrack_service_pb2.PlaybackRequest(cmd= "GET_QLOG_LIST")
    # construct request to get all available qlog/datasets from datasets folder
playback_resp = stub.Playback(playbck_req, metadata=metadata)
    # Send request to Playback API in server, get qlog/dataset list as response
print playback_resp
```

Step 2

```
playbackmodereq = qtrack_service_pb2.StartPlayRequest(mode= "PLAYBACK",
qlog_name=<qlog_name>)
    # Construct request to switch to playback mode with mode=PLAYBACK and
    qlog_name=<any one qlog name from step1
playbackmodresp = stub.StartPlay(playbackmodereq, metadata=metadata)
    # Send the request to StartPlay API in server and get bool response
return playbackmodresp.result
```

PAUSE, RESUME or REPLAY

To PAUSE, RESUME or REPLAY a dataset `Playback` API with a different `cmd` value, see the following example. The request should contain the required command and the qlog/dataset name. The constructed request must be sent via stub to `Playback` API in the server.

```
playbck_req =
qtrack_service_pb2.PlaybackRequest(cmd="PAUSE",qlog_name=<qlog_name>)
    # Request to pause a running dataset
playbck_req =
qtrack_service_pb2.PlaybackRequest(cmd="RESUME",qlog_name=<qlog_name>)
    # Request to resume a paused dataset
playbck_req =
qtrack_service_pb2.PlaybackRequest(cmd="REPLAY",qlog_name=<qlog_name>)
    # Request to replay a dataset
```

STEP or SEEK

To STEP or SEEK, the `Playback` API must import `google.protobuf.timestamp_pb2`. The protobuf timestamp represents a point in time independent of any time zone or calendar, represented as seconds and fractions of seconds at nanosecond resolution in UTC Epoch time. For example, 15.23 seconds is represented in Google timestamp as `Timestamp(seconds=15, nanos=230000000)`. STEP allows step dataset play for a constant value, and it can be performed only when the dataset is paused. SEEK allows the dataset to jump to the required time, and it can be performed in a paused or running state.

```
playbck_req = qtrack_service_pb2.PlaybackRequest(cmd= "STEP",
qlog_name=<qlog_name>,timestamp=Timestamp(seconds=0, nanos=100000000))
   # Request to step 100 ms
playbck_req = qtrack_service_pb2.PlaybackRequest(cmd= "SEEK",
qlog_name=<qlog_name>,timestamp=Timestamp(seconds=15, nanos=230000000))
   # Request to seek/jump to 15.23 seconds (eg values)
```

Change Speed

To change the speed of playback, a running playback/recorded dataset can be sped up or slowed down. The speed of the dataset increments is twice the previous value (2x, 4x, 8x, 16x, 32x, 64x ,128x) until the max value of 128x for every call of `SPEED_UP` is reached. Similarly, `SLOW_DOWN` reduces the speed at twice its earlier speed up to a maximum value of 0.001x for every call of `SLOW_DOWN`. For example:

```
playbck_req =
qtrack_service_pb2.PlaybackRequest(cmd="SPEED_UP",qlog_name=<qlog_name>)
   # Request to speed up a dataset
playbck_req =
qtrack_service_pb2.PlaybackRequest(cmd="SLOW_DOWN",qlog_name=<qlog_name>)
   # Request to slow down a dataset
```

ResetTrack API

Resets the trackables on the server side. The server restarts the tracking from scratch.

Send and response: `ResetTrack`

```
rpc ResetTrack(Empty) returns(RequestResult) {}
```

Sample

```
resetReq = qtrack_service_pb2.Empty()
   # Construct an empty request
resetResp = stub.ResetTrack(resetReq, metadata=metadata)
   # Send the Empty request to ResetTrack API in server and get response
print "The reset track response is: ",resetResp.result
   # print bool result response
```

Set_Qlog API

Used to add the datasets folder path to the QORTEX DTC server configuration so the server can read/write dataset in `settings.xml` file.

```
playbck_req =
qtrack_service_pb2.PlaybackRequest(cmd="SET_QLOG",qlog_name=<qlog_name>)
   # Request to change the path to datasets folder path.
```

SetRecordingParam API

Sets recording parameters such as Recording folder name and compression level of recording data. The folder name provided is created inside the `datasets` folder located in `/home/quanergy/quanergy/qortex`. All recordings are stored inside the created folder. If no folder name is given, the recordings are created inside the datasets folder. The API does not accept multiple folder level names, such as `Recordings/Zone1` or `/home/user/Recordings/`.

For the compression level, any integer value from `0` to `9` is accepted (`0` = no compression; `9` = high compression). A location has to be running/playing in live or in playback mode in order to call this API. If location is not running/playing in live or playback, this error occurs: `Server-side pipeline is not running. Please start play first`. Recording cannot be started when a playback is running.

Send and response: `SetRecordingParams`

```
rpc SetRecordingParams(RecordingParams) returns(Empty) {}
```

Sample

```
setRecParm = qortex_server_modes_pb2.RecordingParams(record_path="Recordings", compression_level=<0 to 9>)
    # Construct RecordingParams request with folder name and compression level
SetRecParmResp = stub.SetRecordingParams(setRecParm,metadata=metadata)
    # Send request to SetRecordingParams API in server,get EMPTY/BLANK response
 print "Recording params settings response is EMPTY", SetRecParmResp
```

SetRecordingStatus API

Used to **START** or **STOP** a recording. The status parameter value should be either **START** or **STOP**. The recording starts only when a location is running in live mode, or else this error occurs: `Server-side pipeline is not running. Please start play first`.

Send and response: `SetRecordingStatus`

```
rpc SetRecordingStatus(RecordingStatus) returns(Empty) {}
```

Sample

```
setrecsts_req = qortex_server_modes_pb2.RecordingStatus(status= "START or STOP")
    # Construct RecordingStatus with status value as START or STOP
recresp = stub.SetRecordingStatus(setrecsts_req, metadata=metadata)
    # Send request to SetRecordingStatus API in server,get EMPTY/BLANK response
print "Set(Start/Stop) Recording status response is EMPTY ", recresp
```

StartPlay API

Starts sensor connection and specifies **LIVE** or **PLAYBACK** mode in the request parameter. If **PLAYBACK** is specified, this API must also specify the additional `qlog` name parameter. If the mode is **LIVE**, the API starts a location in **Live** mode.

Send and response: `StartPlay`

```
rpc StartPlay(StartPlayRequest) returns(RequestResult) {}
```

Sample

```
liveModReq = qtrack_service_pb2.StartPlayRequest(mode= "LIVE")
    # Construct startPlayrequest with mode value as LIVE
liveModResponse = stub.StartPlay(liveModReq,metadata=metadata)
    # send the request to StartPlay API in server and get response.
print "The start play result response is: ", liveModResponse.result
    # print the bool response result value
```

StopPlay API

Stops the sensor connection/live mode location. This API is also used to stop the playback data set.

Send and response: `StopPlay`

```
rpc StopPlay(Empty) returns(RequestResult) {}
```

Sample

```
stopLoc = qtrack_service_pb2.Empty()
    # Construct an empty request
stop_loc = stub.StopPlay(stopLoc, metadata=metadata)
    # Send the Empty request to StopPlay API in server and get response
print "The stop play result response is: ", stop_loc.result
    # print bool result response
```

Configuration Module

Configuration Module APIs allow you to configure settings in the QORTEX DTC server. Most of these are simple RPCs. These APIs enable control of location, settings, zones, counter lines, and client.

Sample Python code for using a Configuration API. See *Figure 109. Sample Python Code: Configuration API* and *Figure 110. Sample Python Code: Add Zone.*

```
def add_zone(stub, metatdata, name, UUID, zMin, zMax, ZoneClass, Enabled,
X_vertices_list, Y_vertices_list, vertices_list, sensor_list, zone_info):
"""
    API call to add a zone in settings.xml file
    :param stub: stub object; :param metadata: metadate to be sent
    :param name: Name of the zone to be created in string
    :param UUID: Unique ID of the zone // string representation of a 128-bit Universally Unique ID
    :param zMIN: Z axis minimum height
    :param zMax: Z axis maximum height
    :param ZoneClass: Type of Zone (EVENT or EXCLUSION or INCLUSION)
    :param Enables: Enabled value in bool, true/false. true to view Exclusion zones.
    :param X_vertices_list: list of x axis vertices of the zone
    :param Y_vertices_list: list of y axis vertices of the zone
    :return: bool True or False
"""
vertices_list = []
try:
   zone_info = qtrack_service_pb2.zone—3d__pb2.QZone(name=str(name),
type="POLYGON") for x, y in zip(X_vertices_list, Y_vertices_list):
vertices_list.append(
qtrack_service_PB2.zone__3d__pb2.qortex__geometric__types__pb2.Vector2
(x=float(x), y=float(y)))
   Zone_vertices =
qtrack_service_PB2.zone__3d__pb2.qortex__geometric_types__pb2.Polygon2D
(vertices=vertices_list)
   zone_vert = qtrack_service_PB2.zone__3d__pb2.QPolygonGeometry
(vertices=Zone_vertices)
   zone_params = qtrack_service_PB2.zone__3d__pb2.QZone3D(uuid=str(UUID),
zone=zone_info, z_min=float(zMin), z_max=float(zMax), zone_class=str(ZoneCass),
enabled=str_to_bool(Enables), polygon_3d=zone_vert)
   zone_req = zone_requests_pb2.AddZoneRequest(zone=zone_params)
   req_resp = stub.AddZone(zone_req, metadata=metadata)
      # Sending add zone request to server
   print "Add Zone response is: ", req_resp.result
except Exception as err:
   print err
If __name__ == '__main__':
channel = grpc.insecure_channel("192.168.0.1:17177")
    # Creates an insecure Channel to a server
stub = qtrack_service_pb2_grpc.QortexServiceStub(channel)
    # Creating stub. Client sends a request to the server using the stub and waits for a response to
    # come back, just like a normal function call
client_id = "82ef205c-ed14-4013-89cf-85ellb6827b1")
    # An example client ID
metadata = (("client_id", client_id),)
    # Optional metadata to be transmited to the service-side of the RPC. Here metadata with
    # client ID is a must
add_zone(stub, metadata, name="parking", UUID=”61de205c-ed14-4013-89cf-
75ellb6827b2", zMin="0", zMax="4", ZoneClass="EVENT", Enabled=True,
X_vertices_list=[2.1,3.2,5.4], Y_vertices_list=[4.1,2.1,9.3])
```

Figure 109. Sample Python Code: Configuration API

```
def add_zone(stub, metadata):
    Y_vertices_list = [2.6, -1.9, 2.4]
    X_vertices_list = [6.3, 5.3, -9]
    vertices_list = []
    sensor_list = ["10.1.22.160"]
    zone_info = qtrack_service_pb2.zone__3d__pb2.QZone(name="Zone_12",
      type="POLYGON")
    for x, y in zip(X_vertices_list, Y_vertices_list):
        vertices_list.append(qortex_geometric_types_pb2.Vector2(x=float(x),
          y=float(y)))
    Zone_vertices = qortex_geometric_types_pb2.Polygon2D(vertices=vertices_list)
      # creating a request with vertices list for polygon2
    zone_vert =
qtrack_service_pb2.zone__3d__pb2.QPolygonGeometry(vertices=Zone_vertices)
    # creating a request with vertices list for QPolygonGeometry
    zone_params = qtrack_service_pb2.zone__3d__pb2.QZone3D(uuid="9a748505-d8b8-
4119-bbd7-89d95568513b", zone=zone_info,
    z_min=-3.0, z_max=4.0,zone_class="EXCLUSION", enabled=True,
polygon_3d=zone_vert,exclusion_sensors=sensor_list)
    zone_req = zone_requests_pb2.AddZoneRequest(zone=zone_params)
      # Constructing AddZone request
    req_resp = stub.AddZone(zone_req, metadata=metadata)
    print  (req_resp)
```

Figure 110. Sample Python Code: Add Zone

Counter Line APIs

Use the Counter Line to count trackable objects that cross a specified line within an area. The line can have multiple points. The minimum number of points to create the line is two. Trackables are counted as they cross the line in either a FORWARD or BACKWARD direction. Forward or Backward are typically relative to entering or exiting a zone or area. The data collected is the number of and timestamp for trackables crossing the line.

The command contains:

- ID: message ID, in this case is AddCounterLine
- Params: see below

The params contain:

- Name: CounterLine name assigned on the client
- Points: the points representing counter line segment(s)
- UserDirection: the direction of trackable given by user
- Point0: the starting point of the user given direction
- Point1: the end point of the user given direction

- segmentIndex: the segment index of counter line where the user direction is pointing at
- direction: the crossing type of trackable object when it goes according to the given direction
- enabled: indicate whether this counterline is enabled (true) or not (false)

CounterLine Settings in settings.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<Settings>
    <CounterLines>
        <CounterLine>
            <name>0</name>
            <Points>
              <Point>
                <x>0</x>
                <y>10</y>
              </Point>
              <Point>
                <x>0</x>
                <y>0</y>
              </Point>
              <Point>
                <x>10</x>
                <y>0</y>
              </Point>
              <Point>
                <x>10</x>
                <y>10</y>
              </Point>
            </Points>
            <UserDirection>
              <Point0>
                <x>-2</x>
                <y>5</y>
              </Point0>
              <Point1>
                <x>-1</x>
                <y>5></y>
              </Point1>
              <direction>FORWARD</direction>
              <segmentIndex>1</segmentIndex>
            </UserDirection>
        </CounterLine>
    </CounterLines>
</Settings>
```

Counter Line Object List Message (Port 17163)

```
message QCounterLineObjectArray {
  QCounterLineObjectsHeader header = 1;     // header
  repeated QCounterLineObject object = 2;   // group of published objects
}

message QCounterLineObjectsHeader {
  string messageType = 1;   // message type, "DETECTION"
  uint64 numObjects = 2;    // number of objects
  uint64 sequence = 3;      // message sequence id
  uint64 timestamp = 4;     // Epoch time in milliseconds
  string version = 5;       // application protobuf version, "1.0.0"
}

message QCounterLineObject {
  uint64 id = 1;                  // Unique ID
  uint64 timestamp = 2;           // Epoch time in milliseconds
  Vector3 position = 3;           // Position (x, y, z) unit: meter

  float speed = 4;                // Scalar speed
  string objectClass = 5;         // QTrack object class
  string direction = 6;           // trackable direction crossing the line, one of:
["FORWARD", "BACKWARD"]
  string lineName = 7;            // CounterLine name
  uint64 duration = 8;            // object duration
}
```

Counter Line Detection Message

```
{
  "header": {
    "messageType": "DETECTION",
    "numObjects": "1",
    "sequence": "1",
    "timestamp": "1539029356950793000",
    "version": "1.0.0"
  },
  "object": [
    {
      "id": "1",
      "timestamp": "1539029251157398000",
      "position": {
       "x": 0.1,
       "y": 0.1,
       "z": 0.1
       },
      "speed": 0.0656,
      "objectClass": "HUMAN",
      "direction": "FORWARD",
      "lineName": "door1",
      "duration": ""
    }
  ]
}
```

Counter Line Protobuf

This is also compatible with People Counter on S3 sensors.

```
syntax = "proto3";
package quanergy.qortex.protobuf;
import "qortex_geometric_types.proto";
message QCounterLineObjectArray {
  QCounterLineObjectsHeader header = 1;     // header
  repeated QCounterLineObject object = 2;   // group of published objects
}

message QCounterLineObjectsHeader {
  string messageType = 1;   // message type, "DETECTION"
  uint64 numObjects = 2;    // number of objects
  uint64 sequence = 3;      // message sequence id
  uint64 timestamp = 4;     // Epoch time in milliseconds
  string version = 5;       // application protobuf version, "1.0.0"
}

message QCounterLineObject {
  uint64 id = 1;                  // Unique ID
  uint64 timestamp = 2;           // Epoch time in milliseconds
  Vector3 position = 3;           // Position (x, y, z) unit: meter

  float speed = 4;                // Scalar speed
  string objectClass = 5;         // QTrack object class
  string direction = 6;           // trackable direction crossing the line, one of:
                                  // ["FORWARD", "BACKWARD"]
  string lineName = 7;            // CounterLine name
  uint64 duration = 8;            // object duration
}
```

Counter Line TCP Publisher Output

```
{
 "header": {
  "timestamp": "1596710280552",
  "frameId": "quanergy",
  "sequence": 3
 },
 "zones": [
  {
   "uuid": "92d38325-7315-4312-bfc4-9d3a9bd26e08",
   "timestamp": "1596710280552",
   "name": "EVENT ZONE 0",
   "shape": {
    "vertices": [
     {
      "x": 4.98169136,
      "y": -16.4128914
     },
     {
      "x": 4.75,
      "y": -7.11669922
     },
     {
      "x": 8.85139465,
      "y": -6.75
     },
     {
      "x": 10.75,
      "y": -15.6122742
     }
    ]
   },
   "objectCount": 2,
   "objectIds": [
    "45",
    "46"
   ],
   "zMin": 0,
   "zMax": 4,
   "zoneClass": "FENCE",
   "objectCounts": [
    {
      "objectClass": "person",
      "objectCount": 1,
      "objectIds": ["45"]
    },
    {
      "objectClass": "unknown",
      "objectCount": 1,
      "objectIds": ["46"]
    }
   ]
  }
 ]
}
```

Counter Line TCP Publisher when objectCounts are Zero

```
{
 "header": {
  "timestamp": "1596710283705",
  "frameId": "quanergy",
  "sequence": 19
 },
 "zones": [
  {
   "uuid": "92d38325-7315-4312-bfc4-9d3a9bd26e08",
   "timestamp": "1596710283705",
   "name": "EVENT ZONE 0",
   "shape": {
    "vertices": [
     {
      "x": 4.98169136,
      "y": -16.4128914
     },
     {
      "x": 4.75,
      "y": -7.11669922
     },
     {
      "x": 8.85139465,
      "y": -6.75
     },
     {
      "x": 10.75,
      "y": -15.6122742
     }
    ]
   },
   "objectCount": 0,
   "objectIds": [],
   "zMin": 0,
   "zMax": 4,
   "zoneClass": "FENCE",
   "objectCounts": []
  }
 ]
}
```

Counter Line TCP Publisher Output in NDJSON

```
U{"header":{"messageType":"DETECTION","numObjects":"1","sequence":"422","timestamp"
:"1568309378714","version":"1.0.0"},"object":[{"id":"189762","timestamp":"156830937
8714","position":{"x":1.68940258,"y":-1.36005783,"z":-
0.221662849},"speed":1.5546937,"objectClass":"person","direction":"FORWARD","lineNa
me":"Counter Line 0","duration":"0"}]}
```

AddCounterLine command message

```
{
  "group": "COUNTERLINE",
  "command": {
    "id": "AddCounterLine",
    "params": {
      "name": "CounterLine 1",
      "Points": [
        {
          "x": 0,
          "y": 10
        },
        {
          "x": 0,
          "y": 0
        },
        {
          "x": 10,
          "y": 0
        },
        {
          "x": 10,
          "y": 10
        }
      ],
      "UserDirection": {
        "Point0": {
          "x": -2,
          "y": 5
        },
        "Point1": {
          "x": -1,
          "y": 5
        },
        "segmentIndex": 0,
        "direction": "FORWARD"
      },
      "enabled": true
    }
  }
}
```

AddCounterLine response message

```
{
  "reply":"OK"
}
```

CounterLine state message

Server publishes this message to all connected clients to indicate that counter lines state on the server has changed. The message includes:

- id: State
- mode: indicate the type of update, with three possible values: AddCounterLine, EditCounterLine, or RemoveCounterLine. The example below with AddCounterLine indicate that a counter line has been added.
- params: content of the change requested

Client that does not initiate the update needs to apply the update accordingly.

```
{
  "group":"COUNTERLINE",
  "command":{
    "id":"State",
    "mode":"AddCounterLine",
    "params":{
      "name":"Counter Line 0",
      "Points":[
        {
          "x":"-15.75",
          "y":"6.04919"
        },
        {
          "x":"-9.87442",
          "y":"5.5"
        }
      ],
      "UserDirection":{
        "Point0":{
          "x":"-12.8809",
          "y":"5.04015"
        },
        "Point1":{
          "x":"-13.1799",
          "y":"1.84115"
        },
        "segmentIndex":"0",
        "direction":"FORWARD"
      },
      "enabled":"true"
    }
  }
}
```

EditCounterLine command message

EditCounterLine command message contains details about an existing CounterLine to be updated. See AddCounterLine for more details.

Note: newName field is optional and when it's present then CounterLine name will be updated using the newName.

```
{
  "group": "COUNTERLINE",
  "command": {
    "id": "EditCounterLine",
    "params": {
      "name": "CounterLine 1",
      "newName": "CounterLine new name",
      "Points": [
        {
          "x": 0,
          "y": 10
        },
        {
          "x": 0,
          "y": 0
        },
        {
          "x": 10,
          "y": 0
        },
        {
          "x": 10,
          "y": 10
        }
      ],
      "UserDirection": {
        "Point0": {
          "x": -2,
          "y": 5
        },
        "Point1": {
          "x": -1,
          "y": 5
        },
        "segmentIndex": 0,
        "direction": "BACKWARD"
      },
      "enabled": false
    }
  }
}
```

EditCounterline response message

```
{
  "reply":"OK"
}
```

GetCounterLines command message

GetCounterLines command message can be used to request all existing Counter Lines from server. Params would be empty for this message.

```
{
  "group": "COUNTERLINE",
  "command": {
    "id": "GetCounterLines",
    "params": {}
  }
}
```

GetCounterLines response message

```
{
  "reply":"OK",
  "result":[
    {
      "name":"Counterline 1",
      "Points":[
        {
          "x":"0",
          "y":"10"
        },
        {
          "x":"0",
          "y":"0"
        },
        {
          "x":"10",
          "y":"0"
        },
        {
          "x":"10",
          "y":"10"
        }
      ],
      "UserDirection":{
        "Point0":{
          "x":"-2",
          "y":"5"
        },
        "Point1":{
          "x":"-1",
          "y":"5"
        },
        "segmentIndex":"0",
        "direction":"FORWARD"
      },
      "enabled":"true"
    },
    {
      "name":"Counterline 2",
      "Points":[
        {
          "x":"0",
          "y":"-2"
        },
        {
          "x":"0",
          "y":"-7"
        },
        {
          "x":"10",
          "y":"-7"
        }
      ],
      "UserDirection":{
        "Point0":{
          "x":"-2",
          "y":"-5"
        },
        "Point1":{
          "x":"-1",
          "y":"-5"
        },
        "segmentIndex":"0",
        "direction":"FORWARD"
      },
      "enabled":"false"
    }
  ]
}
```

GetCounterLinesEnabled command message

GetCounterLinesEnabled command message can be used to request the status of (all) CounterLines from server. Params would be empty for this message.

```
{
  "group": "COUNTERLINE",
  "command": {
    "id": "GetCounterLinesEnabled",
    "params": {}
  }
}
```

GetCounterLinesEnabled response message

GetCounterLinesEnabled response message contains counterLinesEnabled field which indicate the current CounterLines status, whether they are enabled (true) or disabled (false).

```
{
  "reply":"OK",
  "counterLinesEnabled":"true"
}
```

RemoveCounterLine command message

RemoveCounterLine command message contains info about an existing CounterLine to be removed.

```
{
  "group": "COUNTERLINE",
  "command": {
    "id": "RemoveCounterLine",
    "params": {
      "name": "CounterLine 1"
    }
  }
}
```

RemoveCounterLine response message

```
{
  "reply":"OK"
}
```

SetCounterLinesEnabled command message

SetCounterLinesEnabled command message can be used to set the status of (all) CounterLines on server, whether they are enabled (true) or disabled (false).

```
{
  "group": "COUNTERLINE",
  "command": {
    "id": "SetCounterLinesEnabled",
    "params": {
      "enabled": true
    }
  }
}
```

SetCounterLinesEnabled response message

```
{
  "reply":"OK"
}
```

PTZ Camera Configuration APIs

Here is a list of PTZ Camera Configuration APIs, followed by an explanation of each one and an example of the call and response.

- AddCamera
  - AddCamera Polygon FOV
  - AddCamera Sector FOV
  - AddCamera Rectangle FOV
- EditCamera
  - EditCamera Polygon FOV
  - EditCamera Sector FOV
  - EditCamera Rectangle FOV
  - EditCamera Sector FOV new_ip
- GetCameras
  - GetCameras request
  - GetCameras reply
- GetCameraModels
- GetCameraTiming
  - EditCameraTiming
- Calibration
  - AddCameraCalibration command msg
  - AddCameraCalibration response msg
- Move Camera
  - SetCameraFollowObject
  - GetCameraFollowObject
  - StopFollowingAll
  - SetHomePos
  - GotoHomePos
- PTZ Control Enabled
  - SetPTZControlEnabled
  - GetPTZControlEnabled
- RemoveCamera
- SendJsonText

PTZ Camera API Requirements

- Requires QORTEX DTC 2.3 or later.
- Requires `config` class.

Note: PTZ Cameras APIs do not work with Playback mode.

AddCamera API

Add a PTZ camera to the QORTEX DTC network. Provide identifying and usage parameters. Select a field of view shape: `Polygon`, `Sector`, or `Rectangle`.

Sample

```
{"group": "PTZ",
"command":
   {"id": "AddCamera","params":
   {"PTZCamera":
   {"FieldOfView":
      {"shape":                            #Choose one of polygon, sector, or rectangle
      "polygon", "Polygon": [[0,0], [0,1], [1,1], [1,0]]},
      "sector","Sector": [0,1.57,32]},
      "rectangle","Rectangle": [1.5708,1,1,64,64]},
   "counter_clockwise_rotation":0,
   "custom":"{\"pan_gain\":1, \"tilt_gain\":1}",
   "ip": "2.3.4.111",
   "name": "test-camera-007",
   "password": "123",
   "pose":
      {"position": [1,2,3],
      "ypr": [0.314159,0.75,0.33]},
   "type": "OnVif",
   "camera_model": "Axis",
   "video_url": "",
   "onvif_activation_url": "",
   "user_name": "user",
   "control": {"home_pan":0,"home_tilt":0,"home_zoom":0}
}}}}
```

EditCamera API

Change configuration of an existing PTZ camera to the QORTEX DTC network. Provide identifying and usage parameters. Select a field of view shape: `Polygon`, `Sector`, or `Rectangle`. Option, `new_ip`, to change the PTZ camera IP address. QORTEX DTC supports both IPv4 and IPv6 IP addresses.

Sample

```
{"group": "PTZ",
"command":
    {"id": "EditCamera","params":
    {"PTZCamera":
    {"FieldOfView":
        {"shape":                          #Choose one of polygon, sector, or rectangle
        "polygon", "Polygon": [[0,0], [0,1], [1,1], [1,0]]},
        "sector","Sector": [0,1.57,32]},
        "rectangle","Rectangle": [1.5708,1,1,64,64]},
    "counter_clockwise_rotation":0,
    "custom":"{\"pan_gain\":1, \"tilt_gain\":1}",
    "ip": "2.3.4.111",
    "new_ip": "2.3.4.112",                 #Edit option to change IP address
    "name": "test-camera-007",
    "password": "123",
    "pose":
        {"position": [1,2,3],
        "ypr": [0.314159,0.75,0.33]},
    "type": "OnVif",
    "camera_model": "Axis",
    "video_url": "",
    "onvif_activation_url": "",
    "user_name": "user",
    "control": {"home_pan":0,"home_tilt":0,"home_zoom":0}
}}}}
```

GetCameras API

Returns an array of PTZ cameras and details: name, FOV, position, orientation, home location, IP address, camera name and credentials, model, and URLs.

Sample request and response

```
{"group": "PTZ",
"command":
   {"id": "GetCameras","params”: {}
}}

{
  "reply": "OK",
  "result": [
    {
   {"PTZCamera":
   {"FieldOfView":
      {"shape":
      "rectangle","Rectangle": [0,0,0,8,8]},
   "counter_clockwise_rotation":0,
   "custom":"{\"pan_gain\":1, \"tilt_gain\":1}",
   "ip": "10.10.10.1",
   "name": "name1",
   "password": "123",
   "pose": {"position": [1,2,3],
      "ypr": [-0,0,-0]},
   "type": "OnVif",
   "camera_model": "Axis",
   "video_url": "",
   "onvif_activation_url": "",
   "user_name": "user",
   "control": {"home_pan":0,"home_tilt":0,"home_zoom":0}
}}},
    {
   {"PTZCamera":
   {"FieldOfView":
      {"shape":
      "rectangle","Rectangle": [0,0,0,8,8]},
   "counter_clockwise_rotation":0,
   "custom":"{\"pan_gain\":1, \"tilt_gain\":1}",
   "ip": "10.10.10.2",
   "name": "name2",
   "password": "123",
   "pose":
      {"position": [1,2,3],
      "ypr": [-0,0,-0]},
   "type": "OnVif",
   "camera_model": "Axis",
   "video_url": "",
   "onvif_activation_url": "",
   "user_name": "user",
   "control": {"home_pan":0,"home_tilt":0,"home_zoom":0}
}}}]}
```

GetCameraModels API

Returns entire JSON with PTZ camera model data to the client.

PTZ camera model data stored in JSON file at the same location as `qortex_client.ini`. PTZ camera model for IP address stored in `qortex_client.ini`. This is added as part of PTZ camera activation. The PTZ camera model is inserted to the model field in QORTEX DTC client after the PTZ camera is activated. The model data populates corresponding fields for PTZ camera editing. For example, streaming and activation URL.

Sample

```
{
    "group": "PTZ",
    "command":
    {"id": "GetCameras","params”: {}
}}
```

GetCameraTiming API

To set the number and frequency of switching between tracked camera objects.

EditCameraTiming message PTZTimeSwitching : true

```
{
   "group":"RULES",
   "command":{
      "id":"EditCameraTiming",
      "params":{
         "PTZTimeSwitching":"true",
         "timeSwitching":{
            "interval":"1",
            "bufferSize":"32"
      }
   }
}
```

EditCameraTiming message with PTZTimeSwitching : false

```
{
   "group":"RULES",
   "command":{
      "id":"EditCameraTiming",
      "params":{
         "PTZTimeSwitching":"false"
      }
   }
}
```

GetCameraTiming message

```
{
   "group":"RULES",
   "command":{
      "id":"GetCameraTiming",
      "params":{
      }
   }
}
```

GetCameraTiming PTZTimeSwitching : true

```
{
   "reply":"OK",
   "result":{
      "PTZTimeSwitching":"true",
      "timeSwitching":{
         "interval":"1",
         "bufferSize":"32"
 }    }    }
```

GetCameraTiming PTZTimeSwitching: false

```
{
   "reply":"OK",
   "result":{"PTZTimeSwitching":"false"}
}
```

PTZ Camera Calibration

AddCameraCalibration API

Updates PTZ camera position information. Returns validation command accepted. Provide changed `position` and/or `ypr` values.

Sample request and response

```
{
  "group": "PTZ",
  "command": {
    "id": "AddCameraCalibration",
    "params": {
      "ip": "2.3.4.55",
      "pose": {
        "position": [1, 2, 3],                                    # Change values
        "ypr": [0.11, 0.22, 0.33]
}}}}

{
  "reply":"OK"
}
```

RemoveCamera API

Removes PTZ camera from sensor network. PTZ camera specified by IP address.

```
{
   "group": "PTZ",
   "command":
      {"id": "RemoveCamera",
      "params":
         {"ip": "1.2.3.222"}
}}
```

PTZ Camera Move Camera

SetCameraFollow API

Sets camera to follow a specific trackable manually. Trackable object ID defined in QORTEX DTC server side.

Sample request

```
{
 "group":"PTZ",
 "command":{
 "id":"SetCameraFollowObject",
 "params":{
   "ip":"10.1.22.28",
   "id":1035,                              # Trackable object ID from server
  "on":1                                   # Set manual tracking on.
}}}
```

GetCameraFollowObject API

Returns the trackable object ID that the PTZ camera is following manually.

Sample request and response

```
{
 "group":"PTZ",
 "command":{
```

```
 "id":"GetCameraFollowObject",
 "params":{
   "ip":"10.1.22.28"
}}}
{
 "reply":"OK",
 "result":{
 "track_id":1035                                  # ID of object being tracked
}}
```

StopFollowingAll API

Stops all the PTZ cameras from following all the QORTEX DTC derived trackable objects.

Sample request

```
{
 "group":"PTZ",
 "command":{
 "id":"StopFollowingAll",
 "params":{ }
}}
```

SetHomePos API

For the specified PTZ camera, set the current Pan/Tilt/Zoom values as the default home location. The PTZ camera returns to the home location when it is idle after movement stops. PTZ camera specified by IP address.

Sample request

```
{
 "group": "PTZ",
 "command": {
   "id": "SetHomePos",
   "params": {
     "ip": "1.2.3.222"
}}}
```

GotoHomePos API

Directs the specified PTZ camera to return immediately to the set default home Pan/Tilt/Zoom location. PTZ camera specified by IP address.

Sample request

```
{
  "group": "PTZ",
  "command": {
    "id": "GotoHomePos",
    "params": {
      "ip": "1.2.3.222"
}}}
```

PTZ Camera Control Enabled

SetPTZ Control API

Enables PTZ camera automatic control. PTZ camera automatically follows trackable objects according to defined rules.

Sample request

```
{
 "group":"PTZ",
 "command":{
   "id":"SetPTZControlEnabled",
   "params":{
   "enabled":1                                        # Value 1 enables automatic tracking.
}}}
```

GetPTZ Control API

Returns the PTZ camera control enabled status.

Sample request and response

```
{
 "group":"PTZ",
 "command":{
   "id":"GetPTZControlEnabled",
   "params":{ }
}}

{
  "reply":"OK",
  "result":{
    "enabled":1                # Enabled status 1: PTZ camera automatically tracking objects.
}}
```

SendJsonText API

This API is used to send a JSON formatted string message from QORTEX DTC client to QORTEX DTC server. This API is used to Add/Remove/Get/enable/disable or move/calibrate PTZ camera settings and rules with help of JSON formatted message.

To configure PTZ camera and rules, see *PTZ Camera Configuration APIs* (page 214).

An example implementation is shown below for adding a PTZ camera, similar approach can be used to edit, remove, get, disable, or enable camera or rules and move or calibrate camera.

```
jsonmsg_addCamera = {
"group": "PTZ",
"command": {
"id": "AddCamera",
"params": {
"PTZCamera": {
"FieldOfView": {
"shape": "polygon",
"Polygon": [
                [
0,
0
 ],
                 [
0,
1
 ],
                 [
1,
1
 ],
                 [
1,
0
 ]
            ]
         },
"counter_clockwise_rotation": 0,
"ip": "2.3.4.111",
"name": "test-camera-007",
"pan_gain": 0.11,
"password": "123",
"pose": {
"position": [
1,
2,
3
 ],
"ypr": [
0.314159,
0.75,
0.33
 ]
          },
"tilt_gain": 0.22,
"type": "OnVif",
"camera_model": "Axis",
"user_name": "user",
"zoom_gain": 0.33,
"control": {
"home_pan": 0,
"home_tilt": 0,
"home_zoom": 0
 }
                }
            }
        }
    }
   ### Converting json request to string format
jsontext = json.dumps(jsonmsg_addCamera)
   ### construct the API call request with the json string text
jsonreq = qtrack_service_pb2.server__message__pb2.JsonMessage(text=jsontext)
jsonresp = stub.SendJsonText(jsonreq, metadata=metadata)
   ## Sending the json request to server and waiting for response
print jsonresp
```

Rules APIs

Here is a list of Rules APIs, followed by an explanation of each one and an example of the call and response.

```
AddRule command msg
      AddRule response msg
EditRule command msg
      EditRule response msg
GetRules command msg
      GetRules response msg
RemoveRule command msg
      RemoveRule response msg
```

Rules API Requirements

- Requires QORTEX DTC 2.3 or later.
- Requires `config` class.

Each Rule command message contains:

- **ID**: message ID, for example AddRule
- **Params**: The params contain:
  - **Type**: The Rule type. Options: `Recording`, `PTZ`, or `NetworkAction`
  - **Order**: The rule order and priority assigned on the client
  - **Name**: The rule name assigned on the client
  - **Zone ID**: The UUID for the Event Zone

- **Track type**: The Violation Classification type. Options: `person`, `vehicle`, `person & vehicle`, or *`none`*
- **Track start**: The condition to start tracking. For rule type PTZ, option: `Enter`
- **Track stop**: The condition to stop tracking. For rule type PTZ, options: `Exit`, `Disappear`, `NewObject`
- **Enabled flag**: Always true for AddRule. EditRule can have `true` (to enable Rule) or `false` (to disable Rule).
- **Polymorphic**: An optional object that can be provided for certain types. For NetworkAction type, this contains `entry_urls` and `exit_urls` fields.

AddRule API

Adds a rule of type PTZ, recording, or network action rule.

Sample request and response

```
{
  "group": "RULES",
  "command": {
    "id": "AddRule",
    "params": {
      "type": "NetworkAction",
      "order": 1,
      "name": "Zone Rule 1",
      "zone_id": "7feb24af-fc38-44de-bc38-04defc3804de",
      "track_type": 8,
      "track_start": "None",
      "track_stop": "None",
      "enabled": true,
      "Polymorphic": {
          #For rule type: network action
        "entry_urls": {
          "url1": "http://url1/entry",
          "url2": "http://url2/entry",
          "url3": "http://url3/entry"
        },
        "exit_urls": {
          "url1": "http://url1/entry",
          "url2": "http://url2/entry",
          "url3": "http://url3/entry"
}}}}}
 // No Polymorphic elements
{
  "group": "RULES",
  "command": {
    "id": "AddRule",
    "params": {
      "type": "Recording",
      "order": 1,
      "name": "Zone Rule 1",
      "zone_id": "7feb24af-fc38-44de-bc38-04defc3804de",
      "track_type": 8,
      "track_start": "Enter",
      "track_stop": "Exit",
      "enabled": true
}}}

{
  "reply":"OK"
}
```

EditRule API

Edits existing rules. All Rule fields except `zone_id` and type can be edited. To change a `zone_id`, send RemoveRule for the old type, then issue an AddRule for the new type.

To disable a Rule, set EditRule > `enabled` to `false`.

To enable a disabled rule, set EditRule > `enabled` to `true`.

Sample request and response

```
{
  "group": "RULES",
  "command": {
    "id": "EditRule",
    "params": {
      "type": "Recording",
      "order": 1,
      "name": "Zone Rule 1",
      "zone_id": "7feb24af-fc38-44de-bc38-04defc3804de",
      "track_type": 8,
      "track_start": "Enter",
      "track_stop": "Exit",
      "enabled": true
}}}

{
  "reply":"OK"
}
```

GetRule API

Returns all existing Rules from server.

Sample request and response

```
{
  "group": "RULES",
  "command": {
    "id": "GetRules",
    "params": {}
}}

{
  "reply":"OK",
  "result":[
    {
      "type":"Recording",
      "order":"0",
      "name":"Zone Rule 0",
      "zone_id":"273100ff-f33b-466b-bba5-5489008dbff0",
      "track_type":"644",
      "track_start":"Enter",
      "track_stop":"Exit",
      "enabled":"true",
      "Polymorphic":{
        "urls":{"url":"https://www.google.com"}
      }
    },
    {
      "type":"PtzCamera",
      "order":"1",
      "name":"Zone Rule 1",
      "zone_id":"3b1c7599-bdd4-4d2a-9bd0-b4f996024a0e",
      "track_type":"388",
      "track_start":"Enter",
      "track_stop":"Exit",
      "enabled":"true"
    },
    {
      "type":"NetworkAction",
      "order":"2",
      "name":"Zone Rule 2",
      "zone_id":"861810e6-6d28-4e07-b0e8-d73093f9d0a4",
      "track_type":"644",
      "track_start":"Enter",
      "track_stop":"Disappear",
      "enabled":"true",
      "Polymorphic":{
        "entry_urls":{
          "url1": "http://url1/entry",
          "url2": "http://url2/entry",
          "url3": "http://url3/entry"
        },
        "exit_urls":{
          "url1": "http://url1/entry",
          "url2": "http://url2/entry",
          "url3": "http://url3/entry"
}}}]}
```

RemoveRule API

RemoveRule command message contains details about an existing Rule to be removed.

```
{
  "group": "RULES",
  "command": {
    "id": "RemoveRule",
    "params": {
      "type": "NetworkAction",
      "name": "Zone Rule 1",
      "zone_id": "7feb24af-fc38-44de-bc38-04defc3804de"
    }
 } }
```

Event Zone Rule Type: Object Stitching

When an object disappears then reappears within a zone and object stitching is enabled, Qortex DTC assigns the re-found object the same object ID it was previously assigned.

Object stitching identifies objects that have temporarily disappear then reappeared. Object stitching then updates the newly found object ID to match its original object ID.

Object Stitching parameters

```
<stitching_distance_threshold>10.0</stitching_distance_threshold>
<!--This parameter is the distance threshold in meters. Any new appeared id with a
distance < threshold from the disappeared object is considered for stitching. -->

<stitching_dead_time_threshold>10</stitching_dead_time_threshold>
<!--This parameter is the time threshold in seconds to remember the disappeared
objects, after this time, we throw away the disappeared object and never consider
for stitching.-->

<stitching_add_dead_objects_to_trackable>false</stitching_add_dead_objects_to_track
able>
<!--This boolean if set to true, adds disappeared objects in the trackable_map.-->

<stitching_classification_type_check>644</stitching_classification_type_check>
```

```
<!--This parameter is a bitset which specifies which classes are considered for
stitching. Currently supported:
{any(value: 0), human(value: 644), vehicle(value: 388), human&vehicle(value: 900),
unknown(value: 1)}-->
```

EditObjectStitchingParams Message

```
{
   "group":"RULES",
   "command":{
      "id":"EditObjectStitchingParam",
      "params":{
         "stitching_distance_threshold":"7.0",
         "stitching_dead_time_threshold":"100",
         "stitching_add_dead_objects_to_trackable":"false",
         "stitching_classification_type_check":"0"
      }
   }
}
```

GetObjectStitchingParams Message

```
{
   "group":"RULES",
   "command":{
      "id":"GetObjectStitchingParam",
      "params":{
      }
   }
}
```

GetObjectSticthingParams Result

```
{
   "reply":"OK",
   "result":{
      "stitching_distance_threshold":"7.0",
      "stitching_dead_time_threshold":"100",
      "stitching_add_dead_objects_to_trackable":"false",
      "stitching_classification_type_check":"0"
      }
   }
}
```

Security APIs

gRPC Push notification server requests to the client.

Security State change notification

Security State Push Notification to specific or all clients

```
 // Push notification to one or all clients for "the system is secured" or
"credentials change" or both
 std::stringstream ss;
 ss << "{
   "\"group\" : \"Security\","
   "\"command\" :  {"
    "\"id\" : \"State\","
    "\"params\": {"
     "\"secured\" : ";
 ss << (secured ? "\"v2\"" : "\"none\"");
 if (credentialsChange) {
  ss << ", "
     "\"pwd_change\" : \"true\"";
 }
 ss << "}"
   "}"
  "}";

"secured" state can be now either "v2" or "none" which means was secure for version
2 or not secured at all
"pwd_change" state set to "true" for the case when the password was changed for the
user by the admin
```

Security User Account Expiring notification

Security Account Expiring Push Notification to specific client

```
// Push notification to one client for "the account is expiring in given time
[hours]"
std::stringstream ss;
ss << "{"
  "\"group\" : \"Security\","
  "\"command\" :  {"
   "\"id\" : \"Expiring\","
   "\"params\": {"
    "\"hours\" : ";
ss << hours;
ss << "}"
  "}"
 "}";
```

Security User Account Expired notification

Security Account Expired Push Notification to specific client

```
// Push notification to one client for "the account is expired"
"{"
  "\"group\" : \"Security\","
  "\"command\" :  {"
    "\"id\" : \"Expired\""
  "}"
"}";
```

Settings Configuration APIs

Here is a list of Zone Configuration APIs, followed by an explanation of each one and an example of the call and response.

```
AddSettings
GetSettings
GetSettingsTemplateList
GetSimplifiedSettings
SetSettings
Settings
```

AddSettings API

Used to create a single-sensor or multi-sensor location. This API call deletes the existing settings file in the locations folder and creates a new settings file with the help of the provided template. The `AddSettings` API is a bidirectional stream API. This API has two request modes. One is for creating a single-sensor location (it has information for only one sensor/lidar). The other is for creating a multi-sensor location (it has multiple sensors/lidars) from the Transformation xml file (calibration file) or from the `settings.ini` file (qguard file - legacy `settings.ini` file), which has information for multiple sensors/lidars.

Paired with template selection, calibration file, or single sensor name and IP address, it creates settings file on server.

1. **To create a single-sensor location**, the API call needs the sensor IP address, sensor name (user-given name), and the settings template name (`general_tracking_settings.xml` or `person_tracking_settings.xml` or `vehicle_tracking_settings.xml`).
2. **Before creating a multi-sensor location**, the `calibration.xml` / `transform.xml` file or `settings.ini` should be present in the locations folder. To upload a `settings.ini` or `transformation.xml` file to locations folder, use the `PutFile` API call.
3. **To create a multi-sensor location**, the API call needs the template name, world lidar name, list of lidars, calibration file available, bool value, and qguard settings available bool value. When creating a multi-sensor location from the `transform.xml` or `calibration.xml` file, list of lidars that needs to be picked from the file can be provided,

if the location is created from `qguard.settings.ini` file, the list of lidars has no effect, since all lidars in the `qguard.settings.ini` file are taken into account when creating the `settings.xml` file.

4. **Once uploaded to the location folder**, the transform/calibration xml file changes its name to `qortex.calib.temp.xml` and the `qguard.settings.ini` file changes its name to `qortex.qgsettings.temp.ini`. When these newly renamed files are available in the locations folder, the `AddSettings` API can create a location, or else it produces an error.

5. **The AddSettings API call is made** as a stream API due to multi-sensor location creation functionality. For multi-sensor location creation, a `qguard.settings.ini` file or calibration file should be available, so the `AddSettings` API call initiates the stream from client with location information and client status as `SENT_LOCATION_INFO` first, then makes a `PutFile` API call to upload the relevant `qguard.ini` or calibration file to the server, then makes an `AddSettings` API call with location info and client status while `SENT_ALL_DATA` is performed. Once this three-step process is done, a multi-sensor location `settings.xml` file gets created in the locations folder.

Send and response: `AddSettings`

```
rpc AddSettings(stream AddSettingsRequest) returns (stream AddSettingsResponse) {}
```

Sample: Required options

qguard settings ini file	calibration_file_available = False	qguard_settings_available = True
calibration/ transform xml file	calibration_file_available = True	qguard_settings_available = False

Sample: For creating single-sensor location.

```
def addSinglelocation_request(template, sensorName, sensorIP):
 single_sensor_loc =
 qtrack_service_pb2.AddSettingsRequest.SingleSensorLocationInfo(
  settings_template_name=template, sensor_name=sensorName,
  sensor_ip=sensorIP)
   # Create a request to add single-sensor location
 yield qtrack_service_pb2.AddSettingsRequest(
  single_sensor_loc=single_sensor_loc,
  client_status= 'SENT_LOCATION_INFO')
   # Send the single-sensor location data with client status as SENT_LOCATION_INFO'
 yield qtrack_service_pb2.AddSettingsRequest(
  single_sensor_loc=single_sensor_loc, client_status='SENT_ALL_DATA')
   # Send the single-sensor location data with client status as SENT_ALL_DATA'
def addSinglelocation(template, sensorName, sensorIP, metadata, stub):
 addsingle = stub.AddSettings(addSinglelocation_request(
  template, sensorName,sensorIP), metadata=metadata, timeout=5)
  for r in addsingle:
    print("Create Single Location out is " + str(r))
```

Sample: For creating a multi-sensor location.

```
def addMultiplelocation_request(template, worldlidar,
 list_of_lidars, calibration_file, qguard_settings):
  li = tuple(list_of_lidars.split(","))
  multi_sensor_loc =
  qtrack_service_pb2.AddSettingsRequest.MultiSensorLocationInfo(
   settings_template_name=template, world_lidar_name=worldlidar,
   lidars=li, calibration_file_available=calibration_file,
   qguard_settings_available=qguard_settings)
 yield qtrack_service_pb2.AddSettingsRequest (
  multi_sensor_loc=multi_sensor_loc,
  client_status=ENT_LOCATION_INFO')
    # Send the single-sensor location data with client status as SENT_LOCATION_INFO'
```

Sample: For multi-sensor location, After `client_status` as `SENT_LOCATION_INFO`, call putfile API to upload the qguard or calibration file to server and then send `client_status` as `SENT_ALL_DATA`.

```
yield qtrack_service_pb2.AddSettingsRequest(
 multi_sensor_loc=multi_sensor_loc,
 client_status='SENT_ALL_DATA')
    #send the single-sensor location data with client status as SENT_ALL_DATA'
def addMultiplelocation(template, worldlidar,list_of_lidars,
 calibration_file,qguard_settings, metadata, stub):
  addmulti = stub.AddSettings
(addMultiplelocation_request
(template, worldlidar, list_of_lidars, calibration_file, qguard_settings),
metadata=metadata, timeout=5)
   for r in addmulti:
 print("Create Multi Location out is " + str(r))
```

Sample: `AddSettings` API call and response in protobuf format is:

```
rpc AddSettings (stream AddSettingsRequest) returns
(streamAddSettingsResponse) {}
message AddSettingsRequest {
 message MultiSensorLocationInfo {
 string settings_template_name = 1;
 string world_lidar_name    = 2;
 repeated string lidars    = 3;
 bool calibration_file_available = 4; /* The calibration file in xml
format */
 bool qguard_settings_available = 5; /* The calibration file in ini format
*/
 }
 message SingleSensorLocationInfo {
```

```
 string settings_template_name = 1;
 string sensor_name   = 2;
 string sensor_ip = 3;
 }

 enum ClientStatus {
 STARTED   = 0;
 SENT_LOCATION_INFO = 1;
 PROCESSING   = 2;
 SENT_ALL_DATA  = 3;
 FINISHED   = 4;
 CLIENT_FAILED  = 5;
 }

 oneof LocationInfo {
 MultiSensorLocationInfo multi_sensor_loc = 1;
 SingleSensorLocationInfo single_sensor_loc = 2;
 }
 ClientStatus client_status  = 3;
}
```

GetSettings API

Used to get/fetch the location settings or publisher settings from `settings.xml` file. This API call has two sections. One section fetches location data such as the sensor name, sensor IP, and transformation information from the `settings.xml` file. The other section fetches the publisher information such as port number, port name, publishing format, `adddatasize` bool value, and `networkbyteorder` bool value from the `settings.xml` file.

Add type in `gRPC GetSettings` when it calls `SENSOR_LOCATION` section

Use `<maxBinDistance>` parameter to just be the max range from any sensor that they care about.

Send and response: `GetSettings`

```
rpc GetSettings(GetSettingsRequest) returns (GetSettingsResponse) {}
```

Sample

```
getSetReq =
qtrack_service_pb2.GetSettingsRequest(section=<PUBLISHER or SENSOR_LOCATION>)
getSetResp = stub.GetSettings(getSetReq, metadata=metadata)
print "The Get Settings response is: \n", getSetResp
```

Sample: `GetSettings` API call and response in protobuf format:

```
rpc GetSettings (GetSettingsRequest) returns (GetSettingsResponse) {}
message GetSettingsResponse {
 // SENSOR_LOCATION parameter
 message Sensor {
 string name  = 1;
 string ip  = 2;
 Vector3 position = 3;

 message Transform {
  string from_name = 1;
  string to_name = 2;
  oneof Orientation {
   Quaternion quaternion = 3;
   EulerYPR euler_ypr = 4;
  }
 }
 Transform transform = 5;
 string sensor_type = 6;
 }

 message SensorLocation {
 repeated Sensor sensor = 1; // oneof doesn't support repeated field
 }

 SettingsSection section = 1;
 oneof SettingsResponse {
 SensorLocation sensor_location = 2;
 }
}
 // Publisher parameter
  message Publisher {
  string name               = 1;
  OutputFormat format       = 2;
  uint32 port               = 3;
  bool   add_data_size      = 4;
  bool   network_byte_order = 5;
  }
  message Publishers {
    repeated Publisher publisher = 1;
  }

  SettingsSection section = 1;
  oneof SettingsResponse {
    SensorLocation sensor_location = 2;
    Publishers publishers = 3;
  }
}
```

GetSettingsTemplateList API

Used to fetch the template names for a `settings.xml` file. Available in configuration mode. Three templates can be used to create a `settings.xml` file. This API fetches the template names: `general_tracking_settings.xml`, `person_tracking_settings.xml`, and `vehicle_tracking_settings.xml`.

Send and response: `GetSettingsTemplateList`

```
rpc GetSettingsTemplateList(Empty) returns (GetSettingsTemplateResponse) {}
```

Sample

```
templReq = qtrack_service_pb2.Empty()
templResp = stub.GetSettingsTemplateList(templReq, metadata=metadata)
print "Templates are: ", templResp.settings_template
```

GetSimplifiedSettings API

`GetSimplifiedSettings` message is sent by client to request the content of simplified settings JSON file.

GetSimplifiedSettings command

```
{
  "group": "SIMPLIFIED_SETTINGS",
  "command": {
    "id": "GetSimplifiedSettings",
    "params": {}
  }
}
```

Sample response

GetSimplifiedSettings response message (Successful)

```
{
  "reply": "OK",
  "result": {
    "name": "SimplifiedSettings",
    "parameters": [
      {
        "name": "maxBinDistance",
        "value": {
          "path": "BackgroundFilter.maxBinDistance",
          "display_name": "Max Scanned Range",
          "description": "The maximum distance to which the sensor distinguishes
between the background (static points) and the foreground (moving points)."
        }
      },
      {
        "name": "useNanAsUnobstructed",
        "value": {
          "path": "BackgroundFilter.useNanAsUnobstructed",
          "display_name": "Open Space",
          "description": "Select True when it is possible that there is open space
beyond the maximum range of the lidar (e.g. in an open outdoor space) and False
when the environment constrains the maximum range of the lidar (e.g. indoors)."
        }
      },
      {
        "name": "cellSize",
        "value": {
          "path": "ClusterPipeline.CCClusterer.cellSize",
          "display_name": "Min Distance Between Objects",
          "description": "Detected objects below this distance will be merged
together."
        }
      },
      {
        "name": "minNumClusterPoints",
        "value": {
          "path": "ClusterPipeline.ClusterFilter.minNumClusterPoints",
          "display_name": "Object Detection Sensitivity",
          "description": "Sets the minimum number of points for a cluster to be
considered an object. A low value can increase sensitivity which can increase false
positives while a high value can increase the likelihood of missing objects."
        }
      },
      {
        "name": "minSize",
        "value": {
          "path": "OutputFilter.minSize",
          "display_name": "Minimum Object Size",
          "description": "Objects whose largest dimension is smaller than this are
filtered out."
        }
      },
      {
        "name": "maxSize",
        "value": {
          "path": "OutputFilter.maxSize",
          "display_name": "Maximum Object Size",
          "description": "Objects whose largest dimension is larger than this are
filtered out."
        }
      },
      {
        "name": "maxArea",
        "value": {
          "path": "OutputFilter.maxArea",
          "display_name": "Maximum Object Area",
          "description": "Objects whose area (length * width) is larger than this
are filtered out."
        }
      },
      {
        "name": "minSpeed",
        "value": {
          "path": "OutputFilter.minSpeed",
          "display_name": "Minimum Object Speed",
          "description": "Objects whose speed (m/s) is lower than this are filtered
out."
        }
      },
      {
        "name": "maxSpeed",
        "value": {
          "path": "OutputFilter.maxSpeed",
          "display_name": "Maximum Object Speed",
          "description": "Objects whose speed (m/s) is higher than this are
filtered out."
        }
      }
    ]
  }
}
```

SetSettings API

Used to set the publisher settings in the `settings.xml` file. This API is used to set the QORTEX DTC object list, zone list, sensor state and QORTEX DTC state publishing port number, format value, data size bool value, and network byte order bool value. The API response is a bool value.

`SetSettings` allows values to be set when the when the location is not running. To set values, stop the location and run `SetSettings` API. The `SetSettings` API does not allow multiple channels to be set on the same Port, this means the Port numbers must be unique.

Send and response: `SetSettings`

```
rpc SetSettings(SettingsSectionResponse) returns (RequestResult){}
```

Sample

```
def sub_setsettings():
   yield qtrack_service_pb2.SettingsSectionResponse.Publisher(name="Qortex
1_OBJECT_LIST", port=17171, format= "XML", add_data_size =
True, network_byte_order = True)
   yield qtrack_service_pb2.SettingsSectionResponse.Publisher(name= "Qortex
1_ZONE_LIST", port=17172, format= "JSON", add_data_size =
True, network_byte_order = True)
   yield qtrack_service_pb2.SettingsSectionResponse.Publisher(name=
"QTRACK_OBJECT_LIST", port=17161, format= "XML", add_data_size =
True, network_byte_order = False)
   yield qtrack_service_pb2.SettingsSectionResponse.Publisher(name=
"SENSOR_HEALTH_STATE", port=17168, format= "PROTOBUF", add_data_size =
False, network_byte_order = False)
   yield qtrack_service_pb2.SettingsSectionResponse.Publisher(name= "Qortex
1_STATE", port=17178, format="NONE", add_data_size = True, network_byte_order
= True)

setsetting_req =
qtrack_service_pb2.SettingsSectionResponse.Publishers
(publisher=sub_setsettings())
requ =
qtrack_service_pb2.SettingsSectionResponse
(section="PUBLISHER",publishers=setsetting_req)
resp = stub.SetSettings(requ, metadata=metadata)
print resp.result
```

Settings API

Used to set or get the values of various parameters in the `settings.xml` file. The API has `command = SET_COMMAND` or `GET_COMMAND` and `format = XML_FORMAT` or `JSON_FORMAT`. if the format is xml, the stream of data should be in xml format. If the format is JSON, the stream of data should be in JSON format. The stream data should be formatted according to the `settings.xml` tag hierarchy.

Send and response: `Settings`

```
rpc Settings(SettingsRequest) returns (SettingsResponse) {}
```

Sample

```
setreq = qtrack_service_pb2.SettingsRequest(
 command=SET_COMMAND or GET_COMMAND>, stream=stream,
 format=<XML_FORMAT or JSON_FORMAT>)
settingresp = stub.Settings(setreq, metadata=metadata)
print "Settings response is: \n", settingresp
```

State Module

The state module includes two simple unary RPCs. State gRPC API commands enable access to a report of the status of the server, zone, or trigger-based alarm. This module has the following APIs.

```
GetLicenseInitState                    GetRulesEnabled
GetLicenseState                        GetSensorState
GetPTZEnabled                          GetServerState
```

Sample Python code for using a State API, see *Figure 111. Sample Python Code: State API*.

```
def getserverstate(metadata, stub):
   """
   API call to get server state
   :param stub: stub object
   :param metadata: metadata to be sent
   :return: the server current state info such as server mode, version, in
      capturing, is paused, etc. .
   """
   state_req = qtrack_service_pb2.Empty()   # Creating an empty request
   state_resp = stub.GetServerState(state_req, metadata=metadata)
      # Sending the request to server via stub and getting response
   print "Sever state is :\n", state_resp
   return state_resp

def getzone metadata, stub, zonetype):
   """
   API call to get server state
   :param stub: stub object
   :param zonetype: type of zone (EVENT or EXCLUSION)
   :return: zone data such as zone name, UUID, class, vertices, etc. .. of all the zone
      (Event/Exclusion)
   """


   getzone_req = zone_requests_pb2.ZonesRequest(zone_class=zonetype)
      # Creating a request with zone_class=<required zone class>
   getzone_resp = stub.GetZones(getzone_req, metadata=metadata)
      # Sending the request to server via stub and getting response
   print "The zone info is ", getzone_resp
   return getzone_resp

if __name__ == '__main__':

   channel = grpc.insecure_channel("192.168.0.1”17177")
      # Creates an insecure Channel to a server
```

```
    stub = qtrack_service_pb2_grpc.QortexServiceStuf(channel)
        # Creating stub. Client sends a request to the server using the stub and waits for a response
        # to come back, just like a normal function call
    client_id = "82ef205c_ed14-4013-89cf-85e11b6827b1"
        # An example client ID
    metadata = (("client_id", client_id),)
        # Optional metadata to be transmitted to the service_side. Here metadata with client ID is a
        # must
```

Figure 111. Sample Python Code: State API

Each API in the status module is explained below, followed by an example of the call and response.

GetLicenseInitState, GetLicenseState API

```
rpc GetLicenseInitState(GetLicenseInitStateRequest) returns (RequestResult) {}
rpc GetLicenseState(GetLicenseStateRequest) returns (RequestResult) {}
message GetLicenseInitStateRequest {
string app_name = 1;
string app_version = 2;
int32 app_license_product_id = 3;
}
message GetLicenseStateRequest {
int32 sensor_count = 1;
string license_type = 2;
}
```

Suggest adding the number sensors per license as a custom parameter set, for example, version and tier. The implementation must allow FAEs to input the number of sensor licenses in the customer parameter field when the license is created. User defined field entry would no longer be needed.

GetPTZEnabled API

Get PTZ enabled (enabled by license). Works in either configuration or monitor mode. Requires QORTEX DTC 2.3 or later.

This API is used to get PTZ camera enabled value. The enabled value is `True` or `False`. This value is retrieved from the License of the Qortex-Server.If the `ptz_Count >0` this API returns `True`, else returns `False` . To check the License verify the command below.

```
./Qortex-Server --license info
```

Sample

```
req = qtrack_service_pb2.Empty()
resp = stub.GetPTZEnabled (req, metadata=metadata)
print "The Get PTZEnabled 'result' value is : ", resp.result
print "The Get PTZEnabled 'enabled' value is : ", resp.enable
```

Send and response

```
rpc GetPTZEnabled (Empty) returns (EnableResult) {}
```

GetRulesEnabled API

Get rules enabled (enabled by license). Works in either configuration or monitor mode. Requires QORTEX DTC 2.3 or later.

This API is used to get the rules enabled value true or false. When the Add-Ons in license is having values as `Rules` then this API respond `True`, if the `Add-Ons` value is empty in license, then this API shall respond `False`. Location has to be running in order to get the True or False response. To check the license `Add-Ons` type below command.

```
./Qortex-Server --license info
```

Sample

```
req = qtrack_service_pb2.Empty()
resp = stub.GetRulesEnabled (req, metadata=metadata)
print "The Get rule status 'result' value is : ", resp.result
print "The Get rule status 'enabled' value is : ", resp.enable
```

Send and response

```
rpc GetRulesEnabled (Empty) returns (EnableResult) {}
```

GetSensorState API

Used to get the health status such as sensor temperature, frame rate, NMEA status, and error codes, as well as sensor model, name, IP address, PTZ camera state, and connection status (`connected` or `disconnected`). The API accepts a list of sensor IP addresses and responds back with its status.

Send and response

```
rpc GetSensorState(GetSensorStateRequest) returns (GetSensorStateResponse) {}
```

Sample

```
SensorReq = qtrack_service_pb2.GetSensorStateRequest(
 sensor_ip_list=<list of sensor ip>)
 sensorResp = stub.GetSensorState(SensorReq, metadata=metadata)
print "Get sensor state response is: ", sensorResp
```

GetServerState API

Used to get the current state of the server. An empty request from the `GetServerState` API to the server produces a response stating the **server** mode (`Live` or `Playback`), settings file path, version, config client port number, monitor client port number, `is_paused` bool value (data set indicating whether it is paused or not), is_capturing bool value (whether a location is running live or not), and a qlog name (whether a data set is running).

Send and response

```
rpc GetServerState(Empty) returns(ServerState) {}
```

Sample

```
req=qtrack_service_pb2.Empty()
resp=stub.GetServerState(req, metadata=metadata)
print "Status request is :\n", resp
```

Streaming Module

The streaming module includes bidirectional read-write RPCs. See *Bidirectional Streaming RPCs* (page 192). These APIs send a request to the server to download or upload files with a specialized type. Each API in the streaming module is explained below, followed by an example of the call and response. The APIs for this module are:

```
GetFile
PutFile
putfile_request
```

GetFile API

Used to Get/Download a file from the server to the client. The GetFile API call works in monitor mode as well, so the SwitchToConfigMode API call does not need to run in parallel. In GetFile API call it is possible to download only GENERAL_FILE and LOG_FILE (qortex.server.log). When the API call has FileType= “LOG_FILE”, the server searches for the qortex.server.log file in the /home/quanergy/quanergy/qortex folder location, and if the file is found, it sends it to the client, or it produces a file-not-found error. The construction of an API call for GetFile is very similar to the PutFile API call.

1. In GetFile API call, Construct first level of request with file type and file name (file info) and send to server, the server responds back with total number of chunks client shall receive.

```
file_info = qtrack_service_pb2.FileChunk.FileInfo(file_type=filetype,
file_name=Getfilename)
return qtrack_service_pb2.FileChunk(file_info=file_info)
```

2. After the chunk count is received, Construct the second level of request with file info and chunk index and send to server and get the raw data as response from server.
3. After the 1st chunk of raw data is received, send next request with file info and incremented chunk index, repeat this until all the chunk index are sent and its corresponding raw data is received.

Send and response

```
rpc GetFile(stream FileChunk) returns (stream FileChunk) {}
```

If the `GetFile` points to a directory of files, such as with the user logs, a userLogs.zip file is returned.

Complete sample python code for `GetFile` API call

```
def getFileChunk(Getfile):
    get_filetype = getfiletype(Getfile)
        ## getting file type from file name (refer PutFile API function for its def
    file_info = qtrack_service_pb2.FileChunk.FileInfo(file_type=get_filetype,
file_name=Getfile)
        ## First level of request
    return qtrack_service_pb2.FileChunk(file_info=file_info)
def getFileChunk_count(Getfile):
     yield getFileChunk(Getfile)
def getfile_request(chunk_count, Getfile):
     yield getFileChunk(Getfile)
        ## Second level request sending file info
    for i in range(chunk_count):
          yield qtrack_service_pb2.FileChunk(chunk_index=
(qtrack_service_pb2.FileChunkIndex(chunk_index=i)))
            # Seconds level of request sending chunk index
def getFile(Getfile,metadata, stub):
    ##### Main Function #####
     chunk_count = 0
     items = stub.GetFile(getFileChunk_count(Getfile), metadata=metadata)
        # Send first level of request to server and get total chunk of file data, calling getFileChunk_count
function
    for item in items:
        chunk_count_str = str(item)
        try:
            chunk_count_str = chunk_count_str.split("\n")[4]
            chunk_count = int(filter(str.isdigit, chunk_count_str))
                # Extracting chunk count from the received response
        except:
            pass
        print "The total chunk count to be received is : {} \n
\n".format(chunk_count)
    getfileout = stub.GetFile(getfile_request(chunk_count, Getfile),
metadata=metadata, timeout=50000)
        ## Seconds level of request with chunk count(chunk index) and file info
    for resp, i in zip(getfileout, range(-1,chunk_count+2)):
        ## Getting raw data response
    print resp
        # Printing response
    text = str(resp).split("\n")[1].strip()
        # separating raw data snd chunk index and printing chunk index
    if i == 0 or i == -1 or "data" in text or "chunk_count" in text:
        continue
    else:
        searchText = "chunk_index: " + str(i)
    if searchText in text:
        print ('correct chunk index received : {}\n \n'.format(text))
    else:
        print "inconsitent chunk index"
```

PutFile API

Used to put a file or upload a file from the client host to the server host. `PutFile` API needs the client to be in config mode, so the `SwitchToConfigMode` API should be running in parallel when calling a `PutFile` API stream. `PutFile` API uploads only general file (txt), `qguard.settings.ini` file, and the `calibration.xml` file to a default `/home/quanergy/quanergy/qortex` filepath. This folder location cannot be changed.

Send and response

```
rpc PutFile(stream FileChunk) returns (stream FileChunkIndex) {}
```

Sample

```
CALIBRATION_FILE = 0;
  /* can only be uploaded from client to server; for location setup purpose */
GUARD_SETTINGS_FILE = 1;
  /* can only be uploaded from client to server; for location setup purpose */
LOG_FILE = 2;
  /* can only be downloaded from server to client */
RECORDING_QLOG_FILE = 3;
  /* can only be downloaded from server to client */
GENERAL_FILE = 4;
  /* can be downloaded/uploaded from/to server to/from client, such as text file, deb package, zip file, etc.. */
```

putfile_request API

Uploads QORTEX DTC `settings.ini` file to locations folders and changes the file name to `qortex.settings.temp.ini`, then uploads `calibration.xml` file to the location folder and changes the file name to `qortex.calib.temp.xml`. A target file name cannot be set for calibration file and `qguard.settings.ini` file. Uploading a file with `PutFile` API requires the following process:

1. **The client side checks** if the file exits and can be opened; if so, get the file name and total chunks to send; total chunks = file_size/chunk size for each message (by default 16 kb, but it can be 32 kb or 64 kb). Split the file into these smaller chunks. Now the split file has n number of chunks for an n chunk index. For example, a 128 kb file is split into

smaller chunks with each chunk size at 16 kb, so there are 8 chunks of data, so chunk index is 8.

2. **Construct first level of request** with

```
FileChunk.FileInfo(
file_type=
   <CALIBRATION_FILE | QGUARD_SETTINGS_FILE | LOG_FILE |
   RECORDING_QLOG_FILE |GENERAL_FILE>,
file_name=<TargetFileName>,
chunk_count=<number of chunks>
)
```

   Then send this first level of request to server.

```
inside putfile_request() function
 file_info = qtrack_service_pb2.FileChunk.FileInfo(
 file_type='CALIBRATION_FILE', chunk_count=8)
 yield qtrack_service_pb2.FileChunk(file_info=file_info)
```

3. **Construct a second level of request** with the split data/chunk data along with its chunk index and send them to the server.

```
inside putfile_request() function
dataList contains the split data as list format
for data, i in zip(dataList, range(8)):
 print "The value of i is ", i
 print "starting chunk {} .......\n".format(i)
 raw_data = qtrack_service_pb2.FileChunk.RawData(
  data=data, chunk_index=i)
 yield qtrack_service_pb2.FileChunk(raw_data=raw_data)
 print "finished chunk {} .......\n".format(i)
```

4. **The server sends a response** with the expected chunk index that the client is going to send (should start with 0). Client then reads 16 KB binary data from the file and puts them into the request together with the chunk index to send. Meanwhile, the client checks if all chunks have been sent (expected_chunk_index >= total chunk index). If so, the client closes the connection. (This step may repeat multiple times depending on the total size to upload.)

```
items = stub.PutFile(putfile_request(dataList, filePath), metadata=metadata)
 for item in items:
 print "the response is {} \n".format(item)
```

Sample PutFile and putfile_request code. If the code is not formatted properly, check if there may be missing tab or multiple spaces/tab.

```
def defaultfiletype(fileExt):
   if fileExt == "xml":
      filetype = "CALIBRATION_FILE"     # 0
   elif fileExt == "ini":
      filetype = "QGUARD_SETTINGS_FILE" # 1
   elif fileExt == "log":
      filetype = "LOG_FILE"             # 2
   elif fileExt == ".q*":
      filetype = "RECORDING_QLOG_FILE"  # 3
   else:
      filetype = "GENERAL_FILE"         #  4
   return filetype
def getfiletype(filename):
   try:
      filenameExt = filename.rsplit(".", 1)[1]
      get_file_type = defaultfiletype(filenameExt)
      return get_file_type
   except Exception as err:
   print err
def read_in_chunks(file_object, chunk_size=1024):
   # first function call from main
   """Default chunk size: 1k."""
   while True:
      dataOut = file_object.read(chunk_size)
   if not dataOut:
      break
   yield dataOut
def putfiletype(file_Path):
   # function call from putfile_request function
   try:
      TargetFileName = file_Path.rsplit("/", 1)[1]
         # from the file path, getting the file name
   filenameExt = (file_Path.rsplit("/", 1)[1]).split(".")[1]
      # from file path getting file extension eg .txt file or . ini file or . xml
file, etc..
   put_file_type = defaultfiletype(filenameExt)
      # Getting the file type (Calibration_file or Log_File etc..) from the file
extension
   return put_file_type, TargetFileName
      # return the file type and file name to caller function
   except Exception as err:
   print err
def putfile_request(dataList,filePath):
   # Second function call from main
   numOfChunk = len(dataList)
      # From dataList list get total number chunks
   put_file_type, TargetFileName = putfiletype(filePath)
      # With putfiletype function, extract the file type and file name from the
filepath provided
```

```
    file_info = qtrack_service_pb2.FileChunk.FileInfo (file_type=put_file_type,
file_name=TargetFileName, chunk_count=numOfChunk)
        # constructing first level of request which has fileinfo, filename and total
chunk count
    yield qtrack_service_pb2.FileChunk(file_info=file_info)
        # Send the fileinfo request to server
    for data, i in zip(dataList, range(numOfChunk)):
        # from the dataList and total chunk count, iterate each data chunk and its
chunk index and send them to server one by one
    print "The value of i is ", i
    print "starting chunk {} .......\n".format(i)
        raw_data =
qtrack_service_pb2.FileChunk.RawData(data=data,chunk_index=i)
            # constructing request to send chunked data and its index to server
    yield qtrack_service_pb2.FileChunk(raw_data=raw_data)
        # sending chunked data and data index to server as raw data
    print "finished chunk {} .......\n".format(i)

def putFile(filePath, metadata, stub):
    ######## Main function #########
    dataList = []
        # Create empty list to append the split file content/data
    data = open(filePath)
        # Open the file to read the content
    for piece in read_in_chunks(data, 16000):
        # call read_in_chunks functions which splits the file content in to 16kb
    dataList.append(piece)
        # Add the split file content in dataList list
    items = stub.PutFile(putfile_request(dataList, filePath), metadata=metadata)
        # send the constructed request to server and get chunk index as response
    for item in items:
    print "the response is {} \n".format(item)
```

Zone Configuration APIs

Here is a list of Zone Configuration APIs, followed by an explanation of each one and an example of the call and response.

```
AddZone
EditZone
ObjectStitching
GetZones
GetZonesEnabled
RemoveZone
SetZonesEnabled
ZoneViolation
  ChangeEventZoneViolationAction
  ChangeEventZoneViolationTrigger
  GetEventZoneViolationActions
  GetEventZoneViolationHandlerEnabled
  GetEventZoneViolationRecordingEnabled
  SetEventZoneViolationRecordingEnabled
```

AddZone API

Used to add zone information into a settings file. The Event, Execution, or Inclusion zone information such as zone name, unique id, dimensions, height, type, and enabled parameters are sent via this API to the server and are written to the `settings.xml` file. Creating a zone requires at least three vertices (3 X and Y values). Every zone has a unique UUID, a UUID is 128-bit Universally Unique ID value such as `a1ebb897-ed68-4af9-8f62-cac86512db2b`. Each zone of a certain class should have a unique UUID value. Duplicating UUID values between different `Zone_Class` is allowed (the UUID value of an Event zone can be used as the UUID value for an Exclusion zone), but that is not recommended. When the `Zone_Class` parameter is used to reference a zone that is either an Event zone, Exclusion zone, or Inclusion zone, its values are `EVENT`, `EXCLUSION`, or `INCLUSION`. `Zmin` and `Zmax` represent the minimum/start and maximum/stop of zone height values. The enabled value represents that the zone is visible in the UI client. If false, the zone is not visible in the QORTEX DTC client UI. When the `Zone_Class` parameter is selected as Exclusion zone, then an extra parameter is added to select an exclusion zone sensor.

Send and response: `AddZone`

```
rpc AddZone (AddZoneRequest) returns (RequestResult) {}
```

Sample

```
vertices_list=[]
    # Creating empty list to store zone vertices / dimensions
zone_info =
 zone_3d_pb2.QZone(name=str(name), type= "POLYGON")
    # creating request with zone name and zone type
 for x, y in zip(X_vertices_list, Y_vertices_list):
    # iterating loop of x and y values of zone vertices, a minimum of 3 x and y values are required
 vertices_list.append(qortex_geometric_types_pb2.Vector2(x=float(x),
 y=float(y)))
    # Adding vertices (x and y) to vector and creating request
Zone_vertices =
 qortex_geometric_types_pb2.Polygon2D(vertices=vertices_list)
    # creating a request with vertices list for polygon2
zone_vert =
 zone_3d_pb2.QPolygonGeometry(vertices=Zone_vertices)
    # creating a request with vertices list for QPolygonGeometry
zone_params =
 zone_3d_pb2.QZone3D(uuid=str(UUID), zone=zone_info,
 z_min=float(zMin), z_max=float(zMax), zone_class=str(ZoneClass),
 enabled=str_to_bool(Enabled), polygon_3d=zone_vert,
    # constructing complete zone request with all information such as name, uuid, vertices, type, etc.
zone_req =
 zone_requests_pb2.AddZoneRequest(zone=zone_params)
    # Constructing AddZone request
req_resp = stub.AddZone(zone_req, metadata=metadata)
    # sending addzone request to server and getting bool result as response
print "Add Zone response is: ", req_resp.result
```

EditZone API

Used to edit an already existing or added zone. The `EditZone` API identifies a zone with the help of UUID, so, to edit an existing zone in a `settings.xml` file, the UUID of that zone should be provided. The edit zone request construction is the same as the `AddZone` request construction shown above, but it also calls the `EditZone` API when sending a request to the server. It is not possible to edit the UUID value of a zone, and a new UUID value represents a new zone. Using a UUID that does not exist in a settings xml file produces an error for the `EditZone` API.

Send and response: `EditZone`

```
rpc EditZone (QZone3D) returns (RequestResult) {}
```

Sample

```
req_resp =
 stub.EditZone(zone_params, metadata=metadata)
    # Edit Zone API request with zone_params request construction similar to add Zone API request
construction
print "Edit Zone response is: ", req_resp.result
    # Sending Edit Zone API request to server and getting bool response
```

GetZones API

Used to retrieve all the available zone information from the `settings.xml` file. It is used to retrieve all Event, Exclusion, or Inclusion zones.

```
getzone_req =
 zone_requests_pb2.ZonesRequest(zone_class=<zone_class>)
    # Construct zonerequest with zone class as either 'EVENT' or 'EXCLUSION' or ‘INCLUSION’
getzone_resp =
 stub.GetZones(getzone_req, metadata=metadata)
    # Send the request to server and get zone information as response
print(getzone_resp)
```

GetZonesEnabled API

Used to get the value of enabled property from the `settings.xml` file. If the enabled value is true under the zone in the settings xml file, then the API response is true. If the enabled value is false under the zone in the settings xml file, then the API response value is false.

Send and response: `GetZonesEnabled`

```
rpc GetZonesEnabled (ZonesRequest) returns (ZonesEnableStatus) {}
```

Sample

```
getzoneEnbReq =
 zone_requests_pb2.ZonesRequest(zone_class=zone_class)
getZoneEnbResp =
 stub.GetZonesEnabled(getzoneEnbReq, metadata=metadata)
print "Get Zone Enabled value is: ", getZoneEnbResp.enable
```

RemoveZone API

Used to remove a zone with its associated information from the `settings.xml` file. The remove zone request identifies the zone with its UUID value and the `Zone_Class` value of the zone and removes it.

Send and response: `RemoveZoneRequest`

```
rpc RemoveZone (RemoveZoneRequest) returns (RequestResult) {}
```

Sample

```
zone_info=
 zone_requests_pb2.RemoveZoneRequest(zone_class=ZoneClass, uuid=UUID)
    # Construct a remove zone request with zone_class and UUID
RemoveZoneresp =
 stub.RemoveZone(zone_info, metadata=metadata)
    # send the request to server and get bool response
print "Remove Zone response is: ", RemoveZoneresp.result
```

SetZoneEnabled API

Used to enable (make visible) an EVENT/EXCLUSION zone in QORTEX DTC client UI. If the enabled value is true, the zone is visible in the QORTEX DTC client UI. If the enabled value is false, the zone is not visible in the QORTEX DTC client.

Send and response: `SetZoneEnabled`

```
rpc SetZoneEnabled (EnableZonesRequest) returns (RequestResult) {}
```

Sample

```
setzoneEnbReq =
 zone_requests_pb2.EnableZonesRequest(zone_class=zone_class,
 enable=eval(bool_enable))
   # construct a request with zone class and enable value
setZoneEnbResp =
 stub.SetZonesEnabled(setzoneEnbReq, metadata=metadata)
   # send the request to server and get bool response
print "Set Zone Enabled response is: ", setZoneEnbResp.result
```

ZoneViolation APIs

ZoneViolations listed as deprecated apply to older versions of Qortex DTC.

ChangeEventZoneViolationAction API

Deprecated in QORTEX DTC 2.3. See *AddRule API* (page 225), *EditRule API* (page 226), and *GetRule API* (page 226).

For QORTEX DTC 2.0 or older only: Used to Add or Edit or Remove an Event zone violation action. `OCCUPIED` and `UNOCCUPIED` are the two trigger states for a zone violation. `START_RECORD` and `STOP_RECORD` are the two actions. When a trigger state is raised, the action is performed. If the Event zone violation action is set as `START_RECORD` for an `OCCUPIED` trigger, then when an Event zone becomes occupied, the recording starts.

- **To create an Event zone violation action**, the following are required: command `cmd=ADD`, Event zone UUID, trigger state (`OCCUPIED` or `UNOCCUPIED`), and action (`START_RECORD` or `STOP_RECORD`). (An old action can be ignored when creating a new violation action).
- **To edit an existing violation action**, the following are required: command `cmd=EDIT`, Event zone UUID, trigger state, action, and old action.
- **To remove a violation action**, command `cmd=REMOVE` and zone UUID are required. (When an Event zone is deleted with the RemoveZone API call, the Event zone violation action associated with the removed zone is also deleted.)

```
requ = zone_requests_pb2.ChangeViolationActionRequest(
 cmd=<ADD or EDIT or REMOVE>, zone_uuid=zone_uuid,
 trigger_state=trigger_state, action=action, old_action=old_action)
resp = stub.ChangeEventZoneViolationAction(requ, metadata=metadata)
print "Change Event zone violation action response is: ", resp.result
```

ChangeEventZoneViolationTrigger API

Deprecated in QORTEX DTC 2.3. See *AddRule API* (page 225), *EditRule API* (page 226), and *GetRule API* (page 226).

For QORTEX DTC 2.0 or older only: Used to `EDIT` or `REMOVE` an Event zone Violation trigger. It is not possible to `ADD` a trigger state with this API call. The two trigger states for an Event zone are `OCCUPIED` and `UNOCCUPIED`.

- **To update a trigger state** requires the following: `cmd=EDIT`, zone UUID, trigger state, and old trigger state.
- **To remove a trigger state** requires: `cmd=REMOVE` and zone UUID.

```
requ = zone_requests_pb2.ChangeViolationTriggerRequest(
 cmd=EDIT or REMOVE>, zone_uuid=zone_uuid,
 trigger_state=trigger_state, old_trigger_state=old_trigger_state)
resp = stub.ChangeEventZoneViolationTrigger(requ, metadata=metadata)
print "The change Event zone violation trigger response is: ",
 resp.result
```

GetEventZoneViolationActions API

Deprecated in QORTEX DTC 2.3. See *AddRule API* (page 225), *EditRule API* (page 226), and *GetRule API* (page 226).

For QORTEX DTC 2.0 or older only: Used to fetch Event zone violation action information such as the current trigger state and the current trigger action. To fetch the Event zone violation action requires the Zone UUID.

```
QTRACK-2000 – add cover enable/disable Rules tab
req = zone_requests_pb2.GetEventZoneViolationRequest(uuid=zone_uuid)
resp = stub.GetEventZoneViolationActions(req, metadata=metadata)
print "The Get Event zone violation action value is : ", resp
```

GetEventZoneViolationRecordingEnabled API

Used to get the bool value of the Event zone violation recording enabled parameter from settings xml file.

Send and response: `GetEventZoneViolationRecordingEnabled`

```
rpc GetEventZoneViolationRecordingEnabled (Empty) returns (ViolationRecodingStatus)
{}
```

Sample

```
req = qtrack_service_pb2.Empty()
resp = stub.GetEventZoneViolationRecordingEnabled(
 req, metadata=metadata)
print "The Get Event zone violation recording result value is : ",
 resp.result
print "The Get Event zone violation recording enabled value is : ",
 resp.enable
```

SetEventZoneViolationRecordingEnabled API

Used to set the Event zone recording enabled value true or false. If the Event zone violation recording Enabled value is True, then if a violation occurs, the recording starts. But if its value is False, then if a violation occurs, the recording will not start.

Send and response: `SetEventZoneViolationRecordingEnabled`

```
rpc SetEventZoneViolationRecordingEnabled (ViolationRecodingStatus) returns
(RequestResult) {}
```

Sample

```
requ = zone_requests_pb2.ViolationRecodingStatus(enable=<Bool value>)
resp = stub.SetEventZoneViolationRecordingEnabled(
 requ, metadata=metadata)
print "The set Event zone violation recording enabled response is: ",
 resp.result
```

Appendix 3. Cybersecurity

QORTEX DTC 2.3 adds support for user authentication and data encryption. This is an optional mode for the QORTEX DTC server. By default, **Security** mode is `off`. Enable it through the QORTEX DTC client in **Configuration** mode.

- **HTTP Digest** authentication protocol—Internet access is not required. Certificate of Authority (CA) is not required. Authentication setting applies for the duration of the QORTEX DTC client session.
- **AES-256** encryption—The object list, zone list and sensor health APIs are encrypted.

The QORTEX DTC cybersecurity API enables the entire data flow to be controlled and secured upon request. The flow is controlled with events initiated by the user and delivered through signal-slot connections.

When Security mode is enabled, user authentication is required for the following actions:

- Connecting the QORTEX DTC client to the QORTEX DTC server
- Connecting any third-party client to the QORTEX DTC server
- Sending and receiving gRPC API calls.
- Reading QORTEX DTC API (such as, object list, zone list, sensor health). Anytime an application is reading the QORTEX API TCP ports. The Quanergy TCP listener includes HTTP digest and AES-256 support. See *Server TCP Ports to Monitor* (page 92).

To enable the QORTEX DTC server **Security** feature, install QORTEX DTC server and Debian, and activate or refresh the Quanergy license.

- Every user can secure or unsecure the entire QORTEX DTC client to server system to enable or disable the secured data flow.
  - First user logging in starts the secured encrypted data flow with the new security token.
  - Last user logging out stops the secured encrypted data flow and drops the security token.
  - The QORTEX DTC data session is secured in between new security token created and dropped.

QORTEX HTTP Server

The QORTEX HTTP server provides a command line interface for all QORTEX DTC clients connecting to the QORTEX DTC server. The URL is `https://<IP_Address>:8080/v1/<login>`.

HTTP Digest for Authentication

Hiding credentials. The HTTP authentication *digest* principle enables you to hide all or part of the access credentials. Most commonly, the username is sent by plain text but the password with the client uses calculated MD5 (or other) hash against which the server. The server compares its own hash calculated using the same formula. See Wikipedia article, Digest access authentication.

Two-phased request. HTTP authentication is a two-phased version of HTTP GET request. Briefly, a first phase triggers the server to a nonce (number once value) to enforce the password hash. In the second phase the client sends the password hash with the HTTP GET. The server checks and confirms both client and server hashes match, then the server replies with a standard `HTTP 200 OK` response and completes internal actions to accommodate the client secure session.

QORTEX implementation. In QORTEX DTC, we use custom HTTP header for additional bi-directional communication. The client always sends `Q-Auth` with the command and parameters delimited by (`:`) colon. The server sends the reply as `Q-*`. Where `*` can be multiple variables filled with data depending on the command and its succession.

Application-Level Commands

The following describes the QORTEX DTC custom client headers and server replies.

Every command is sent though its HTTP header. One command per request. Authentication can be with either License ID/ password or username/ password. Recommended preference is to use username authentication and use the license/ID as a backup.

GetSecured Command

Sending the `Q-Auth` header with the `GetSecured` request does not require authentication. The HTTP response sets the `Q-Secured` header to either `true` or `false`. A `true` value means QORTEX DTX is now secured. This means, authenticate the client with `GetToken`.

The `GetSecured` request also sends `Q-UserPresent` header. The HTTP response is either `true` or `false`. A `true` value means the username is set up in the system. The License ID/ License Password is always available, but it is intended for administrative operations.

GetToken Command

Sending the `Q-Auth` header with the `GetToken` request requires authentication. The HTTP response is either `get HTTP 401K unauthenticated`, which means authentication failed due to wrong credentials or the `Q-Token` header is set to a 64-character hex token. This token is then unhexed to AES 256 key. QORTEX DTC Python and C++ TCP Listeners are used for the actual data decryption. Due to listener traits, make sure the of the corresponding port, using `<TCPPublisher><AddDataSize>true</AddDataSize></TCPPublisher>`. The QORTEX DTC client does not need the port configuration for ports in use.

SetSecured Command

Sending the `Q-Auth` header with the `SetSecured:true` or `SetSecured:false` requires authentication. Depending on `true` or `false` after `SetSecured:` the server decides whether to secure the system for `true` or unsecure it for `false`. The HTTP response is `Q-SetSecured` header set to either `ok` or `failed`.

SetCredentials Command

Sending the `Q-Auth` header with the `SetCredentials:username:encryptedPassword`, for encrypted password, see *HTTP Password Encryption Formula* (page 263) and *HTTP User Security Options* (page 263). The QORTEX DTC client enforces username complexity now of 4 or more characters including both letters and digits. The user password complexity is of 6 or more characters and both capital and lowercase letters plus digits required. The HTTP response for `Q-CredentialsSet` can be `true` or `false`. A `true` value indicates the request was successful. This HTTP response includes a security token in `Q-Token` header, as well. See *GetToken Command* (page 256).

Authentication Example

The `QAuthenticator` object is used via the slot for `QNetworkAccessManager::authenticationRequired` signal with `QORTEX:`

QORTEX example authentication

```
/// Basic semantics for passing the credentials [client side code callback]
QNetworkAccessManager nam;
    connect(&nam_, &QNetworkAccessManager::authenticationRequired,
        this, &HttpClient::authRequired);

void HttpClient::authRequired(QNetworkReply* reply, QAuthenticator* authenticator)
{
    Q_UNUSED(reply)
    qDebug() << "Authentication required: " << authenticator->realm();
        // we get realm@quanergy

    authenticator->setUser(user_);
        // Set the user name [or License ID for configuration]
    authenticator->setPassword(pwd_);
        // Set the user password [or License password for configuration]
}
```

Client Authentication Python Example

```
import sys
import requests
import os
import binascii
import socket
import struct
import uuid
import hashlib
import json
from requests.auth import HTTPDigestAuth
from Crypto.Cipher import AES
from Crypto.Util import Counter

def https_authentication(server_ip, username, password):
    #Get authentication url
    url = 'http://' + server_ip + ':8080/v1/login'
    r = requests.get(url)
    print "url", url
    print 'r', r
    #Get another request with authentication header added
    r = requests.get(url, auth=HTTPDigestAuth(username, password),
headers={'Q-Auth': 'GetToken'})
    #Error proofing if url is down or if authentication failed
    if r.status_code != 200:
        print("Could not connect to url, check ip address")
        return False, False
    else:
        if "Authorized" in r.text:
            qtoken = r.headers['Q-Token']
            return r, qtoken
        else:
            print("Could not authenticate, check username or password")
            return False, False
def get_secured(server_ip):
    #Get authentication url
    url = 'http://' + server_ip + ':8080/v1/login'
    r = requests.get(url)
    #Get another request with authentication header added
    r = requests.get(url, headers={'Q-Auth': 'GetSecured'})
    secure_flag = r.headers['Q-Secured']
    return r, secure_flag
def read_tcp(server_ip, port, packet_size):
    # Get data stream
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((server_ip, port))
    msg = s.recv(packet_size)
    return msg
def get_msg_size(msg, uuid_obj):
    """
    - Msg and uuid_obj is extracted from the tcp socket
    - The next 8 bytes after the uuid_obj is the msg size represented as
bytes
    - Split msg with uuid, get next 8 bytes, unpack back to an int
    - For the decryption module IV = md5hash(bytes(msg_size^2))
    - Return the Square the msg_size in byte form
    """
    #Split byte string from uuid and rest of the msg
    split = msg.split(uuid_obj.bytes)
    #Get the first 8 bytes of the second chunk of msg
    msg_size_byte = split[1][0:8]
    ciphertext = split[1][8:]
    #Decode the byte string into an integer to square before converting back into
bytes
    msg_size = struct.unpack("Q", msg_size_byte)
    msg_size_square = msg_size[0]**2
    msg_size_square_bytes = struct.pack("Q", msg_size_square)
    return msg_size_square_bytes, ciphertext, msg_size_byte
def int_of_string(s):
    return int(binascii.hexlify(s), 16)
def decrypt_message(key, msg_size_square, ciphertext):
    """
    - AES CTR requires a key and counter, here counter and IV can be used
interchangeably
    - Key = Qtoken in its 32byte binary for
    - IV = md5hash(byte(msg_size_square)) which is 16 bytes
    - counter needs a ctr object
    - Counter.new(nbits, other params), nbits is the number of bits.
128bits = 16bytes
    """
    #Unhex qtoken to byte form, and return byte form of md5hash for IV
    key = key.decode('hex')
    iv = hashlib.md5(msg_size_square)
    iv_hash = iv.digest()
    #Initialize counter with the int_of_string of the iv, then decrypt
    ctr = Counter.new(128, initial_value=int_of_string(iv_hash))
    aes = AES.new(key, AES.MODE_CTR, counter=ctr)
    decrypted_msg = aes.decrypt(ciphertext)
    return iv, int_of_string(iv_hash),decrypted_msg
def encrypt_message(key, msg_size_square, ciphertext, uuid_obj,
msg_size_byte):
    """
    - AES CTR requires a key and and counter, here counter and IV can be
used interchangeably
    - Key = Qtoken in its 32byte binary for
    - IV = md5hash(byte(msg_size_square)) which is 16 bytes
    - counter needs a ctr object
    - Counter.new(nbits, other params), nbits is the number of bits.
128bits = 16bytes
    """
    #Unhex qtoken to byte form, and return byte form of md5hash for IV
    key = key.decode('hex')
    iv = hashlib.md5(msg_size_square)
    iv_hash = iv.digest()
    #Initialize counter with the int_of_string of the iv, then decrypt
    ctr = Counter.new(128, initial_value=int_of_string(iv_hash))
    aes = AES.new(key, AES.MODE_CTR, counter=ctr)
    encrypted_msg = aes.encrypt(ciphertext)
    #Add in the uuid and byte size headers to the front of the encrypted string
    encrypted_msg = uuid_obj.bytes + msg_size_byte + encrypted_msg
    return iv, int_of_string(iv_hash),encrypted_msg
def main():
    """
    - Get qortex to enable sercure mode
    - Connect to https authentication server with username and password
    - If authentication is successful, a QToken will be returned as a
header
    - For decryption, we need the key and the msg_size
    - The key is the QToken returned from https authentication, will be a
hex represenation of a string
    - The msg_size will be part of the data stream returned from TCP
    - Secured TCP data stream format: [separator_uuid: 16
bytes][orig.size: 8 bytes][encrypted data]
    - TCP data stream will be an encrypted byte string
    - The uuid will be 3debc6c9-08cc-7a02-9af0-1a082c39f4d2 in byte form
    - The msg size will be the next 8 bytes after the uuid
    - After obtaining both the Qtoken msg size we are ready for AES CTR
decryption
    """
    username = 'user1'
    password = 'Letmein433'
    port = 17171
    server_ip = '127.0.0.1'
    packet_size = sys.maxint / 10000000000
    uuid_str = '3debc6c9-08cc-7a02-9af0-1a082c39f4d2'
    uuid_obj = uuid.UUID('{' + uuid_str +'}')
    # while True:
    # Ping Qortex HTTPS server with credentials to receive Q-Token
    r_authentication, qtoken = https_authentication(server_ip, username,
password)
    r_secure, secure_flag =  get_secured(server_ip)
    #Get Data Stream from TCP
    msg = read_tcp(server_ip, port, packet_size)
    #Find msg size
    msg_size_square, ciphertext, msg_size_byte = get_msg_size(msg, uuid_obj)
    #AES CTR Decryption
    iv, int_of_string, decrypted_msg = decrypt_message(qtoken,
msg_size_square, ciphertext)
    # AES CTR Encryption
    iv, int_of_string, encrypted_msg = encrypt_message(qtoken,
msg_size_square, decrypted_msg, uuid_obj, msg_size_byte)
    #Test
    print msg
    print(msg, type(msg), len(msg))
    print("\n")
    # print("Secure status:", secure_flag)
    print("UUID bytes (Hex/Bytes):", uuid_str, uuid_obj.bytes)
    print("Msg Square Bytes:", msg_size_square)
    print("Qtoken (Hex/Bytes):", qtoken, qtoken.decode('hex'))
    print("IV (Hex/Bytes)", iv.hexdigest(), iv.digest())
    print("Int of string", int_of_string)
    print decrypted_msg
    # test = decrypted_msg[4:]
    # print test
    # print decrypted_msg[1]
    # json.loads(test)
    # print("Reencrypted msg", encrypted_msg, type(encrypted_msg),
len(encrypted_msg))
    # print("Org msg == Reencrypted Msg?:", msg == encrypted_msg)
if __name__ == "__main__":
    main()
```

Client Q-Auth HTTP Request Header Example

Set the request with custom `Q-Auth` header with QORTEX:

QORTEX example requests

```
/// Basic semantics for all requests
QNetworkAccessManager nam;
QNetworkReply* request(QString command) {
   QNetworkRequest req("http://127.0.0.1/v1/login");
   req.setRawHeader("Q-Auth", command);
   return nam.get(req);
 }
request("GetSecured");
   // Request if system is secured [does not need credentials]
request("SetSecured",
   // Request to set system secured [requires either user or license credentials]
   true);
```

```
        // {true/ false} where "true" is to set the secure mode and "false" to
unsecure
request("SetCredentials"
    // Request to set new user credentials [requires either user or license
credentials]
        we only have one user and we replace its credentials]
    "New_UserName_:New_Password_AES_128_CBC_Hash");
        // The encryption is described below
request("GetToken");
    // Request user security token [requires user credentials]
    // Will get reply as [256 bit key as hex string, will be used with AES 256 CTR
encryption]
```

Brief Server HTTP Reply Header Example

Set the reply with a few `key : value` pairs.

QORTEX example response handling

```
/// Basic spec for passing the server replies back to the client
void HttpClient::requestDone()
{
    QString errMsg = QStringLiteral("Unknown reason");
    if (reply_ && reply_->isFinished()) {
        QByteArray qbData = reply_->readAll();
        reply_->deleteLater();
        qDebug() << "HTTP reply body:" << qbData;
        QString token = reply_->rawHeader("Q-Token");
        QString secured = reply_->rawHeader("Q-Secured");
        QString setSecured = reply_->rawHeader("Q-SetSecured");
        QString credSet = reply_->rawHeader("Q-CredentialsSet");
        if (!token.isEmpty())
            emit securityTokenUpdate(uuid_token_ = token);
        if (!secured.isEmpty())
            emit securedState(is_secured_ = secured == "true");
        if (!setSecured.isEmpty())
            emit setSecuredSuccessful(setSecured == "ok");
        if (!credSet.isEmpty())
            emit credentialsSet(credSet == "true");
        auto httpResponseCode = reply_->attribute(
QNetworkRequest::HttpStatusCodeAttribute );
        if (!httpResponseCode.isValid() || httpResponseCode.toInt() != 200) {
            QString reason = reply_->attribute(
QNetworkRequest::HttpReasonPhraseAttribute ).toString();
            emit lastRequestFailed(reason);
        }
        return;
    }
    emit securityTokenUpdate(uuid_token_ = QString());
    qWarning() << "HTTP Digest failed due to" << errMsg;
}
```

HTTP Password Encryption Formula

The expectation is that encryption is primarily managed by the application library. QORTEX DTC handles, the HTTP digest MD5 combined hash and has only one explicit encryption in the client code for this part of functionality: Encrypting the new password for `Q-Auth:SetCredentials` request header.

For example, an unsecured `Q-Auth` request header value might be: `quser:433LakeSide`, and encrypted version would be: `quser:19eb639bb525204842e7e7daeb3fbc69`.

The hash formula is:

New password encryption formula

```
const auto& userNameHex = crypto::MD5Hash(user);
const auto& pwdHex = crypto::MD5Hash(pwd);
crypto::EncryptHelper<crypto::AES_128_CBC_Cipher> dh(userNameHex, pwdHex);
const auto& newPwdCrypted = dh.encode(newPwd.toStdString());
```

The encryption with MD5 is always 128-bit value and while not reversible to the source it is same for same source. AES CBC 128 bit accepts both `key` and `input vector` as 128-bit hashes while processing `newPwd`. This ensures the return is one possible decryption for the same value. This is really hard to break as there is no old password in plain text sent via HTTP. The QORTEX DTC server only confirms the old password by comparing the expression hash with the QORTEX DTC client HTTP digest result hash.

HTTP User Security Options

Use these as part of `SetCredentials`.

```
DeleteCredentials(string admin_hash, string username) -> {success/failed}
EnableUser(string admin_hash, string username, bool on) -> {success/failed}
EditGlobalSettings(admin_hash,

{JSON for multiple variables}
)
EditUserSettings(admin_hash, user_name,

{JSON for user role maybe more}
)
ListUsers(admin_hash)
```

QORTEX DTC TCP Listener with Security Enabled

The QORTEX DTC TCP listener uses the `getToken` API for decrypting secured messages. All other decryption handling is through the QORTEX DTC client. Qortex DTC uses AES 256 and AES 128 encryption.

- AES 256 CTR—for encrypting TCP data and decrypting per the TCP listener’s example.
- AES 128 CBC—for encrypting new passwords sent from the client.

TCP Listener Requirements

To use the Qortex DTC TCP listener, install the following:

- CMake 2.8.1 or later
- Boost 1.66 or later

Sample TCP Listener Actions

All the examples will use 0.0.0.0 as sample host IP as sample port number.

To start the listener for detected objects, run the following command for both Qortex-Server:

```
 $ ./qortex_listener -h 0.0.0.0 -p 17171 -t qobject
```

To start the listener for defined zone list, run the following command for both Qortex-Server:

```
 $ ./qortex_listener -h 0.0.0.0 -p 17172 -t qzone
```

To start the listener for current sensor status, run the following command for Qortex-Server:

```
 $ ./qortex_listener -h 0.0.0.0 -p 17178 -t qstate
```

To start the listener for point cloud, run the following command for Qortex-Server:

```
 $ ./qortex_listener -h 0.0.0.0 -p 17173 -t pointcloud
```

To start the listener for detected objects with new format, run the following command for Qortex-Server:

```
$ ./qortex_listener -h 0.0.0.0 -p 17161 -t qtrack
```

To start the listener for current sensor state list, run the following command for Qortex-Server:

```
$ ./qortex_listener -h 0.0.0.0 -p 17168 -t qsensorstate
```

To start the listener for detected objects with authentication using username (user) and password (pwd), run the following command for both Qortex-Server:

```
$ ./qortex_listener -h 0.0.0.0 -p 17171 -t qobject -u user -w pwd
```

Sample HTTP TCP Listener Encryption

```
/*****************************************************************
 **                                                             **
 **  Copyright(C) 2021 Quanergy Systems. All Rights Reserved.  **
 **  Contact: http://www.quanergy.com                           **
 **                                                             **
 *****************************************************************/
#include <string>
// The HttpClient class below provides a way to run HTTP requests
// against Qortex Server to obtain a security token
class HttpClient
{
public:
  /** Construction */
  HttpClient();
  /** Return the instance of this class */
  static HttpClient* instance();
 /**
  * @brief login with the following info
  * @param user user name
  * @param pwd password
  */
  void logIn(const std::string& user, const std::string& pwd);
 /**
  * @brief setServerIp sets HTTP Server IP which expected to be same as gRPC
  *      and TCP publisher's and we also assume the port will be 8080 or
  *      "HTTP Alternative" standard port which we control ourselves.
  * @param ip IP address
  * @param port Port number
  */
  void setServerIp(const std::string& ip, int port = 8080);
  bool isSecured() const { return is_secured_; }
 /**
  * @brief securityToken
  * @return the previously retrieved security token
  */
  const std::string& securityToken() const { return uuid_token_; }
protected:
 /**
  * @brief getToken initialize HTTP Digest Authentication for obtaining
  *      an authentication token
  * @param user user name
  * @param pwd password
  */
  void getToken(const std::string& user, const std::string& pwd);
 /**
  * @brief httpGetRequest call GET request with the given credentials
  * @param user user name
  * @param pwd password
  * @param auth given Q-Auth command
  */
  void httpGetRequest(const std::string& user, const std::string& pwd,
                      const std::string& auth);
  bool is_secured_{ false };
  std::string url_;
  std::string user_;
  std::string pwd_;
  std::string uuid_token_;
  static HttpClient* instance_;
private:
};
```

Sample TCP Listener API Python

```
/*************************************************************
 **                                                         **
 **  Copyright(C) 2017 Quanergy Systems. All Rights Reserved.  **
 **  Contact: http://www.quanergy.com                       **
 **                                                         **
 *************************************************************/
#ifndef QGUARD_TCP_LISTENER_H_
#define QGUARD_TCP_LISTENER_H_
#include <vector>
#include <boost/asio.hpp>
#include <crypto_helper.h>
class QortexTcpListener
{
  static constexpr std::size_t HEADER_LENGTH = sizeof(std::uint32_t);
  static constexpr std::size_t UUID_LENGTH = 16;
  static constexpr std::size_t RAW_DATA_BUFFER_LENGTH = 1024 * 1024;
  static constexpr std::size_t MAX_EXPECTED_DATA_LENGTH = 10 * 1024 * 1024;
public:
  enum MessageType
  {
    QOBJECT = 0,
    QZONE = 1,
    QSTATE = 2,
    QTRACKABLE = 3,
    POINTCLOUD = 4,
    QSENSORSTATE = 5
  };
  using ByteVector = std::vector<std::uint8_t>;
public:
 /**
   * @brief constructor
   * @param[in] boost::asio::io_service object as a reference
   *            boost::asio::ip::tcp::resolver
   *            message_type -- is enum of type QOBJECT, QZONE, QSTATE, QTRACKABLE,
POINTCLOUD
   */
  QortexTcpListener(boost::asio::io_service& io_service,
                    boost::asio::ip::tcp::resolver::iterator
endpoint_iterator,
                    MessageType message_type,
                    const std::string& user,
                    const std::string& pwd);
  /**
   * @brief close closes the tcp connection
   */
  void close();
private:
  /**
   * @brief connect connects to tcp endpoint specified by endpoint_iterator.
   *        calls the readHeader handler if connection was established
   * @param[in] endpoint_iterator tcp endpoint to connect to
   */
  void connect(boost::asio::ip::tcp::resolver::iterator endpoint_iterator);
  /**
   * @brief readHeader reads the first 4 (HEADER_LENGTH) bytes of the recieved
   *        message to obtain the message body length. Calls readBody handler
   *        if body length is non-zero.
   */
  void readHeader();
  /**
   * @brief readBody reads the 4th+(body_length_) bytes of the received message,
   *        tries to deserialize the message to a QObjectArray protobuf object
   *        and prints out the objects. Otherwise, prints out the received bytes
   *        as plain text (helpful in case of json/xml formats)
   */
  void readBody();
  /** @brief readData and do decryption if necessary */
  void tryReadEncrypted(const std::vector<uint8_t>& new_data);
  /** @brief do an async read operation */
  void doRead();
  /**
   * @brief byteArrayToInt converts a four-byte char array into an int32.
   * @param[in] source the four-byte char byte-array
   * @return a 32bit integer represented by the byte-array
   */
  std::size_t byteArrayToInt(std::array<char, HEADER_LENGTH> source );
  /** @brief find UUID to set isEncrypted flag */
  int findUuid(const std::vector<uint8_t>& data, bool& isEncrypted) const;
  /** @brief request authentication token */
  bool requestToken(std::string& token);
  /** @brief reset decryptor with the given token */
  void resetDecryptor(const std::string& token);
  void readHeader(const std::vector<uint8_t>& decoded);
  void readBody(const std::vector<uint8_t>& decoded);
  void parseProtobuf(const void* databuf);
private:
  boost::asio::io_service& io_service_;
  boost::asio::ip::tcp::socket socket_;
  std::size_t body_length_ = 0;
  std::vector<uint8_t> data_;
  ByteVector tcp_buffer_;
  ByteVector slack_;
  /**
   * @brief  message_type is an enum of type QOBJECT, QZONE, QSTATE, QTRACKABLE,
POINTCLOUD
   */
  MessageType message_type_;
  /**
   * @brief Incoming data stream order of data size is cached for referencing in
the class.
   */
  bool reversed_byte_order_ = {true};
  std::string user_;
  std::string password_;
  boost::asio::ip::tcp::tcp::resolver::iterator endpoint_iterator_;
  std::shared_ptr<
    quanergy::crypto::TcpDecryptHelper<quanergy::crypto::AES_256_CTR_Cipher,
ByteVector>
  > decryptor_;
  std::array<std::uint8_t, UUID_LENGTH> uuid_buf_sep_;
};
#endif // #QGUARD_TCP_LISTENER_H_
```

Sample TCP Listener API C++

HTTP Authentication and getToken

```
//------------------------------------------------------------------------------
// :: HTTP Authentication and get Token (uuid_token_) [see qortex_http_client.cpp]
//------------------------------------------------------------------------------

void HttpClient::logIn(const std::string& user, const std::string& pwd)
{
  getToken(user, pwd);
}
//------------------------------------------------------------------------------

void HttpClient::getToken(const std::string& user, const std::string& pwd)
{
  uuid_token_.clear();
  if (user.empty() || pwd.empty() || url_.empty()) {
    std::cerr << "Cannot authenticate due to empty username/password/URL\n";
    return;
  }
  // Auth is GetToken
  httpGetRequest(user, pwd, "GetToken");
}
//------------------------------------------------------------------------------
// Send GET request with Auth "GetToken"
void HttpClient::httpGetRequest(const std::string &user, const std::string &pwd,
                                const std::string &auth)
{
  if (user.empty() || pwd.empty() || url_.empty()) {
    std::cerr << "Cannot authenticate due to empty username/password/URL\n";
    return;
  }
  static const int timeout_msec = 2000;
  user_ = user;
  pwd_ = pwd;
  int result = -1;
  std::string reason;
  boost::asio::io_context io_context;
  try
  {
    // Create client.
    auto client = std::make_shared<simple_http::get_client>(
      io_context,
      [&result, &reason, this](simple_http::empty_body_request& req,
                               simple_http::string_body_response& resp)
      {
        result = resp.result_int();
        if (resp.result_int() != 200)
        {
          reason = std::string(resp.reason());
          std::cerr << "Failed response " << resp << "\n";
        } else {
          std::cout << "Success response " << resp << "\n";
          uuid_token_ = std::string(resp.base()["Q-Token"]);
          is_secured_ = true;
        }
      });
    client->setAuthorization(user_, pwd_);
    // Optionally set a fail action
    client->setFailHandler(
      [&result, &reason](const simple_http::empty_body_request& req,
                         const simple_http::string_body_response& resp,
                         simple_http::fail_reason fr,
                         boost::string_view message)
      {
        result = 0;
        reason = std::string(message);
      });
    client->get(simple_http::url(url_));
    // Run this HTTP conversation up to the time limit specified
    io_context.run_for(std::chrono::milliseconds(timeout_msec));
    if (result == 200)
    {
      std::cout << "Request completed, response: " << result << "\n";
    }
    else // Other response code
    {
      // Although there are other successful responses that are not 200,
      // this is likely a failure, log as such.
      std::cerr << "Request failed, response: " << result << "\n";
      if (result == -1)
      {
        std::cerr << "Can't complete request: Connection timed out.\n";
      }
      // Log other error message [as TCP or another protocol failed]
      else if (result == 0)
      {
        std::cerr << "Can't complete request: " << reason << "\n";
      }
    }
  }
```

```
  catch(const std::exception& e)
  {
    std::cerr << "Can't complete authentication request: " << e.what() << '\n';
  }
}
```

TCP Listener requestToken

```
//------------------------------------------------------------------------------
// :: TCP Listener request Token and decrypt message [see qortex_tcp_listener.cpp]
//------------------------------------------------------------------------------
bool QortexTcpListener::requestToken(std::string& token)
{
  std::cout << "Getting Token ..." << std::endl;
  auto* httpClient = HttpClient::instance();
  const auto& ip = endpoint_iterator_->endpoint().address().to_string();
  httpClient->setServerIp(ip);
  httpClient->logIn(user_, password_);
  token = httpClient->securityToken();
  return !token.empty();
}
//------------------------------------------------------------------------------
// Create decryptor_ which is a crypto class that can decrypt message
void QortexTcpListener::resetDecryptor(const std::string& tokenHexStr)
{
  if (!tokenHexStr.empty())
  {
    std::array<std::uint8_t, 32> key_arr;
    std::vector<std::uint8_t> token =
quanergy::util::unhex<std::uint8_t>(tokenHexStr);
    std::copy(token.begin(), token.end(), key_arr.data());
    // Setup decryptor
    decryptor_ = std::make_shared<
      quanergy::crypto::TcpDecryptHelper<quanergy::crypto::AES_256_CTR_Cipher,
ByteVector>>
      ( key_arr, uuid_buf_sep_ );
    std::cout << "decryptor created" << std::endl;
  }
  else
  {
    decryptor_.reset();
  }
}
//------------------------------------------------------------------------------
// Reading encrypted data and call the handle: tryReadEncrypted(encrypted_data)
void QortexTcpListener::doRead()
```

```
{
  socket_.async_read_some(
    boost::asio::buffer(tcp_buffer_),
    [this](boost::system::error_code ec, std::size_t bytes_transferred) {
      if (!ec)
      {
        tcp_buffer_.resize(bytes_transferred);
        // Signal that data has arrived
        ByteVector data = std::move(tcp_buffer_);
        tcp_buffer_.resize(RAW_DATA_BUFFER_LENGTH);
        tryReadEncrypted(data);
        // Read more data
        doRead();
      }
      else if (ec != boost::asio::error::operation_aborted)
      {
        socket_.close();
      }
    });
}
//-------------------------------------------------------------------------------
// Find UUID to see if buffer is encrypted
int QortexTcpListener::findUuid(const std::vector<uint8_t>& data, bool&
isEncrypted) const
{
  auto it = std::search(data.cbegin(), data.cend(),
                        uuid_buf_sep_.begin(), uuid_buf_sep_.end());
  if (it != data.cend()) {
    isEncrypted = true;
    // std::cout << "The uuid " << uuid_buf_sep_str_ << " found at offset "
    //           << it - slack_.begin() << std::endl;
    return it - data.cbegin();
  }
  isEncrypted = false;
  return -1;
}
//-------------------------------------------------------------------------------
void QortexTcpListener::tryReadEncrypted(const std::vector<uint8_t>& new_data)
{
  try {
    bool encrypted = false;
    slack_.insert(slack_.end(), new_data.begin(), new_data.end());
    auto start_of_uuid = findUuid(slack_, encrypted);
    if (encrypted && slack_.size() > start_of_uuid + UUID_LENGTH + sizeof(size_t))
{
      ByteVector decoded_data;
      const std::uint8_t* encr_ptr = slack_.data() + start_of_uuid + UUID_LENGTH;
      const size_t& count = reinterpret_cast<const size_t&>(*(encr_ptr));
   // Check if the decryption possible to avoid the crash
      if (count > 0 && count <= slack_.size() - UUID_LENGTH) {
    // here we pass the address of [size][encrypted_bytes] portion of data
        decoded_data = decryptor_->decode_with_md5_counter_iv(encr_ptr);
        if (!decoded_data.empty()) {
     // Cut the processed data out. The original size is maybe a bit less than
     // encrypted but the algorithm rely on next UUID to be found anyway
          int old_size = start_of_uuid + UUID_LENGTH + sizeof(size_t) +
decoded_data.size();
          int new_size = slack_.size() - old_size;
          ByteVector clone = std::move(slack_);
          if (new_size > 0) {
            slack_.resize(new_size);
            std::copy(clone.cbegin() + old_size, clone.cend(), slack_.begin());
          }
     // Dispatch for further processing
          readHeader(decoded_data);
        }
      }
    }
    else {
      readHeader(slack_);
      slack_.clear();
    }
  }
  catch (std::exception& exc) {
    std::cerr << "QortexTcpListener::readData: " << exc.what() << std::endl;
    slack_.clear();
  }
  catch (...) {
    std::cerr << "QortexTcpListener::readData (unknown exception)" << std::endl;
    slack_.clear();
  }
  if (slack_.size() > MAX_EXPECTED_DATA_LENGTH) {
    std::cerr << "Could not find encryption separator for too long" << std::endl;
    slack_.clear();
  }
}
//******************************************************************************
```

Appendix 4. Template Parameter Settings

QORTEX DTC 2.3 offers three templates for optimizing the configuration of the software to track person, vehicle, or general (mixture). See *Add or Update Multiple Sensors at a Location* (page 89). Beyond these default optimizations, the settings can be fine-tuned for particular needs by changing the parameter values through the gRPC API (below) or through the client user interface. *Table 21. Sensor Settings Parameters for StaticCloudPipeline*, *Table 22. Sensor Settings Parameters for ClusterPipeline*, *Table 23. Sensor Settings Parameters for TrackerPipeline*, *Table 24. Sensor Settings Parameters for OutputFilter*, and *Table 25. Sensor Settings Parameters for MultiLidarPipeline* list the parameter values.

Changes via gRPC API

One way to change the parameter values is by editing the xml file through Configuration gRPC API commands from the terminal. The following is an example of using the gRPC API when updating parameter settings values.

1. Specify the `<ClusterPipeline><CCClusterer><cellSize>` parameter settings value and the data format (`json` or `xml`). For example:

   For xml:

```
    stream = "<Settings>
    <ClusterPipeline>
    <CCClusterer>
    <cellSize>0.6</cellSize>
    </CCClusterer>
    </ClusterPipeline>
    </Settings>"
```

   For json:

```
stream = "{"Settings": {"ClusterPipeline": {"CCClusterer": {"cellSize": 0.6}}}}"
```

2. Use proto format for the settings API

```
// The new Settings Getter/Setter API
enum SettingsCommand {
 SettingsCommand_UNSPECIFIED = 0;
 SET_COMMAND = 1;
 GET_COMMAND = 2;
}
enum SettingsFormat {
 SettingsFormat_UNSPECIFIED = 0;
 XML_FORMAT = 1;
 JSON_FORMAT = 2;
```

```
}
message SettingsRequest {
 SettingsCommand command = 1;
 bytes stream   = 2;
 SettingsFormat format = 3;
}

Note (for message SettingRequest):
command = SET_COMMAND or GET_COMMAND
stream = json or xml data
format = XML_FORMAT or JSON_FORMAT (use appropriate format because wrong format
matching, such as stream = xml and format = json, may not set or get data.
```

Changes via Client User Interface

Another way to change the parameter values is by editing fields in the **Sensor** tab. See *Figure 112. Settings: Sensor Tab*.

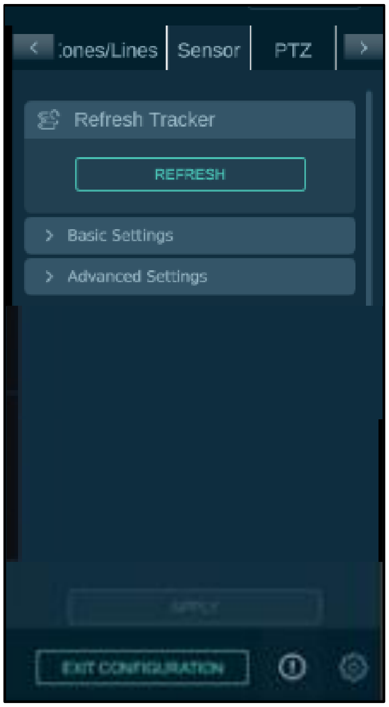


Figure 112. Settings: Sensor Tab

An example portion of the parameter settings is shown with default values in the client user interface and the xml file. See *Figure 113. Settings: User Interface (top) and Template File (bottom) examples*.

qtrack_cluster_algorithm_settings_general.xml

```
<ClusterPipeline>
   <CCClusterer>
```

```
<!-- Size of cells (cells are square). -->
      <cellSize>
         <value>0.40<value>
         <min>0.1</min>
         <max>1.0</max>
         <type>double</type>
      </cellSize>
   </CCClusterer>
   <ClusterFilter>
<!-- Clusters with fewer than this many points are discarded. -->
      <minNumClusterPoints>
         <value>10<value>
         <min>1</min>
         <max>20</max>
         <type>int</type>
      </minNumClusterPoints>
<!-- Clusters whose width and length are smaller than this are discarded. -->
      <minSize>
         <value>0.2<value>
         <min>0.1</min>
         <max>2.0</max>
         <type>double</type>
      </minSize>
<!-- Clusters are discarded if either its width or length exceeds this. -->
      <maxSize>
         <value>100.0<value>
         <min>1.0</min>
         <max>100.0</max>
         <type>double</type>
      </maxSize>
</ClusterFilter>
<ClusterSplitter>
<!-- Switch to enable/disable cluster splitter. -->
<enable>
```

Figure 113. Settings: User Interface (top) and Template File (bottom) examples

Parameter Settings Values

The following tables list the parameters available for a variety of pipelines and filters, and a description of each. Each table includes recommended settings for each parameter with regard to General, Person, or Vehicle sensing (the three columns on the far left of each table), as well as the minimum and maximum value the parameter accepts. The parameter types include double (accepts decimal values) or int (accepts only non-decimal values). Unlabeled parameter types accept only true or false values.

StaticCloudPipeline parameters

`StaticCloudPipeline` filters background and Exclusion zone points and then passes filtered foreground points to next pipeline. *Table 21. Sensor Settings Parameters for StaticCloudPipeline* describes each `BackgroundFilter` parameter and its recommended settings.

Table 21. Sensor Settings Parameters for StaticCloudPipeline

Section, Parameter	**Description**	**General**	**Person**	**Vehicle**

`StaticCloudPipeline`
`BackgroundFilter`
`maxBinDistance`
	Maximum distance for voxelizing in a radial direction in meters.	value: 100
		min: 50
		max: 200
		type: int
			value: 50
			min: 50
			max: 200
			type: int
				value: 10
				min: 50
				max: 200
				type: int

`StaticCloudPipeline`
`BackgroundFilter`
`distanceBinLength`
	The bin size in the radial direction.	value: 1.0
		min: 0.1
		max: 2.0
		type: double
			value: 0.2
			min: 0.1
			max: 2.0
			type: double
				value: 0.5
				min: 0.1
				max: 2.0
				type: double

`StaticCloudPipeline`
`BackgroundFilter`
`timeUntilBackground`
	Maximum amount of time (in seconds) for an occupied location to go from considered as foreground to considered as background. If an object enters an unoccupied location and stays there, it fades into the background in this amount of time (less time if location was recently occupied).	value: 150
		min: 10
		max: 10000
		type: int
			value: 150
			min: 10
			max: 10000
			type: int
				value: 500
				min: 10
				max: 10000
				type: int

`StaticCloudPipeline`
`BackgroundFilter`
`timeUntilForeground`
	Maximum amount of time (in seconds) a location remains unoccupied for an object entering it to be considered foreground. If an object remains in an unoccupied space for this amount of time, the object is guaranteed to be considered foreground (may require less time if previous occupations were short).	value: 300
		min: 10
		max: 10000
		type: int
			value: 300
			min: 10
			max: 10000
			type: int
				value: 8000
				min: 10
				max: 10000
				type: int

`StaticCloudPipeline`
`BackgroundFilter`
`ForegroundLock`
	Makes a location a permanent free space location once observed as free space for a certain consecutive amount of time	value: false
		option: true
		option: false
			value: false
			option: true
			option: false
				value: false
				option: true
				option: false

Section, Parameter	Description	General	Person	Vehicle
StaticCloudPipeline BackgroundFilter timeUntilAlwaysForeground	Time (in seconds) a location must be observed as free space to be considered free space permanently. An object moving into a free space location is considered to be in the foreground.	value: 10 min: 5 max: 50 type: int	value: 10 min: 5 max: 50 type: int	value: 15 min: 5 max: 50 type: int
StaticCloudPipeline BackgroundFilter useNanAsUnobstructed	If true, NaN points are ignored and not used. If false, NaN indicates free space (to infinity) at that point (e.g., if NaN meets maximum range and nothing was detected). Use Exclusion Zones to adjust for false foregrounds, if needed.	value: false option: true option: false	value: false option: true option: false	value: true option: true option: false
StaticCloudPipeline BackgroundFilter updateEveryNthFrame	Sets frequency at which the background filter updates. Value of 1: updates occur every frame. Value of 2: updates occur every 2nd frame, etc. Changing this value affects times in other settings. For example, if value is 10, time moves ten times slower in the background filter, since it counts frames to determine time.	value: 5 min: 1 max: 10 type: int	value: 5 min: 1 max: 10 type: int	value: 5 min: 1 max: 10 type: int

ClusterPipeline parameters

ClusterPipeline groups filtered for group points into clusters, using a different algorithm. *Table 22. Sensor Settings Parameters for ClusterPipeline* describes each CCClusterer, ClusterFilter, and ClusterSplitter parameter and its recommended settings.

Table 22. Sensor Settings Parameters for ClusterPipeline

Section, Parameter	Description	General	Person	Vehicle
ClusterPipeline CCClusterer cellSize	Size of cells. These cells are square.	value: 0.40 min: 0.1 max:1.0 type: double	value: 0.15 min: 0.1 max:1.0 type: double	value: 0.5 min: 0.1 max:1.0 type: double
ClusterPipeline ClusterFilter minNumClusterPoints	Clusters with fewer than this number of points are discarded.	value: 10 min: 1 max: 20 type: int	value: 10 min: 1 max: 20 type: int	value: 8 min: 1 max: 20 type: int

`ClusterPipeline`
`ClusterFilter`
`minSize`
	Clusters whose area (width * length) is smaller than this are discarded.	value: 0.2
		min: 0.1
		max: 2.0
		type: double
			value: 0.2
			min: 0.1
			max: 2.0
			type: double
				value: 0.3
				min: 0.1
				max: 2.0
				type: double

`ClusterPipeline`
`ClusterFilter`
`maxSize`
	Clusters whose area (width * length) is larger than this are discarded.	value: 100.0
		min: 1.0
		max: 100.0
		type: double
			value: 5.0
			min: 1.0
			max: 100.0
			type: double
				value: 100.0
				min: 1.0
				max: 100.0
				type: double

`ClusterPipeline`
`ClusterSplitter`
`enable`
	Enables or disables cluster splitter.	value: false
		option: true
		option: false
			value: false
			option: true
			option: false
				value: false
				option: true
				option: false

`ClusterPipeline`
`ClusterSplitter`
`onlySplitPersonClass`
	Only splits person classes. Setting this to true disables all current splitting, as cluster class labels are unknown.	value: false
		option: true
		option: false
			value: false
			option: true
			option: false
				value: true
				option: true
				option: false

`ClusterPipeline`
`ClusterSplitter`
`onlySplitMultiLidar`
	Only splits multi-lidar clusters. Setting this to true disables splitting of single-lidar clusters.	value: true
		option: true
		option: false
			value: true
			option: true
			option: false
				value: true
				option: true
				option: false

`ClusterPipeline`
`ClusterSplitter`
`minStandardDeviation`
	Minimum standard deviation threshold above which the object is split. This value is the standard deviation of cluster points
	(in meters).
		value: 0.15
		min: 0.01
		max: 5.0
		type: double
			value: 0.15
			min: 0.01
			max: 5.0
			type: double
				value: 0.15
				min: 0.01
				max: 5.0
				type: double

`ClusterPipeline`
`ClusterSplitter`
`threshStandardDeviation`
	Threshold standard deviation threshold each split object should satisfy. This value is the standard deviation of cluster points
	(in meters).
		value: 0.45
		min: 0.01
		max: 5.0
		type: double
			value: 0.45
			min: 0.01
			max: 5.0
			type: double
				value: 0.45
				min: 0.01
				max: 5.0
				type: double

`ClusterPipeline`
`ClusterSplitter`
`maxStandardDeviation`
	Maximum standard deviation threshold below which the object is split. This value is the standard deviation of cluster points
	(in meters).
		value: 1.00
		min: 0.01
		max: 5.0
		type: double
			value: 1.00
			min: 0.01
			max: 5.0
			type: double
				value: 1.00
				min: 0.01
				max: 5.0
				type: double

`ClusterPipeline`
`ClusterSplitter`
`minNumPoints`
	Minimum number of points an object contains required for split.	value: 5
		min: 1
		max: 100
		type: int
			value: 5
			min: 1
			max: 100
			type: int
				value: 5
				min: 1
				max: 100
				type: int

TrackerPipeline parameters

`TrackerPipeline` associates clusters with existing trackables. It then updates existing trackables or generates new trackables. *Table 23. Sensor Settings Parameters for TrackerPipeline* describes each `DataAssociation`, `Tracker`, `TrackableFilter`, `TrackableDimensionsUpdater`, `TrackableProximityFinder`, and `TrackableFilter` parameter and its recommended settings.

Table 23. Sensor Settings Parameters for TrackerPipeline

Section, Parameter	Description	General	Person	Vehicle
`TrackerPipeline` `frequency`	Frequency (in Hz) at which the tracker pipeline outputs trackables.	value: 10 min: 1 max:20 type: int	value: 10 min: 1 max:20 type: int	value: 10 min: 1 max:20 type: int

`TrackerPipeline`
`DataAssociation`
`enableKdTreeLookup`
	Enables k-d tree lookup to speed data association when there are high numbers of clusters and trackables.	value: true
		option: true
		option: false
			value: true
			option: true
			option: false
				value: true
				option: true
				option: false

`TrackerPipeline`
`DataAssociation`
`maxAssociationRadius`
	Clusters and trackables must be closer than this Euclidean distance (as measured from their centroid positions, in meters) to be associated.	value: 4.00
		min: 0.0
		max: 10.0
		type: double
			value: 1.00
			min: 0.0
			max: 10.0
			type: double
				value: 5.00
				min: 0.0
				max: 10.0
				type: double

`TrackerPipeline`
`DataAssociation`
`maxAssociationDistance`
	Clusters and trackables must be closer than this distance to be associated. Note that overlapping clusters and trackables have a negative distance.	value: 1.5
		min: 0.0
		max: 5.0
		type: double
			value: 0.3
			min: 0.0
			max: 5.0
			type: double
				value: 2.0
				min: 0.0
				max: 5.0
				type: double

`TrackerPipeline`
`DataAssociation`
`maxMergeClusterAssociationDistance`
	Clusters and trackables must be closer than this distance to have an additional cluster-trackable association after the first one. Only applies to complex data association. Overlapping clusters and trackables have a negative distance.	value: 1.0
		min: 0.0
		max: 5.0
		type: double
			value: 0.0
			min: 0.0
			max: 5.0
			type: double
				value: 2.0
				min: 0.0
				max: 5.0
				type: double

Section, Parameter	Description	General	Person	Vehicle

`TrackerPipeline`
`DataAssociation`
`numWorkerThreads`
	The number of worker threads on which to run classification. In environments with many objects, this value can be increased to improve the rate at which objects get classified if the server has enough available cores.	value: 2
		min: 1
		max: 10
		type: int
			value: 2
			min: 1
			max: 10
			type: int
				value: 2
				min: 1
				max: 10
				type: int

`TrackerPipeline`
`DataAssociation`
`defaultMeasurementVarianceHeading`
	Default heading variance for measurements.	value: 0.07
		min: 0.007
		max: 0.07
		type: double
			value: 0.07
			min: 0.007
			max: 0.07
			type: double
				value: 0.07
				min: 0.007
				max: 0.07
				type: double

`TrackerPipeline`
`Tracker`
`numWorkerThreads`
	The number of worker threads on which to run classification. In environments with many objects, this number can be increased to improve the rate at which objects get classified if the server has enough available cores.	value: 2
		min: 1
		max: 10
		type: int
			value: 2
			min: 1
			max: 10
			type: int
				value: 2
				min: 1
				max: 10
				type: int

`TrackerPipeline`
`Tracker`
`initVariancePosX`
	Initial position variance of a new trackable in the X-direction.	value: 0.1
		min: 0.1
		max: 0.2
		type: double
			value: 0.1
			min: 0.1
			max: 0.2
			type: double
				value: 0.2
				min: 0.1
				max: 0.2
				type: double

`TrackerPipeline`
`Tracker`
`initVariancePosY`
	Initial position variance of a new trackable in the Y-direction.	value: 0.1
		min: 0.1
		max: 0.2
		type: double
			value: 0.1
			min: 0.1
			max: 0.2
			type: double
				value: 0.2
				min: 0.1
				max: 0.2
				type: double

`TrackerPipeline`
`Tracker`
`initVariancePosZ`
	Initial position variance of a new trackable in the Z-direction.	value: 0.1
		min: 0.1
		max: 0.2
		type: double
			value: 0.1
			min: 0.1
			max: 0.2
				value: 0.2
				min: 0.1
				max: 0.2

Section, Parameter	Description	General	Person	Vehicle
			type: double	type: double

`TrackerPipeline`
`Tracker`
`initVarianceVelX`
	Initial velocity variance of a new trackable in the X-direction.	value: 10
		min: 10
		max: 400
		type: int
			value: 10
			min: 10
			max: 400
			type: int
				value: 400
				min: 10
				max: 400
				type: int

`TrackerPipeline`
`Tracker`
`initVarianceVelY`
	Initial velocity variance of a new trackable in the Y-direction.	value: 10
		min: 10
		max: 400
		type: int
			value: 10
			min: 10
			max: 400
			type: int
				value: 400
				min: 10
				max: 400
				type: int

`TrackerPipeline`
`Tracker`
`initVarianceVelZ`
	Initial velocity variance of a new trackable in the Z-direction.	value: 10
		min: 10
		max: 10
		type: int
			value: 10
			min: 10
			max: 10
			type: int
				value: 10
				min: 10
				max: 400
				type: int

`TrackerPipeline`
`Tracker`
`initVarianceAccX`
	Initial acceleration variance of a new trackable in the X-direction.	value: 1
		min: 1
		max: 1
		type: int
			value: 1
			min: 1
			max: 1
			type: int
				value: 1
				min: 1
				max: 1
				type: int

`TrackerPipeline`
`Tracker`
`initVarianceAccY`
	Initial acceleration variance of a new trackable in the Y-direction.	value: 1
		min: 1
		max: 1
		type: int
			value: 1
			min: 1
			max: 1
			type: int
				value: 1
				min: 1
				max: 1
				type: int

`TrackerPipeline`
`Tracker`
`initVarianceAccZ`
	Initial acceleration variance of a new trackable in the Z-direction.	value: 1
		min: 1
		max: 1
		type: int
			value: 1
			min: 1
			max: 1
			type: int
				value: 1
				min: 1
				max: 1
				type: int

`TrackerPipeline`
`Tracker`
`initVarianceHeading`
	Initial heading variance of a new trackable.	value: 2.46
		min: 2.46
		max: 2.46
		type: double
			value: 2.46
			min: 2.46
			max: 2.46
			type: double
				value: 2.46
				min: 2.46
				max: 2.46
				type: double

`TrackerPipeline`
`Tracker`
`initVarianceHeadingRate`
	Initial heading variance rate of a new trackable.	value: 0.01
		min: 0.01
			value: 0.01	value: 0.01

Section, Parameter	Description	General	Person	Vehicle
		max: 0.01 type: double	min: 0.01 max: 0.01 type: double	min: 0.01 max: 0.01 type: double

`TrackerPipeline`
`Tracker`
`processNoiseVariancePosX`
	Process noise variance of the position in the X-direction.	value: 0.01
		min: 0.01
		max: 0.04
		type: double
			value: 0.01
			min: 0.01
			max: 0.04
			type: double
				value: 0.04
				min: 0.01
				max: 0.04
				type: double

`TrackerPipeline`
`Tracker`
`processNoiseVariancePosY`
	Process noise variance of the position in the Y-direction.	value: 0.01
		min: 0.01
		max: 0.04
		type: double
			value: 0.01
			min: 0.01
			max: 0.04
			type: double
				value: 0.04
				min: 0.01
				max: 0.04
				type: double

`TrackerPipeline`
`Tracker`
`processNoiseVariancePosZ`
	Process noise variance of the position in the Z-direction.	value: 0.01
		min: 0.01
		max: 0.04
		type: double
			value: 0.01
			min: 0.01
			max: 0.04
			type: double
				value: 0.01
				min: 0.01
				max: 0.04
				type: double

`TrackerPipeline`
`Tracker`
`processNoiseVarianceVelX`
	Process noise variance of the velocity in the X-direction.	value: 0.05
		min: 0.05
		max: 0.1
		type: double
			value: 0.05
			min: 0.05
			max: 0.1
			type: double
				value: 0.1
				min: 0.05
				max: 0.1
				type: double

`TrackerPipeline`
`Tracker`
`processNoiseVarianceVelY`
	Process noise variance of the velocity in the Y-direction.	value: 0.05
		min: 0.05
		max: 0.1
		type: double
			value: 0.05
			min: 0.05
			max: 0.1
				value: 0.1
				min: 0.05
				max: 0.1

Section, Parameter	Description	General	Person	Vehicle
			type: double	type: double
TrackerPipeline Tracker processNoiseVarianceVelZ	Process noise variance of the velocity in the Z-direction.	value: 0.1 min: 0.1 max: 0.1 type: double	value: 0.1 min: 0.1 max: 0.1 type: double	value: 0.1 min: 0.1 max: 0.1 type: double
TrackerPipeline Tracker processNoiseVarianceAccX	Process noise variance of the acceleration in the X-direction.	value: 1e–6 min: 1e–6 max:1e–6 type: double	value: 1e–6 min: 1e–6 max:1e–6 type: double	value: 1e–6 min: 1e–6 max:1e–6 type: double
TrackerPipeline Tracker processNoiseVarianceAccY	Process noise variance of the acceleration in the Y-direction.	value: 1e–6 min: 1e–6 max:1e–6 type: double	value: 1e–6 min: 1e–6 max:1e–6 type: double	value: 1e–6 min: 1e–6 max:1e–6 type: double
TrackerPipeline Tracker processNoiseVarianceAccZ	Process noise variance of the acceleration in the Z-direction.	value: 1e–6 min: 1e–6 max:1e–6 type: double	value: 1e–6 min: 1e–6 max:1e–6 type: double	value: 1e–6 min: 1e–6 max:1e–6 type: double
TrackerPipeline Tracker processNoiseVarianceHeading	Process noise variance of the heading.	value: 1e–3 min: 1e–3 max:1e–3 type: double	value: 1e–3 min: 1e–3 max:1e–3 type: double	value: 1e–3 min: 1e–3 max:1e–3 type: double

Section, Parameter	Description	General	Person	Vehicle
`TrackerPipeline TrackableFilter minSize`	Trackables whose largest dimension is smaller than this are filtered out.	value: 0.2 min: 0.1 max: 2.0 type: double	value: 0.2 min: 0.1 max: 2.0 type: double	value: 0.3 min: 0.1 max: 2.0 type: double
`TrackerPipeline TrackableFilter maxSize`	Trackables whose largest dimension is larger than this are filtered out.	value: 100.0 min: 1.0 max: 100.0 type: double	value: 5.0 min: 1.0 max: 100.0 type: double	value: 100.0 min: 1.0 max: 100.0 type: double
`TrackerPipeline TrackableFilter maxArea`	Trackables whose area (length * width) is larger than this are filtered out.	value: 100.0 min: 1.0 max: 1000.0 type: double	value: 25.0 min: 1.0 max: 1000.0 type: double	value: 100.0 min: 1.0 max: 1000.0 type: double
`TrackerPipeline TrackableFilter maxSpeed`	Trackables whose speed (m/s) is higher than this are filtered out.	value: 50.0 min: 2.0 max: 100.0 type: double	value: 20.0 min: 2.0 max: 100.0 type: double	value: 50.0 min: 2.0 max: 100.0 type: double
`TrackerPipeline TrackableFilter maxPositionStdDev`	Trackables whose position standard deviation along any axis is larger than this are filtered out.	value: 1.0 min: 1.0 max: 5.0 type: double	value: 1.0 min: 1.0 max: 5.0 type: double	value: 2.0 min: 1.0 max: 5.0 type: double
`TrackerPipeline TrackableDimensions Updater sensitivity`	When receiving a new dimension measurement of a trackable, adjusts the old dimensions by this fraction of the measurement only.	value: 0.1 min: 0.1 max: 0.5 type: double	value: 0.1 min: 0.1 max: 0.5 type: double	value: 0.3 min: 0.1 max: 0.5 type: double

Section, Parameter	Description	General	Person	Vehicle

`TrackerPipeline`
`TrackableProximityFinder`
`enable`
	Enables or disables proximity finder. This module helps when working with vehicles but is not beneficial when tracking people. Additionally, it has a high computational cost. As such, some users can disable it for people tracking.	value: true
		option: true
		option: false
			value: false
			option: true
			option: false
				value: true
				option: true
				option: false

`TrackerPipeline`
`TrackableProximityFinder`
`maxDistance`
	The maximum distance between two trackables for them to be considered in proximity of each other (i.e., to be considered for merging).	value: 0.5
		min: 0.5
		max: 5.0
		type: double
			value: 0.5
			min: 0.5
			max: 5.0
			type: double
				value: 2.0
				min: 0.5
				max: 5.0
				type: double

`TrackerPipeline`
`TrackableProximityFinder`
`frameInterval`
	Only process every nth frame of data, where n = frameInterval.	value: 10
		min: 1
		max: 10
		type: int
			value: 10
			min: 1
			max: 10
			type: int
				value: 3
				min: 1
				max: 10
				type: int

`TrackerPipeline`
`TrackableSplitter`
`enable`
	Enables or disables trackable splitter.	value: false
		option: true
		option: false
			value: false
			option: true
			option: false
				value: false
				option: true
				option: false

`TrackerPipeline`
`TrackableSplitter`
`onlySplitPersonClass`
	Splits person classes only. Setting this to true for TrackableSplitter splits the current pedestrian trackable only.	value: true
		option: true
		option: false
			value: false
			option: true
			option: false
				value: true
				option: true
				option: false

`TrackerPipeline`
`TrackableSplitter`
`onlySplitMultiLidar`
	Splits multi-lidar trackables only. Setting this to true for TrackableSplitter disables splitting of single-lidar trackables.	value: true
		option: true
		option: false
			value: true
			option: true
			option: false
				value: true
				option: true
				option: false

`TrackerPipeline`
`TrackableSplitter`
`minStandardDeviation`
	Minimum standard deviation threshold above which an object is split. This is the standard deviation of cluster points (in meters).	value: 0.15
		min: 0.01
		max: 5.0
			value: 0.45
			min: 0.01
				value: 0.15
				min: 0.01

Section, Parameter	Description	General	Person	Vehicle
		type: double	max: 5.0 type: double	max: 5.0 type: double

`TrackerPipeline`
`TrackableSplitter`
`threshStandardDeviation`
	Threshold standard deviation threshold each split object should satisfy. This is the standard deviation of cluster points (in meters).	value: 0.45
		min: 0.01
		max: 5.0
		type: double
			value: 0.45
			min: 0.01
			max: 5.0
			type: double
				value: 0.45
				min: 0.01
				max: 5.0
				type: double

`TrackerPipeline`
`TrackableSplitter`
`maxStandardDeviation`
	Maximum standard deviation threshold below which an object is split. This is the standard deviation of cluster points (in meters).	value: 1.00
		min: 0.01
		max: 5.0
		type: double
			value: 1.00
			min: 0.01
			max: 5.0
			type: double
				value: 1.00
				min: 0.01
				max: 5.0
				type: double

`TrackerPipeline`
`TrackableSplitter`
`minNumPoints`
	Minimum number of points in an object required for split.	value: 5
		min: 1
		max: 100
		type: int
			value: 5
			min: 1
			max: 100
			type: int
				value: 5
				min: 1
				max: 100
				type: int

OutputFilter parameters

`OutputFilter` filters trackables according to a defined rule. *Table 24. Sensor Settings Parameters for OutputFilter* describes each `OutputFilter` parameter and its recommended settings.

Table 24. Sensor Settings Parameters for OutputFilter

Section, Parameter	Description	General	Person	Vehicle
`OutputFilter` `mustBeClassified`	Trackables classified as “unknown” are filtered out.	value: false option: true option: false	value: false option: true option: false	value: false option: true option: false
`OutputFilter` `minMeasCount`	Trackables not measured at least this many times are filtered out.	value: 2 min: 1 max: 5 type: int	value: 2 min: 1 max: 5 type: int	value: 2 min: 1 max: 5 type: int

Section, Parameter	Description	General	Person	Vehicle
`OutputFilter` `minClusterPoints`	Trackables whose latest measurement did not contain at least this number of points are filtered out.	value: 2 min: 1 max: 20 type: int	value: 2 min: 1 max: 20 type: int	value: 2 min: 1 max: 20 type: int
`OutputFilter` `minSize`	Trackables whose largest dimension is smaller than this are filtered out.	value: 0.2 min: 0.1 max: 2.0 type: double	value: 0.2 min: 0.1 max: 2.0 type: double	value: 0.2 min: 0.1 max: 2.0 type: double
`OutputFilter` `maxSize`	Trackables whose largest dimension is larger than this are filtered out.	value: 100.0 min: 1.0 max: 100.0 type: double	value: 100.0 min: 1.0 max: 100.0 type: double	value: 100.0 min: 1.0 max: 100.0 type: double
`OutputFilter` `maxArea`	Trackables whose area (length * width) is larger than this are filtered out.	value: 100.0 min: 1.0 max: 1000.0 type: double	value: 25.0 min: 1.0 max: 1000.0 type: double	value: 100.0 min: 1.0 max: 1000.0 type: double
`OutputFilter` `minSpeed`	Trackables whose speed (m/s) is lower than this are filtered out.	value: 0.0 min: 0.0 max: 5.0 type: double	value: 0.0 min: 0.0 max: 5.0 type: double	value: 0.0 min: 0.0 max: 5.0 type: double
`OutputFilter` `maxSpeed`	Trackables whose speed (m/s) is higher than this are filtered out.	value: 50.0 min: 2.0 max: 100.0 type: double	value: 20.0 min: 2.0 max: 100.0 type: double	value: 50.0 min: 2.0 max: 100.0 type: double
`OutputFilter` `maxPositionStdDev`	Trackables whose position standard deviation along any axis is larger than this are filtered out.	value: 2.0 min: 1.0 max: 5.0 type: double	value: 2.0 min: 1.0 max: 5.0 type: double	value: 2.0 min: 1.0 max: 5.0 type: double

MultiLidarPipeline parameters

`MultiLidarPipeline` is used when multiple sensors are operating. Single sensor data goes through previous pipelines separately, then it is combined into a multi-lidar pipeline that fuses the cluster points from the single-lidar pipelines. `MultiLidarPipeling` also recalculates clustering and tracking. *Table 25. Sensor Settings Parameters for MultiLidarPipeline* describes `MultiLidarPipeling` and other parameters, and their recommended settings.

Table 25. Sensor Settings Parameters for MultiLidarPipeline

Section, Parameter	Description	General	Person	Vehicle
`MultiLidarPipeline` `numWorkerThreads`	The number of worker threads on which to run classification. In environments with many objects, this value can be increased to improve the rate at which objects get classified, provided that the server has enough available cores.	value: 2 min: 1 max: 10 type: int	value: 2 min: 1 max: 10 type: int	value: 2 min: 1 max: 10 type: int
`MultiLidarPipeline` `useSingleLidarTracking`	When enabled, uses Kalman filter models for single-lidar pipelines to predict a fused point cloud. If disabled, skips fused point cloud processing and uses single-lidar clusters and global Kalman filter models directly. Disabling helps process a high number of trackables at the cost of smaller bounding boxes for people, while generating some jitter in visualization.	value: true option: true option: false	value: true option: true option: false	value: true option: true option: false
`MultiLidarPipeline` `useRegulator`	Regulates data flow when useSingleLidarTracking is true. May slow down object list publishing rate when handling high number of trackables, but reduces jitter in Visualizer. Enabling has no effect when useSingleLidarTracking is true.	value: true option: true option: false	value: true option: true option: false	value: true option: true option: false
`MultiLidarPipeline` `TrackableFilter`	Trackables coming from the single-lidar pipeline pass through these filters before their points are used to construct the multi-lidar point cloud.			

Section, Parameter	Description	General	Person	Vehicle

`MultiLidarPipeline`
`TrackableFilter`
`minMeasCount`
	Trackables not measured at least this number of times are filtered out.	value: 2
		min: 1
		max: 5
		type: int
			value: 2
			min: 1
			max: 5
			type: int
				value: 5
				min: 1
				max: 5
				type: int

`MultiLidarPipeline`
`TrackableFilter`
`minClusterPoints`
	Trackables whose latest measurement does not contain at least this number of points are filtered out.	value: 2
		min: 1
		max: 20
		type: int
			value: 2
			min: 1
			max: 20
			type: int
				value: 5
				min: 1
				max: 20
				type: int

`MultiLidarPipeline`
`TrackableFilter`
`minSize`
	Trackables whose largest dimension is smaller than this are filtered out.	value: 0.2
		min: 0.1
		max: 2.0
		type: double
			value: 0.2
			min: 0.1
			max: 2.0
			type: int
				value: 0.3
				min: 0.1
				max: 2.0
				type: int

`MultiLidarPipeline`
`TrackableFilter`
`maxArea`
	Trackables whose area (length * width) is larger than this are filtered out.	value: 100.0
		min: 1.0
		max: 1000.0
		type: double
			value: 25.0
			min: 1.0
			max: 1000.0
			type: double
				value: 100.0
				min: 1.0
				max: 1000.0
				type: double

`MultiLidarPipeline`
`TrackableFilter`
`maxSpeed`
	Trackables whose speed (m/s) is higher than this are filtered out.	value: 50.0
		min: 2.0
		max: 100.0
		type: double
			value: 20.0
			min: 2.0
			max: 100.0
			type: double
				value: 50.0
				min: 2.0
				max: 100.0
				type: double

`MultiLidarPipeline`
`TrackableFilter`
`maxPositionStdDev`
	Trackables whose position standard deviation value along any axis is larger than this are filtered out.	value: 2.0
		min: 1.0
		max: 5.0
		type: double
			value: 2.0
			min: 1.0
			max: 5.0
			type: double
				value: 4.0
				min: 1.0
				max: 5.0
				type: double

Anti-Masking parameters

Set error percent high to detect blockage and lifted percent a little lower. The default 80 for error and 50 for lifted should be fine. The idea is to make the two values not too close to one another (>= 10% difference is recommended) to remove the possibility of bouncing.

```
<MaskingDetector>
    <enabled>false</enabled>
    <!-- true to enable, false otherwise -->
    <maskingErrorPercent>80</maskingErrorPercent>
    <!-- The percentage of points with no detected return must be greater than
    this threshold to consider masking (of the sensor) is detected.
    Valid range : 1.0 to 100 % -->

    <maskingLiftedPercent>50</maskingLiftedPercent>
    <!-- The percentage of points with no detected return must be less than or
    fall below this threshold to consider masking (of the sensor) is no longer
    detected.
    Valid range : 1.0 to 100 % -->
</MaskingDetector>
```

Appendix 5. Cartesian Coordinate System

To properly configure QORTEX DTC 2.3, a basic mathematical understanding of the Cartesian coordinate system for defining 3D locations in space is needed. A quick way to remember the correct orientation of the axes defining the coordinate system is by the right-hand rule. See *Figure 114. Coordinate System: Right-Hand Rule*.

- The index finger points forward for the positive X-axis.
- The middle finger points to the side for positive Y-axis.
- The thumb points up for the positive Z-axis.
- The Origin point (0, 0, 0) is on the palm.

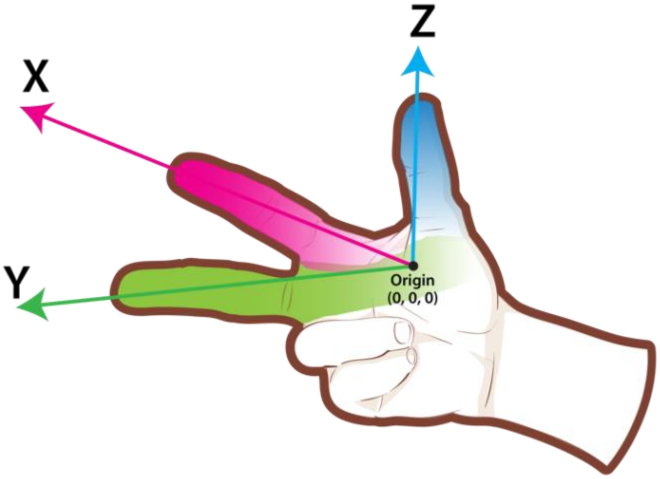


Figure 114. Coordinate System: Right-Hand Rule

> **Note:** In QORTEX DTC 1.x, the coordinate system applied a left-hand rule. This means, the objects x and y coordinates were reversed in QORTEX DTC 1.x object lists.

Every DTC object must share the same frame of reference, that is, the same X-axis, Y-axis, Z-axis, and Origin so that a 3D description in meters (X, Y, Z) expresses precisely one location, such as (3.2, 1.9, -1.8).

In the sensor coordinate system (non-PoE+ version). See *Figure 115. Coordinate System: Non-PoE+ or PoE+ Versions of the Sensor*.

- The front area of the sensor points toward positive X.

  The sensor cable in the back points toward negative X.

- Through the center of the center, perpendicular to the X-axis.

  The right-side points toward positive Y.

  The left side points toward negative Y.

- The top flat face of the sensor points toward positive Z.

  The bottom face of the mounting base points toward negative Z.

- For each sensor, the three axes converge internally at the point of Origin (0, 0, 0).

In the sensor coordinate system (PoE+ version). See *Figure 115. Coordinate System: Non-PoE+ or PoE+ Versions of the Sensor*.

- The connector in the front of the sensor points toward positive X.

  The back points toward negative X.

- Through the center of the center, perpendicular to the X-axis.

  The right-side points toward positive Y.

  The left side points toward negative Y.

- The top flat face of the sensor points toward positive Z.

  The bottom face of the mounting base points toward negative Z.

- For each sensor, the three axes converge internally at the point of Origin (0, 0, 0).

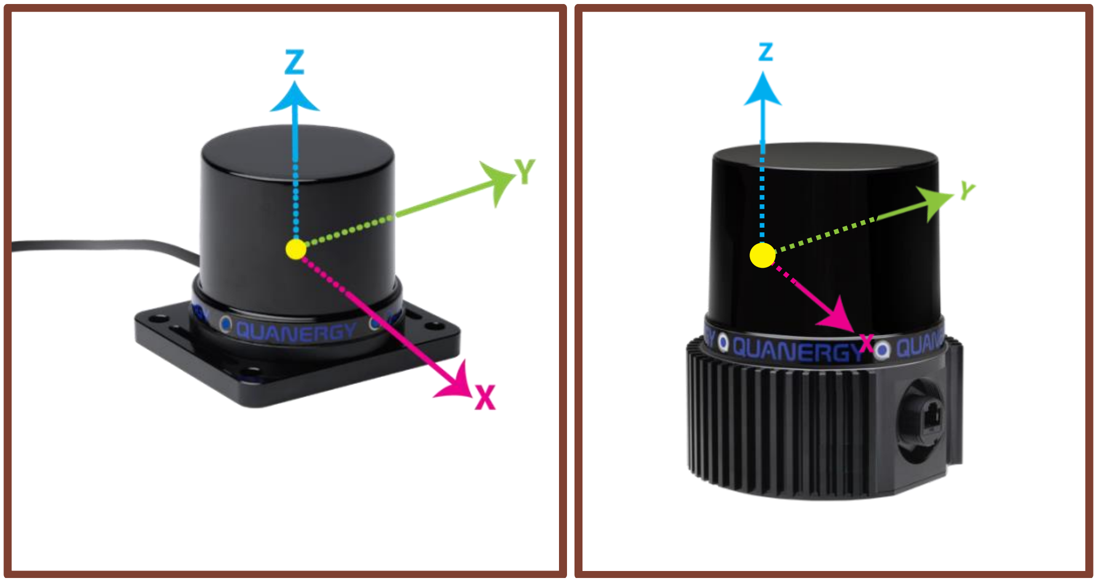


Figure 115. Coordinate System: Non-PoE+ or PoE+ Versions of the Sensor

Appendix 6. QORTEX DTC Fast Track

QORTEX DTC Fast Track is a set of tools that uses source code and business logic to help system integrators and value-added resellers accelerate software development for QORTEX DTC. The Fast Track bundle includes:

- **Geolocation Translator:** converts object location into latitude and longitude, as discussed in the next section.
- **Sensor Coverage Tool:** estimates a sensor coverage area depending on the sensor position.
- **Web Server and Google Maps Tool:** visualizes object location(s) on Google Maps.
- **Port Monitor:** listens and displays the real time output of a TCP port.
- **System Health Monitor:** monitors QORTEX DTC system health and sends an email alert if an abnormality is detected.

To purchase or obtain access to QORTEX DTC Fast Track, please contact Quanergy sales or support@quanergy.com.

Geolocation Translator

The Geolocation Translator provides the absolute latitude and longitude of an object. The script reads the object list and zone list from QORTEX DTC. It uses the location of the QORTEX DTC server World Origin to determine the latitude and longitude of objects in the object list.

The output is broadcast through sockets. Available output formats include XML and JSON.

The script, which currently supports a single QORTEX DTC server, provides the following information:

- Object ID and classification
- Object position in x, y, z locations, and in latitude and longitude
- Object velocity
- Object size
- Zones (contains all zones in which the object is present)
- Timestamp

Requirements

The Geolocation Translator script requires a computer with the following list of items installed or updated. See *Figure 116. Geolocation Translator: Required Updates to settings.xml File*.

Ubuntu 20.04 operating system

QORTEX DTC 2.x or QORTEX DTC 1.2 Server

Python 2.7 or above, installed as follows:

1. Acquire the pip for Python 2.7 package or the pip for Python 3 package:

   For Python 2.7, install with this script:

```
   -- sudo apt-get install python-pip
```

   For Python 3, install with this script:

```
   -- sudo apt-get install python3-pip
```

2. Acquire the lxml package, and install it with this script:

```
   -- sudo pip install lxml
```

3. Acquire the configparser package, then install it with this script:

```
   -- sudo pip install configparser==4.0.2
```

   `settings.xml` **file** is updated.

4. Run the following to specify and open whichever `settings.xml` file is in use:

```
   $ sudo gedit /home/quanergy/quanergy/
      qortex/location/<settings>.xml
```

5. In the text editor, update values where indicated.

   - **Format**: json or xml
   - **Port**: for Object list is 17171 and for Zone list is 17172
   - **AddDataSize**: true

6. Save and close the file.

```
<Publisher>
   <Name>QORTEX1_OBJECT_LIST</Name>
   <Format>json</Format>
   <Port>17171</Port>
   <AddDataSize>true</AddDataSize>
   <NetworkByteOrder>false</NetworkByteOrder>
</Publisher>
<Publisher>
   <Name>QORTEX1_ZONE_LIST</Name>
   <Format>json</Format>
   <Port>17172</Port>
```

```
    <AddDataSize>true</AddDataSize>
    <NetworkByteOrder>false</NetworkByteOrder>
</Publisher>
```

Figure 116. Geolocation Translator: Required Updates to `settings.xml` File

Preparation

Before using the Geolocation Translator, make these preparations:

1. Download/clone the Geolocation Translator script folder from GitHub to the same host machine where the QORTEX DTC server is installed, or to a machine that is on the same network as the QORTEX DTC server machine.
2. Unzip the downloaded folder, navigate to the Geolocation Translator folder, and list the contents. See *Figure 117. Geolocation Translator: Folder*.

```
user:~/Downloads/qguard_geolocation_translator$ ls
config qguard_geolocation_translator.py quanergy
```

Figure 117. Geolocation Translator: Folder

3. Navigate to the config folder, open the config.ini file in a text editor, update the following values if necessary, then save and close the file. See *Figure 118. Geolocation Translator: config.ini File Updates*.
    - **ip_addres**s: of the QORTEX DTC server
    - **message_format**: json or xml
    - **lat**: the latitude of the World Origin
    - **long**: the longitude of the World Origin
    - **heading**: current orientation (direction of the X-axis) of the sensor in degrees

```
[QGuard]
ip_address=127.0.0.1
object_list_port=17171
zone_list_port=17172

[Broadcaster]
port=18181
send_packet_size=true
# file format can be in either xml or json
message_format=json

[Primary Sensor]
lat=37.768943
long=-122.389166
heading=0
```

Figure 118. Geolocation Translator: config.ini File Updates

Execution

Once the config parameters have been edited, execute the Geolocation Translator script. Run the `qguard_geolocation_translator.py` script in one of the following ways. See *Figure 119. Geolocation Translator: Connection and Output*.

```
$ python qguard_geolocation_translator.py
```

or

```
$ ./qguard_geolocation_translator.py
```

The return is the same regardless.

Connect to port 18181 to stream output with this command.

```
$ nc 127.0.0.1 18181
```

An example of the output.

```
user:~/Downloads/qguard_geolocation_translator$ python
qguard_geolocation_translator.py
Attempting connection to 127.0.0.1 on port 17171
Attempting connection to 127.0.0.1 on port 17172
Connected to server at 127.0.0.1 on port 17171
Connected to server at 127.0.0.1 on port 17172

user:~$ nc 127.0.0.1 18181

{
   "objectClass": "VEHICLE",
   "timestamp": "1586206199366",
   "zones": [
      "62"
   ]
   "position": {
      "y": -23.0818291,
      "x": 41.2743149,
      "z": 1.20757973,
      "lat": 37.76931430954076,
      "long": -122.38942868433801
   },
   "velocity": {
      "y": -0.144213989,
      "x": -4.59316397,
      "z": 0.224059537,
      "track": -178.2016450122697,
      "speed": 4.6008866393517026
   }
   "id: "3724805"
```

```
    "size": {
        "y": 2.04645191
        "x": 3.47772717
        "z": 1.02826643
    }
}
```

Figure 119. Geolocation Translator: Connection and Output

Sensor Coverage Tool

The Sensor Coverage tool simulates point clouds from multiple sensors to identify the coverage for an area. The tool considers no obstacles or objects in the area. It supports and simulates M-Series sensors.

Requirements

The Sensor Coverage tool requires a computer running one of the following to operate:

- Python 2.7 (`https://www.python.org/downloads/release/python-2710/`)
- Python 3.7 (`https://www.python.org/downloads/release/python-370/`)

Preparation

Before using the Sensor Coverage tool, make these preparations:

1. Download/clone the `sensor_coverage` folder from GitHub to the same host machine where the QORTEX DTC server is installed, or to a machine that is on the same network as the QORTEX DTC server machine.

2. Unzip the downloaded folder, navigate to the `sensor_coverage` folder, and list the contents. See *Figure 120. Sensor Coverage Tool: Folder*.

```
user: sensor_coverage-master user$ ls
LICENSE       generate_lidar_rings.py
README.md     sensor_coverage.py
```

Figure 120. Sensor Coverage Tool: Folder

3. Edit the `sensor_coverage.py` script in that folder to update the following sensor parameters, if necessary. See *Figure 121. Sensor Coverage Tool: python Script Updates and Additions*.

   Type: specify the sensor as M8 or MQ-8.

   Range: effective sensor range, measured in meters, used for the simulation.

   x, y, z: position of the sensor in space, measured in meters. See Appendix 4: *Cartesian Coordinate System* (page 292) for information about defining 3D locations in space.

   yaw_angle: rotation around the Z-axis of the sensor, measured in degrees.

pitch_angle: pitch and tilt relative to the sensor, measured in degrees.

A positive (+) value refers to the measurement of tilt down.

A negative (–) value refers to the measurement of tilt up.

roll_angle: rotation around the X-axis of the sensor, measured in degrees.

min_angle: minimum angle of the sensor FOV (Field of View), in degrees.

max_angle: maximum angle of the sensor FOV, measured in degrees.

ground_offset: height off the ground at which points are differentiated from the ground, measured in meters.

target_height: maximum height of the target where points/returns are expected to be captured, measured in meters.

Other editable graphical parameters that may be included in the script, or that you could add are:

```
       plt.axis ('equal')
       plt.xlim (-100, 100)
       plt.ylim (-70, 70)


{"type":"m8","range":85,"x":0,"y":0,"z":4.2, "yaw_angel":385, "pitch_angle":6.4,
"roll_angle":0, "min_angle":-180, "max_angle":180, "color":"green","alpha":0.5}

{"type":"m8","range":85,"x":33,"y":3,"z":4.2, "yaw_angel":200, "pitch_angle":-6.2,
"roll_angle":0, "min_angle":-180, "max_angle":180, "color":"green","alpha":0.5}

{"type":"m8","range":85,"x":8,"y":-24,"z":4.2, "yaw_angel":80, "pitch_angle":-6.0,
"roll_angle":0, "min_angle":-180, "max_angle":180, "color":"green","alpha":0.5}

{"type":"m8","range":85,"x":33,"y":-24,"z":4.2, "yaw_angel":170,"pitch_angle":6.4,
"roll_angle":0, "min_angle":-180, "max_angle":180, "color":"green","alpha":0.5}


       plt.axis ('equal')
       plt.xlim (-100, 100)
       plt.ylim (-100, 100)
```

Figure 121. Sensor Coverage Tool: python Script Updates and Additions

Execution

Once the LiDAR and other parameters have been edited, execute the following script:

```
$ python sensor_coverage.py
```

Sensors are plotted on a graph in an image which includes the following elements. See *Figure 122. Sensor Coverage Tool: Graph*.

- **Blue** dots indicate sensors.
- **Light green** areas indicate a single beam from a particular sensor.
- **Dark green** areas indicate the overlap of two or more beams from two or more sensors.
- **White** areas in between beams indicate no beam coverage from any sensor.

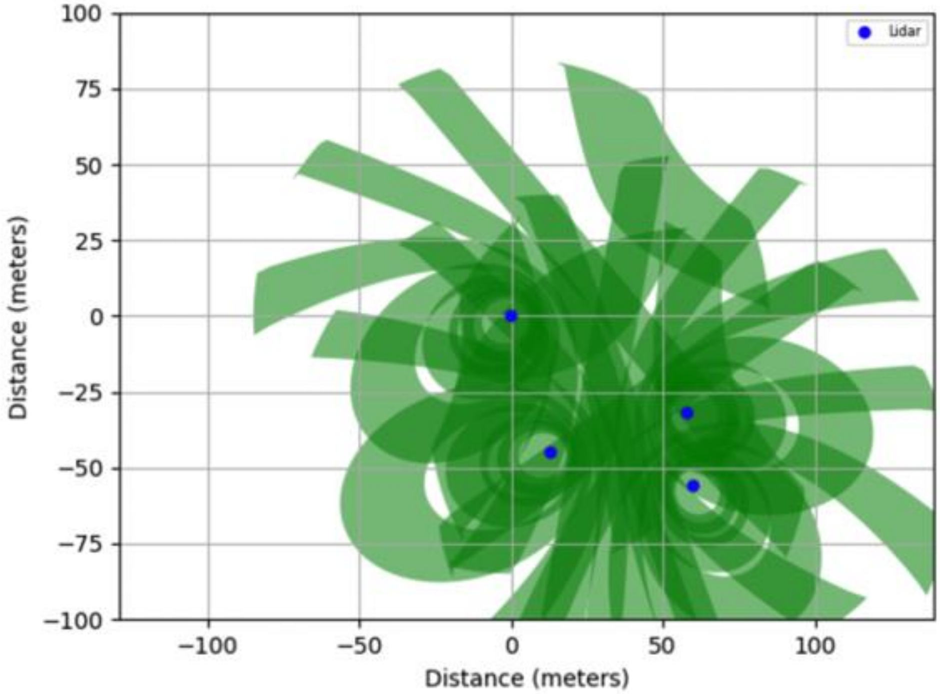


Figure 122. Sensor Coverage Tool: Graph

Web Server and Google Maps Tool

A Web Server serves the Object list streamed from one or more QORTEX DTC servers to a web client that visualizes the objects on Google Maps.

Requirements

The Web Server and Google Maps Tool requires a computer running the following:

- **Ubuntu 20.04** operating system

- **node.js**

  To install node.js, go to https://github.com/creationix/nvm

  To install node.js through NVM, go to the same web page

- **Google Maps API Key** installation instructions are at:

  https://developers.google.com/maps/documentation/javascript/get-api-key

Preparation

Before using the Web Server and Google Maps Tool, make these preparations:

1. Download the Qortex_DTC_WebClient, then unzip the file.
2. Open a terminal window, and navigate to the following folder:

```
Qortex_DTC_WebClient-master/src/clients/google_maps
```

3. Run:

```
npm run build
```

4. Open index.html in a text editor, replace *<insert_your_key>* with the proper Google Maps API Key, save the file, and exit the editor.
5. Navigate to the src/server folder, then run:

```
npm install
```

Configuration

Configure the Web Server and Google Maps Tool as follows:

1. Go the folder Qortex_DTC_WebClient-master/config.
2. Open the config.ini file in a text editor, and update these values, as needed:

   type: the name of the web client served by the Web Server.

   ip_address: the QORTEX DTC server IP address. For multiple servers, add additional [servers.<name>] sections. Each section requires the information below:

   object_port: the port on which the QORTEX DTC server publishes objects.

   zone_port: The port on which the QORTEX DTC server publishes zones.

   Optional geographic calibration of the QORTEX DTC installation against the world:

   lat: The latitude of the QORTEX DTC installation in the world

   lng: The longitude of the QORTEX DTC installation in the world

   heading: The orientation of the QORTEX DTC installation in the world

Cartesian calibration of the QORTEX DTC installation against the world, if needed. These values depend on the map to display objects on, as follows:

x_position: The x position of the QORTEX DTC installation relative to some point

y_position: The y position of the QORTEX DTC installation relative to some point

z_position: The z position of the QORTEX DTC installation relative to some point

3. Save the file, then exit the text editor.

Execution

Visualization of objects requires the Web Server to start up and keep going, as follows:

1. Run the Web Server in one of the following two ways:

   Execute the `server.sh` shell script, available in the bin folder.

   or

   Run the node on `src/server/src/main.js`.

2. Pass in any of the following options you prefer:

   `-h` or `--help` to get a full list of options.

   `-l` or `--log` to set whether the JSON data received from the QORTEX DTC servers is saved to file. Files are saved to the logs folder in the QORTEX DTC Web Client root folder.

   `-p` or `--port` to specify which port the QORTEX DTC Web Client server uses for HTTP communications. It defaults to the port set in the environment, or 3000 if the port is not set.

   `-c` or `--config` to pass in the location of the config file.

3. Open a web browser, and enter the following into the address bar: 127.0.0.1:3000

4. The message `Server Status: Connected` is displayed, and objects are seen moving on Google Maps. See *Figure 123. Web Server and Google Maps Tool: Example Site*.

5. Press **Ctrl+C** when if you are ready to stop the Web Server.

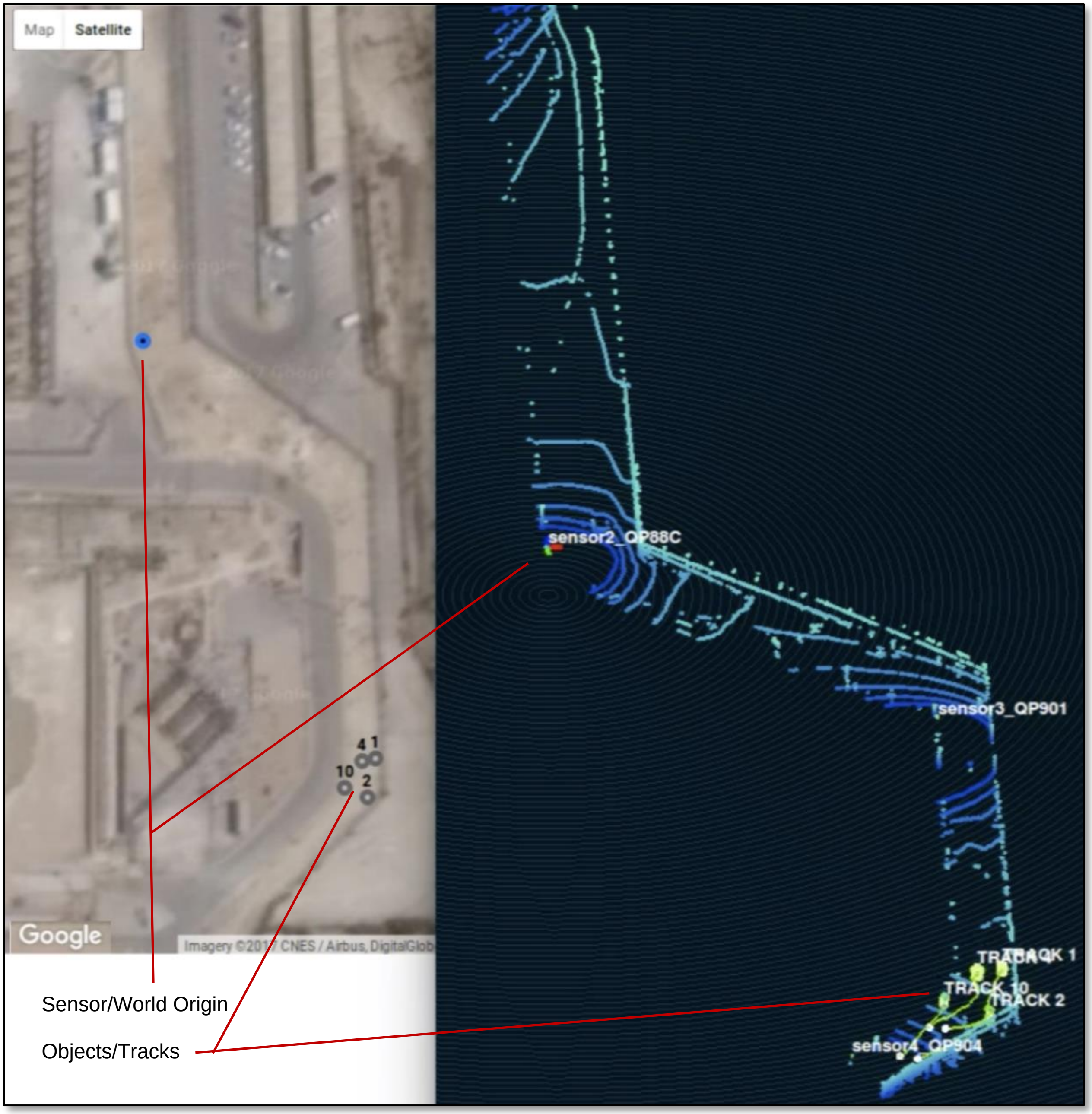


Figure 123. Web Server and Google Maps Tool: Example Site

Port Monitor

The Port Monitor, also called the `qortex_listener` (Listener), is an example to use or adapt. The main routine where the code starts executing exhibits the process flow to create a customized human alert system using the object/trackable list data, for example. See *Figure 124. Port Monitor qortex_listener Main Routine*. This tool uses standard sockets to connect to a port on the server host computer and processes messages that are published there.

```
/* Start here - THESE ARE JUST CODE SNIPS. PLEASE SEE SOURCE APPLICATION. */
int main(int argc, char **argv)
    /* socket: create the socket */
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
    /* Get the server Internet address:port, then connect our socket to the port */
 connect(sockfd, (struct sockaddr *) &serveraddr, sizeof(serveraddr))

 /* Wait for messages and process them */
 processMessage( sockfd );
     /* Every message is preceded by a 4 byte message length, including JSON & XML */
 int bytecount = recv( theSocket, messageBuffer, 4, 0 );
     /* Process the 'protobuf' message, then wait for next */
 if (qobjarr.ParseFromArray( messageBuffer, bytecount )) {
    /* If the format is 'protobuf' use the built-in parser to display the data */
  std::cout << "Parsed QObjects: " << qobjarr.object_size() << " " <<
qobjarr.DebugString();
 }
 else {
    /* If the data is serial text data, just output the text */
  std::cout << "Parsed text object: " << messageBuffer << std::endl;
 }
```

Figure 124. Port Monitor qortex_listener Main Routine

Requirements

There are a few requirements before using the Port Monitor.

1. Have the following set up:

   CMake >2.8.1

   Boost >1.52

   Protocol Buffers v3.5.0 Development Library (https://github.com/google/protobuf/releases/tag/v3.5.0)

2. Install the new Protobuf 3.5.0 library by opening an Ubuntu terminal window, and execute the following commands:

```
   $ ./configure
```

```
    $ sudo make
    $ sudo make install
```

3. Access the source_code/build, and execute the following command to refresh the system environment:

```
    $ sudo ldconfig
```

Preparation

To build the tool, use the make file (included in the source code) on a machine with a C++ compiler.

1. Clone the repository into a directory.
2. Build the application in that directory by executing the following commands:

```
    $ mkdir build && cd build
    $ cmake ..
    $ make
```

The tool will run on any PC, operating system, or any other node on the network so that any third-party machine can receive the data output from QORTEX DTC.

Execution

Start the Port Monitor as follows, providing the IP address and port number of the server host computer, and specifying the type of content expected. See *Server TCP Ports to Monitor* (page 92).

```
$ ./qortex_listener [--host ip] [--port port_number] [-t output_type]
```

If these parameters are not provided, the Port Monitor defaults by trying to connect to localhost (127.0.0.1) on port 17171, where the Object List is output. When the command is executed, the Port Monitor establishes a TCP socket connection and monitors the specified TCP port to directly consume the live (real-time) data and print it to the Ubuntu terminal window in whichever format is specified in the `settings.xml` file.

Use the following executable commands, with the variable (user-specified server host computer IP address) in *`brown`*:

For detected objects (1.x):

```
$ ./qortex_listener -h x.x.x.x -p 17171 -t qobject
```

For detected trackables (2.0):

```
$ ./qortex_listener -h x.x.x.x -p 17161 -t qtrack
```

For defined zones:

```
$ ./qortex_listener -h x.x.x.x -p 17172 -t qzone
```

For current sensor status (1.x):

```
$ ./qortex_listener -h x.x.x.x -p 17168 -t qsensorstate
```

For current sensor status (2.0):

```
$ ./qortex_listener -h x.x.x.x -p 17178 -t qstate
```

Press **Ctrl+C** to terminate the Port Monitor.

Encrypted TCP Port Messages

If **Security** mode is enabled, TCP port messages are encrypted. See *Figure 125. TCP Encrypted and Decrypt with Security Mode Enabled.*

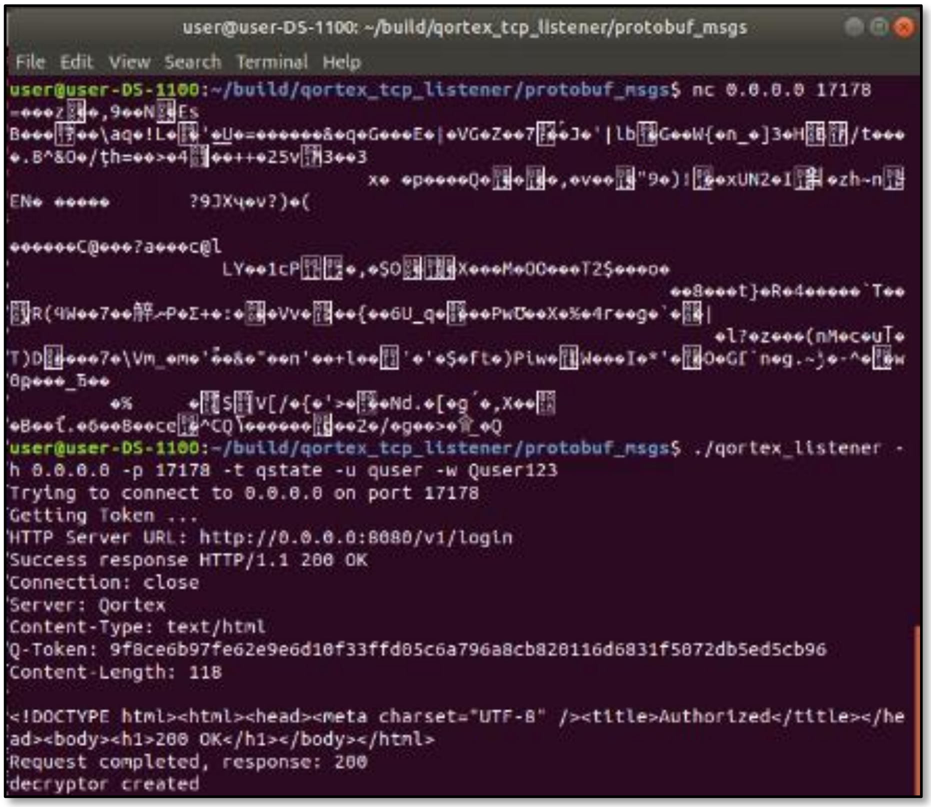
To view unencrypted TCP port messages using the Quanergy TCP listener, `qortex_listener`:

1. From the command line of your QORTEX DTC server, change to the `/Qortex_tcp_listener` directory for the message format selected in your TCP port settings. See *Edit TCP Port Publishing Settings* (94).
2. Enter the command to view the port messages.

```
$ nc <ip_address> <tcp_port>
```

3. If the output is encrypted, use the key from HTTP digestive to decrypt the published message. This includes the TCP type reference and security credentials: username and password. For example:

```
$ ./Qortex_listener -h 0.0.0.0 -p 17171 -t qobject -u <username> -w <password>
$ ./Qortex_listener -h 0.0.0.0 -p 17172 -t qzone -u <username> -w <password>
```



user@user-DS-1100: ~/build/qortex_tcp_listener/protobuf_msgs
File Edit View Search Terminal Help

```
user@user-DS-1100:~/build/qortex_tcp_listener/protobuf_msgs$ nc 0.0.0.0 17178
-���z��,9��N��Es
B�����\aq�!L���'�U�=������&�q�G���E�|�VG�Z��7���J�'|lb��G��W{�n_�]3�H���/t���
�.B^&O�/ţh=��>�4���++�25V�3��3
                                    x� �p����Q������,�v�����"9�)!���xUN2�I���zh-n�
EN� �����       ?9JXч�v?)�(

������C@���?a���c@l
                      LY��1cP����,�$O����X���M�OO���T2$����o
                                                                      ��8���t}�R�4����� `T��
�R(ЧW��7��辭~P�Σ+�:����VV�����{��6U_q����PwƲ��X�%�4r��g�`���|
                                                                            �l?�z���(nM�c�uT�
T)D����7�\Vm_�m�'��&�"��n'��+l����'�'�$�ft�)Piw���W���I�*'���O�GΓ`n�g.~ǰ�-^���w
0ҏ���_ƃ��
           �%      ���$��V[/�{�'>����Nd.�[�g´�,X���
�B��ſ.�б��B���c���^CQ˥������Z�/�g��>��_�Q
user@user-DS-1100:~/build/qortex_tcp_listener/protobuf_msgs$ ./qortex_listener -
h 0.0.0.0 -p 17178 -t qstate -u quser -w Quser123
Trying to connect to 0.0.0.0 on port 17178
Getting Token ...
HTTP Server URL: http://0.0.0.0:8080/v1/login
Success response HTTP/1.1 200 OK
Connection: close
Server: Qortex
Content-Type: text/html
Q-Token: 9f8ce6b97fe62e9e6d10f33ffd05c6a796a8cb820116d6831f5072db5ed5cb96
Content-Length: 118

<!DOCTYPE html><html><head><meta charset="UTF-8" /><title>Authorized</title></he
ad><body><h1>200 OK</h1></body></html>
Request completed, response: 200
decryptor created
```

Figure 125. TCP Encrypted and Decrypt with Security Mode Enabled

System Health Monitor

The System Health Monitor checks the status of various sensor health parameters and sends email alerts about problems.

Requirements

The System Health Monitor requires a computer with the following:

- **Ubuntu 20.04** operating system on a native machine or virtual machine
- **Python 2.7**
- **pip install pyyaml**
- **Postfix** : sudo apt-get install postfix
- QORTEX DTC server publishing JSON data on ports 17171, 17172, 17168, and 18181 (for the Geolocation Translator)

Preparation

Before using the System Health Monitor, make these preparations:

1. Download/clone the System Health Monitor script folder from GitHub to the same host machine where the QORTEX DTC server is installed, or to a machine that is on the same network as the QORTEX DTC server machine.

2. Unzip the downloaded folder, then navigate to the System Health Monitor folder.

3. Open the `config.yaml` file in a text editor, update the following parameters if needed, then save and close the file.

   Site > Streams > geo

   - **host**: 10.36.141.131 (IP address of machine running Geolocation Translator)
   - **port**: 18181 (if streaming port number changed, update this)
   - **docker_image**: qgeo_image
   - **docker_container**: qgeo-server

   QORTEX

   - **host**: 10.36.141.131 (IP of machine running the QORTEX DTC server)
   - **port**: 17171 (if streaming port number changed, update this)
   - **sensor_status**: 17168 (if streaming port number changed, update this)
   - **docker_image**: `Qortex_server_image`
   - **docker_container**: `qortex-server`

   alert_email_sender: email address of the sender

   alert_email_list: email address list of recipients

4. Open a terminal window, and give the `monitor_email_only` script executable rights:

```
    $ sudo chmod 777 monitor_email_only.py
```

Execution

In a terminal window, run the System Health Monitor script in one of the following ways:

```
$ ./monitor_email_only.py
```

or

```
$ python monitor_email_only.py
```

The script checks parameters and completes tasks in the following order:

1. Checks QORTEX DTC object and health/status streams, M-Series sensors, and the geolocation translator script. The script sends an email if any of the streams are unavailable or missing, or if the sensor(s) do not respond to ping requests.

2. Checks data on port 18181 of the geolocation translator script.

   If data on port 18181 is missing, the script checks for data stream on port 17171 (Object List) from QORTEX DTC 1.2/2.0. If the stream is unavailable or missing on port 17171, the script sends an alert email stating: `Qortex was not sending out data at 17171 on $Server_IP. Manual Intervention needed.`

If the data stream is active and available on port 17171 but missing on ports 18181, the script sends an alert email stating: `Geo translator was not sending out data at 18181 on 10.36.141.131. Qortex seems to be running fine. Manual intervention needed!`

3. Checks data on port 17168 (State List) of QORTEX DTC 1.2/2.0.

   If the State List is unavailable or missing, the script sends an alert email stating:

```
Sensor status is not being published by Qortex on port: 17168
Manual intervention needed!
```

   If the data stream is active on 17168, the script searches for sensor(s) with a status of “not connected.”

4. Pings sensor(s) connected to QORTEX DTC 1.2/2.0 at defined time intervals.

Appendix 7. Automated ID Handover

Automated ID Handover™ (AIDH) is an application that unifies the output from multiple QORTEX DTC servers. The application connects to multiple QORTEX DTC servers and combines the individual results into a single output that is in a format identical to the QORTEX DTC output. The object IDs are consistent across the servers (that is, an object traveling from one server area to another server area preserves its object ID). Duplicate objects are suppressed.

As of Automated ID Handover version 2.3, server failure resilience is added. If one of the Qortex DTC servers fails, Automated ID Handover continues to seamlessly run.

This application is available from your support representative at support@quanergy.com.

Supported Versions

As of QORTEX DTC 2.3, Automated ID Handover 2.3 versioning is in sync with QORTEX DTC versioning.

Table 26. Automated ID Handover Version Compatibility

AIDH version	QORTEX DTC version	Features
2.3	2.x,1.x	State List and Counter Line List. 1.x and 2.x support on all ports. Support: NDJSON, IPv6, Ubuntu 20.04
2.2.1	2.x, 1.x	Server resilience when a server is down. 2.x trackables list and 1.x objects list
1.x	1.x	1.x objects list

AIDH 2.3 is backward compatible with QORTEX DTC 2.1. That is, the QORTEX DTC 2.1 features are supported when you are using AIDH version 2.3.

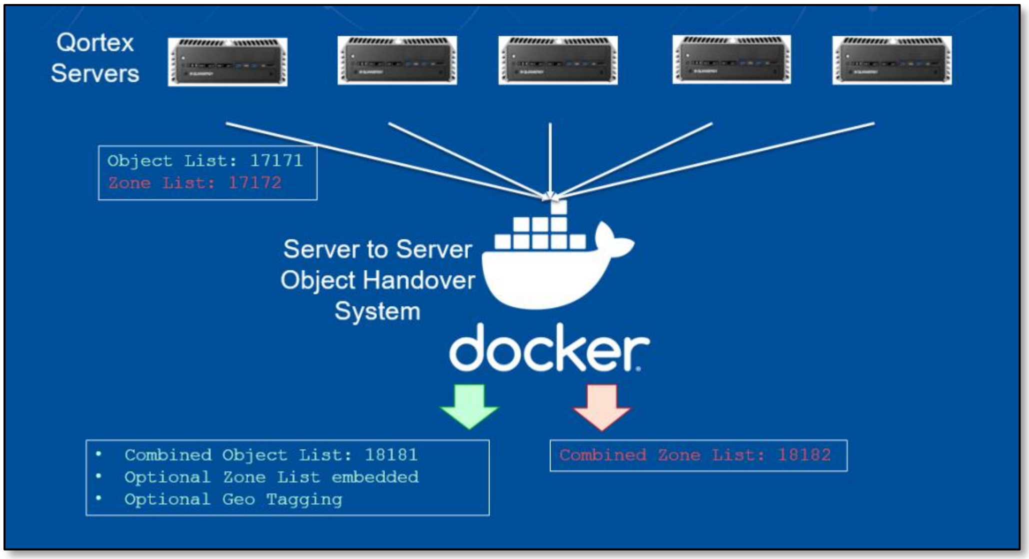
Minimum Requirements

The application is packaged as a Docker image and therefore needs to run in a Docker container. At a minimum, the application also requires the following:

- Hardware: Desktop or laptop with i3 processor, 4GB Memory.
- Operating System: Ubuntu 20.04 LTS or VMware ESXi6.5.

System Architecture

For an example of the system architecture, which uses a Docker container, see *Figure 126. Automated ID Handover System Architecture*.



Qortex Servers
Object List: 17171
Zone List: 17172
Server to Server Object Handover System
docker
• Combined Object List: 18181
• Optional Zone List embedded
• Optional Geo Tagging
Combined Zone List: 18182

Figure 126. Automated ID Handover System Architecture

System Design

This application is designed for the following required minimum configuration:

- Multiple servers, with at least one sensor per server.
- At least one overlap area is between the servers.
- Sensors must be calibrated.

For an example with two servers, and each server has four sensors, see *Figure 127. Automated ID Handover System Design and Configuration*. The servers have an overlap area (the exchange rectangle, shown in **orange**).

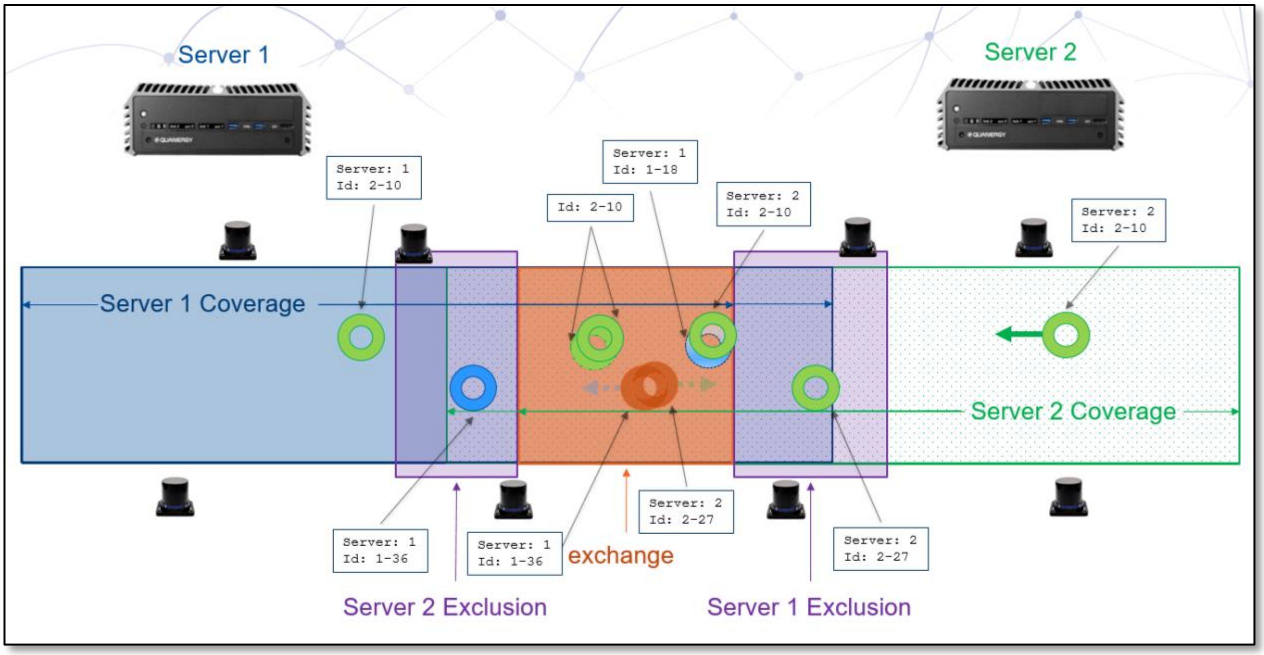


Figure 127. Automated ID Handover System Design and Configuration

Calibration

Prepare all the sensors that will be involved in the Automated ID Handover application and set up their location as follows. Refer to the *Q-View User Guide*, which is available at https://downloads.quanergyworks.com/.

1. In Q-View configure the sensors.

   a. Connect all the sensors in Q-View to a single server. See *Using the Dashboard Tab* in the *Q-View User Guide*.

   b. Calibrate all of the sensors in that group together. See *Using the Calibrate Tab* in the *Q-View User Guide*. This process creates two files:

      - `calibration.ini` file, which is used by QORTEX DTC 1.2
      - `transform_alignment.xml` file, which is used by QORTEX DTC 2.3.

2. In QORTEX DTC client set up a multi-sensor location. Select **Configuration** panel **> Sensor** tab **> Multiple**.

   See *Add or Update Multiple Sensors at a Location* (page 89).

   a. Connect to the first QORTEX DTC server. Select the `transform.alignment` file. Select a single sensor and create a location.

   b. Connect the second QORTEX DTC server. Select the same `transform.alignment` file. Select a different single sensor and create a location.

Configuration

In the `config.yaml` file of settings parameters that was provided with this application, fill in the server information with values for your setup. Explanations and example values are listed below. Other settings values can be customized by the user.

> **Note:** The concept of primary sensor, which is used in this procedure, is QORTEX DTC 1.x legacy language for what QORTEX DTC 2.3 calls World Origin. Both terms are referring to the centerpoint of the geolocation.

```
qortex_server_list:
  10.1.52.209:
    server_name: server1
    server_type: qortex
    port_list:
      object_port_1x: 17171
      object_port_2x: 17161
      zone_port: 17172
      state_port_1x: 17178
      state_port_2x: 17168
      line_port_2x: 17163
  10.1.52.126:
    server_name: server2
    server_type: qortex
    port_list:
      object_port_1x: 17171
      object_port_2x: 17161
      zone_port: 17172
      state_port_1x: 17178
      state_port_2x: 17168
      line_port_2x: 17163

# Object Handover Details, active is always 1 for now. Please list the name(s) of
the exchange zone(s).
# Just one name per overlap as the zone names are identical for the same overlap
area in different servers.
object_handover:
    active: 1
    exchange_zones:
        - exchange_zone

#Qortex encryption block. Be sure that the username and password is the same across
all qortex servers
authentication:
  secure_mode: 0
  username: user00
  password: Letmein433!
  uuid: 3debc6c9-08cc-7a02-9af0-1a082c39f4d2
  primary_server: '10.1.52.209'

#Port versioning. Set to 0 for 1x and 1 for 2x for respective port types
port_version:
  ol_ver: 0
  zl_ver: 0
  sl_ver: 1
  ll_ver: 1

# Broadcaster details. Please leave the ports as is. Changing the ports will also
require a change in Docker run command.
# add_geo is 1 for adding geo location information in the object list
# add_zones is 1 for embedded zone list in the object list.
# broadcast_frequency is in Hz you want to receive the data output.
broadcaster:
    ol_port: 18181
    zl_port: 18182
    sl_port: 18168
    ll_port: 18163
    add_geo: 0
    add_zones: 0
    send_packet_size: 0
    message_format: json
    broadcast_frequency: 10
    nd_json: 0
    IPv6: 0
```

Figure 128. Sample Automated ID Handover `config.yaml` File

Broadcaster Parameter

These parameters relate to the format of the messages that are broadcast by the application.

- **Object list port** (ol_port): Port on which the object list is published.
- **Zone list port** (zl_port): Port on which the zone list is published.
- **State list port** (sl_port): Port on which state list is published.
- **Line list port** (ll_port): Port on which line list is published.
- **Add geolocation** (add_geo): 0-Disable, 1-Enable

  When enabled, geo location details become part of the object data.
- **Add zones** (add_zones): 0-Disable, 1-Enable

  When enabled, zones data is embedded in the object data.

- **Send packet size** (send_packet_size): 0-Disable, 1-Enable

  When enabled, packet size is included in the message.

- **Broadcast frequency** (broadcast_frequency): Values (1-10) Hz

  This is the object and zone list broadcasting frequency. It can be set from a low value to a high value that is equivalent to the highest data output frequency set in any one of the connected servers. Recommended value is same as QORTEX DTC frequency or lower.

  - `nd_json`: When enabled switches broadcast format to NDJSON
  - `ipv6`: When enabled, allows broadcasting on IPv6 system

```
broadcaster:
    ol_port: 18181
    zl_port: 18182
    sl_port: 18168
    ll_port: 18163
    add_geo: 0
    add_zones: 0
    send_packet_size: 0
    message_format: json
    broadcast_frequency: 10
    nd_json: 0
    IPv6: 0
```

Exchange_zones Parameter

List the name(s) of the exchange zone(s). For best performance, all the servers should have an overlapping area, and an Event zone should be created in the overlap area to be used by the application as an exchange zone. In the example below, the exchange zone is named `exchange_zone`.

Copy the details of this exchange zone into the `settings.xml` file in use by the other QORTEX DTC servers to ensure that the exchange zones are identical for every server.

Any other overlapping area between the two servers outside of the exchange zone should be masked by Exclusion zones in at least one of the servers.

```
exchange_zones:
 - exchange_zone
```

Primary_sensor geolocation Parameter

Use any accurate GPS device to discover and provide the primary sensor latitude, longitude, and heading. The heading value, in degrees, is the direction of the X-axis of the system as determined by the GPS or a compass. (This center point geolocation data provides accuracy for the objects, but it is only used if the `add_geo` parameter is enabled in the broadcaster section below. If the `add_geo` parameter is disabled, then this geolocation data is not added to the object list.)

```
primary_sensor:
 lat: 37.767057
 long: -122.389331
 heading: 175
```

Object_handover Parameter

The `object_handover` active parameter is used to enable or disable the object handover functionality.

active: `0-Disable, 1-Enable`

```
object_handover:
 active: 1
```

Port Versioning Parameter

For AIDH 2.3 and above, all ports support 1.x and 2.x port lists.

> **Note:** QORTEX DTC version 1.x defaults to zone lists. QORTEX DTC 2.x defaults to line lists.

Example: 0 - use 1.x objects list, 1 - use 2.x object trackables list.

Use QORTEX DTC 2.x object trackables list

```
port_version:
  ol_ver: 1
  zl_ver: 0
  sl_ver: 1
  ll_ver: 1
```

Use QORTEX DTC 1.x object list

```
port_version:
  ol_ver: 0
  zl_ver: 0
  sl_ver: 1
  ll_ver: 1
```

Server_list Parameter

Provide the QORTEX DTC server IP addresses. Any server in the setup can be the primary server. QORTEX DTC supports both IPv4 and IPv6 IP addresses. Support for server aliases or server names is also included.

- `server_name`: Server Name allows for server aliases
- `server_type`: Server type options are: `qortex`, `mock`
- `port_list`: Port list sets the port for each type of QORTEX DTC list. Can be configurable.

```
qortex_server_list:
  10.1.52.209:
    server_name: server1
    server_type: qortex
    port_list:
      object_port_1x: 17171
      object_port_2x: 17161
      zone_port: 17172
      state_port_1x: 17178
      state_port_2x: 17168
      line_port_2x: 17163
  10.1.52.126:
    server_name: server2
    server_type: qortex
    port_list:
      object_port_1x: 17171
      object_port_2x: 17161
      zone_port: 17172
      state_port_1x: 17178
      state_port_2x: 17168
      line_port_2x: 17163
```

Deployment

Run the Automated ID Handover in a Docker container as follows:

1. Download the AIDH docker image, `aidh_2.3.1.tar.gz`, and the configuration file, `config.yaml`, to the host.
2. Load the docker image on the host.

```
$ docker load < aidh_2.3.1.tar.gz
```

3. Verify the docker image is deployed to the host.

```
$ docker images
```

   Scan the return to see that `aidh` with tag `2.3.1` is part of the docker images command.

4. Update the `config.yaml` file with:
   - QORTEX DTC server information.
   - Set the `primary_server` under authentication to the host IP of the primary QORTEX DTC server.
5. Run the `memcached` docker container. The Memcached container is a prerequisite to running the AIDH container.

When you run the command for the first time, the Memcached image is pulled from the Docker hub.

```
$ docker run --net=host -dit --name memcached bitnami/memcached:latest
```

4. Verify Memcached container is running.

```
$docker ps -a
```

5. Run the AIDH container from the docker image.

   Make sure to provide the complete path for the `config.yaml` file on the host.

```
$ docker run --restart-always --net=host -v
<path_to_config.yaml_file>:/opt/quanergy/aidh/config/config.yaml
-dit --name aidh aidh:2.3.1
```

   For example, if `config.yaml` is located at `/home/user/aidh/`.

```
$ docker run --restart-always --net=host -v
/home/user/aidh/config.yaml:/opt/quanergy/aidh/config/config.yaml
-dit --name aidh aidh:2.3.1
```

6. Verify docker container `aidh` is running.

```
$ docker ps -a
```

7. Verify the object data output.

```
$ netcat 127.0.0.1 18181
```

Common AIDH Docker Commands

- View the AIDH logs.

```
$ docker logs aidh
```

- Stop and remove the AIDH container.

```
$ docker rm -f aidh
```

- Remove the AIDH docker image.

  Stop and remove the AIDH container first.

```
$ docker rm1 -f aidh:2.x.x
```

- Stop and remove the Memcached container.

```
$ docker rm -f memcached.
```

- Apply changes to `config.yaml`.

  These commands start the AIDH container with the new configuration.

```
$ docker rm -f aidh
$ docker run --restart=always --net=host -v
<path_to_config.yaml_file>:/opt/quanergy/aidh/config/config.yaml
-dit --name aidh aidh:2.3.1
```

- Stop Memcached service running on the host.

```
$sudo systemctl stop memcached.service
```

Working with Secure Mode Functionalities

> **Note:** Currently the Security mode feature only supports multiple servers under one set of credentials.

The encryption feature is not available for version 2.1 of either AIDH or QORTEX DTC.

AIDH Security Mode Requirements

To use the Security mode (encryption) requires:

- Both the QORTEX DTC server 2.3 or later and AIDH 2.3 or later.
- Both QORTEX DTC server and AIDH must have **Secure** mode enabled. See *Toggle Security Mode On/Off* (page 64) and *Secure TCP Port Data* (page 96).

Use Security mode features of QORTEX DTC

In Automated ID Handover, complete the following.

1. Add the following authentication section at the bottom of the existing `config.yaml` file.

   The authentication, username and password vary based on user, but `uuid` is always the same.

```
authentication:
  secure_mode: 0
  username: user00
  password: Letmein433!
  uuid: 3debc6c9-08cc-7a02-9af0-1a082c39f4d2
  primary_server: '10.1.52.209'
```

2. Apply the `config.yaml` changes to the AIDH container.

   These commands start the AIDH container with the new configuration.

```
$ docker rm -f aidh
$ docker run --restart=always --net=host -v
<path_to_config.yaml_file>:/opt/quanergy/aidh/config/config.yaml
-dit --name aidh aidh:2.3.1
```

3. If **Security** mode is properly enabled, the QORTEX DTC **Sign-In** pop-up appears. Enter the credentials listed in the `config.yaml` file.

In Qortex DTC, complete the following.

1. Ensure the admin credentials are identical across all QORTEX DTC servers. This is required for AIDH support.

2. Set QORTEX DTC to **Security** mode.

   - If using API, set QORTEX DTC `secure_mode` to 1.
   - If using QORTEX DTC client, **Security** mode in **enabled**.

3. Ensure **Secure TCP Ports** is enabled.

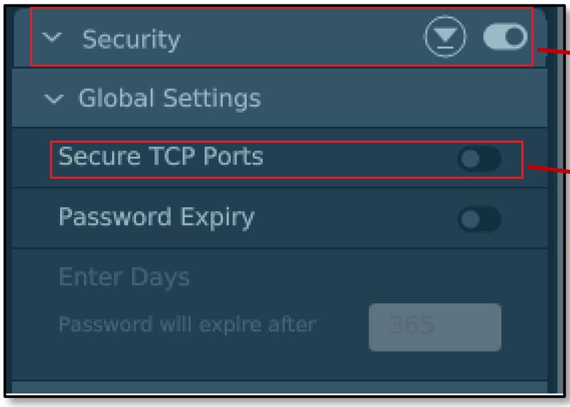
   When using the encryption feature, AIDH 2.3 gets the encryption key from port 8080 of the QORTEX DTC server, as listed in the Automated ID Handover (AIDH) configuration file, `config.yaml`.

   a. Select the **Configuration** panel **> Server** tab **> Security** tab **> Enabled** toggle.

      This toggle switches between secure/unsecure modes.

   b. Select the **Configuration** panel **> Server** tab **> Security** tab **> Global Settings** panel **> Secure TCP Ports** toggle.

      This toggle ensures data transmits through secure TCP ports.



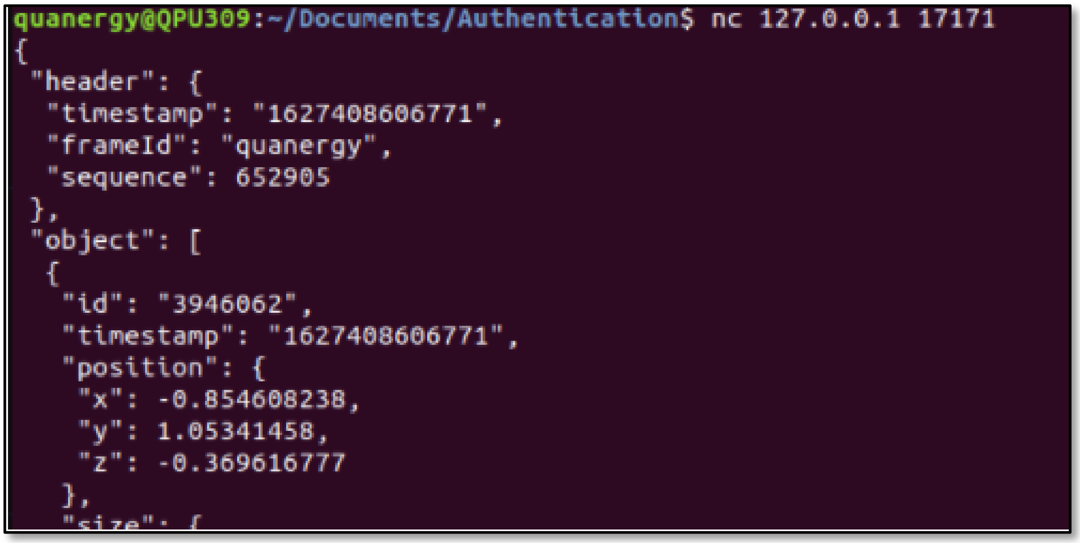
Security mode toggle, indicates Security mode enabled

Security TCP Ports toggle, indicates Security mode or TCP Ports is not enabled

Figure 129. Security mode and Secure TCP Ports Toggles

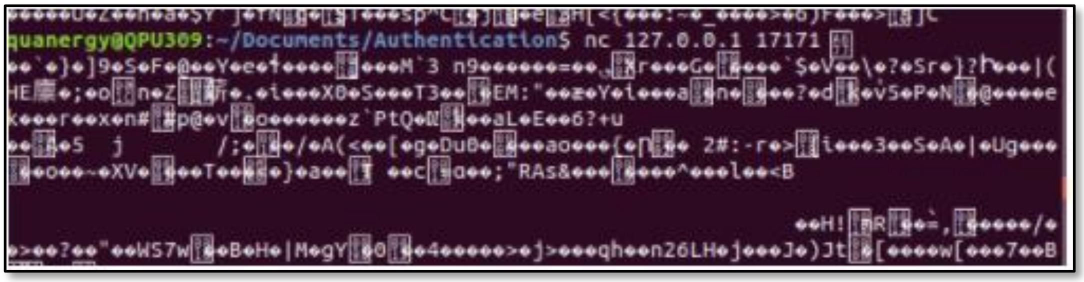
4. Ensure Qortex DTC is broadcasting encrypted data.

5. For decrypting data, ping the HTTPS server using the QORTEX DTC primary server IP address and acquire the `qtoken` for use with the `decryption` command.



```
quanergy@QPU309:~/Documents/Authentication$ nc 127.0.0.1 17171
{
 "header": {
  "timestamp": "1627408606771",
  "frameId": "quanergy",
  "sequence": 652905
 },
 "object": [
  {
   "id": "3946062",
   "timestamp": "1627408606771",
   "position": {
    "x": -0.854608238,
    "y": 1.05341458,
    "z": -0.369616777
   },
   "size": {
```

Figure 130. Sample Unsecure Mode object_list



```
�����U�Z���n�a��$Y J�YN▯�▯T���sp^C▯�}▯��▯H[<{���:~�_����>��)F���>▯]C
quanergy@QPU309:~/Documents/Authentication$ nc 127.0.0.1 17171 ▯
��`�}�]9�S�F�@��Y���ł����▯���M`3 n9�������=��▯r���G�▯���`$�V��\�?�Sr�}?ȟ���|(
HE▯�;�o▯n�Z▯▯�.�i���X0�S���T3��▯EM:"��z�Y�l���a▯n�▯��?�d▯�v5�P�N▯@�����e
k���r��x�n#▯#p@�v▯o�������z`PtQ�N▯��aL�E��6?+u
��▯�5  j      /;�▯�/�A(<��[�g�DuB�▯��ao���{�ſ▯� 2#:-r�>▯i���3��S�A�|�Ug���
▯�o��~�XV�▯��T��▯�}�a��▯ ��c▯a��;"RAs&���▯���^���l��<B

                                                    ��H!▯nR▯��,▯����/�
�>��?��"��WS7w▯�B�H�|M�gY▯�0▯�4�����>�j>���qh��n26LH�j���J�)Jt▯�[����w[���7��B
```

Figure 131. Sample of Secure Mode object_list

Quanergy Systems, Inc

433 Lakeside Drive
Sunnyvale, CA 94085
https://quanergy.com

For purchases made directly from Quanergy: contact support@quanergy.com.

For purchases made from a third party such as a value-added reseller or system integrator: contact them for support.

Specifications are subject to change without notice.