Qosium Probe

Qosium Probe is the heart of the Qosium family. It is the software that carries out the actual measurement as instructed by measurement controllers like Qosium Scope, Qosium Scopemon, or even your own developed upon the QMCP API provided by Kaitotek. Without control, Qosium Probe does nothing, and does not consume any processing resources either.

Table of Contents

1. General Information

Qosium Probe provides high measurement performance. Despite this, it's light-weight, and does not limit the device type where it can be installed to. This is one of the main advantages of using Qosium. Unlike with most other high-performance measurement solutions, truly end-to-end measurement all the way to mobile devices is possible with Qosium. Lightness enables using Qosium also in resource constrained environments, such as in IoT networks.

Qosium Probe operates the packet capturing process and performs almost all the measurement related calculations. It does not have a user interface, but instead, *a measurement controller* is required to operate the measurement. The typical deployment is to install Qosium Probe as a system service to the background in a measurement point. The platform, where it is installed, is called *a measurement point*.

Qosium Probe is a multi-thread server capable of supporting a number of simultaneous controllers. All controller sessions and their measurements are independent. Thus, only a single Probe is required per measurement point.

2. Tech Sheet

- Purpose: The core of Qosium, enabling the passive QoS measurements (measurement agent)
- Current main version: 1.7... (Core: 2.30...)
- Delivery type: Binary executable and binary shared libraries
- Tested platforms (Qosium Probe's availability is not limited to those)
  - Windows 7 and newer, including multiple Windows Server versions
  - Linux operating systems with kernel version of 2.6 and newer
  - OpenWRT/LEDE
  - FreeBSD v9 and newer
  - Android: various versions
  - macOS: 10.12 and newer
  - Raspbian Buster
  - Axis network cameras: various versions
  - Teltonika routers: various versions
  - NanoPi router: R1S

    - Labpano Pilot Era: All-in-One 8K 360 VR Camera
- Other supported platforms
    - As long as build environment with GNU C++ version of 4.8 and newer is available, Qosium Probe can typically be provided.
- Installation and Running Requirements
    - Superuser rights to install and run
    - Packet capture library

3. Overhead & Performance

Qosium Probe is a light-weight software. When it’s running in an idle state, not instructed to measure, the resources it takes from the host machine are negligible. When measuring, the CPU and memory usage depend almost directly on the packet rate of the measurement target flows. You can measure fairly high throughputs even on a constrained device, like an older Raspberry Pi. On a modern standard office computer, measuring a traffic stream of 100 000 pkts/s should not be a problem. At this speed, the machine’s 1 Gbit/s LAN link is already filled with traffic.

When there are multiple simultaneous measurements ongoing, they will all their part to a total load of Qosium Probe. The maximum number of simultaneous controllers per Probe is dependent on capture parameterization and heaviness of the simultaneous measurements vs. the resources available in the platform.

4. Security Considerations

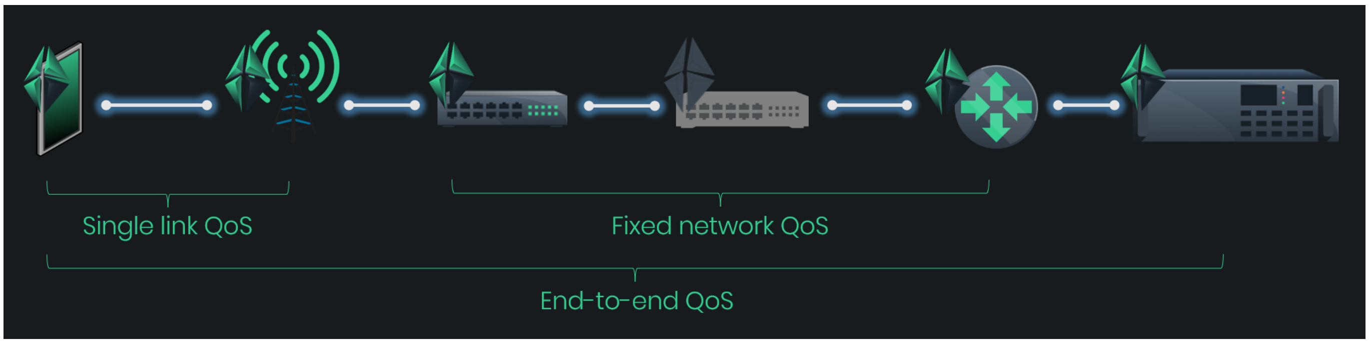
By its nature, Qosium Probe is a remotely controlled measurement unit. The control of Qosium Probe can be limited to a certain IP address space or even only to certain IP addresses see the parameterization. However, this is ineffective if the misuse is committed inside the company’s network. Thus, it is the user’s responsibility to set the control limitations to Qosium Probes, and take care of the security of their network in such a way that no third party can break into the network and potentially misuse Probes installed to the devices in the network.

5. Installing Probe

These instruction comprise Qosium Probe installation to different operating systems.

5.1. Taking Qosium Probe into Use

The locations of Qosium Probes in a network determine the measurement possibilities, that is the measurement paths, as illustrated in the figure below. Qosium Probes are installed to nodes from which measurement results are wanted to be collected. As a passive agent, Qosium Probe needs to see the network traffic being measured. If a network device allows installing new software, Qosium Probe is installed directly to it. In case a device is closed, Probe is suggested to be installed as close as possible to it, or by mirroring traffic to a separate measurement device in the network where Qosium Probe is running.



Qosium Probes come with a registration feature, which helps identify devices available for measurement. It is possible to set a unique identifier for Probes. This is useful when IP addresses of network devices are dynamic. With the unique identifier, a network node can be identified despite the IP address it is currently assigned with.

5.2. System Requirements

5.2.1. Operating System

Most of the Windows and Linux based operating systems are supported. See platform-specific instructions for further information.

5.2.2. Equipment

- The measurement point platform
  - Computer / network device / embedded system, or similar
  - Two devices needed for an end-to-end measurement
- Performance considerations
  - Storage space: < 10 MB
  - Minimum reasonable amount of available system memory: ~ 20 MB
    - Higher system memory is recommended for enabling measurements of higher packet rates
    - Low memory restricts the number of simultaneous measurement controllers per Probe.
  - The processing power of the computer sets the limit to the packet rate of what can be analyzed
- Network interfaces, which are supported by the used packet capturing library

5.2.3. Additional Software

- A packet capturing library
  - Npcap, WinPcap, or Win10Pcap on Windows
  - libpcap on Linux and other systems

5.2.4. External GNSS Receiver

With an external GNSS receiver, accurate time can be got in addition to location. This allows synchronization of the system clocks of the measurement points accurately, enabling high-accuracy delay measurements.

- GNSS receiver device providing PPS output and supporting NMEA or TSIP-protocol. Tested with:

    - Garmin GPS18x LVC
    - Trimble Acutime
- PPS (Pulse Per Second) to Serial interface converter
- Serial port in the PC
    - PCI-based serial port adapters will most likely also work well enough
    - USB-based serial port adapters are not recommended: tests have shown them to cause a milli-second level error to the PPS pulse

The delay measurement is the only one affected by the clock synchronization. Regardless of the synchronization of the clocks, other QoS statistics can be safely measured.

Satellite clock synchronization is currently not supported in Windows, but if this is an interesting feature for you, contact Kaitotek, since this is in our road map.

5.3. Installation on Windows

Qosium has an easy-to-use installer available for Windows. The installation process is very straightforward and is completed typically within a minute.

Qosium's Windows installer contains typically the Qosium *GP* including **Qosium Probe**, **Qosium Scope**, **Qosium Scopemon**, and **Qosium Scope Lite**. A separate installation package only for Qosium Probe can be provided as well if needed.

In some cases, the customer has their own installation routines including software packaging. Thus, if needed, Kaitotek can also deliver Qosium Probe's pure binary files without the installation package.

Follow these steps to install Qosium:

1. Install a packet capture library, if not already installed
2. Sign in to your account
3. Download the installer from your downloads page
4. Launch the installer: an installation wizard guides you through the installation process.

- At the end of the installation, the wizard asks for permission to set up the timing related parameters. If the clock synchronization settings of the environment are already handled, you can ignore this, but if not, it is recommended to allow Qosium to perform the setup.
- If you chose to ignore the timing settings, take a look at, still, the packet capturer timestamping parameterization, since that needs some attention.

After completing these steps, Qosium Probe has been successfully installed to your system.

5.3.1. Installing a Packet Capture Library on Windows

The typical installation package of Qosium does not contain a packet capturing library. If a such library has not been installed in the system earlier, it has to be installed manually. There is also some parameterization to consider.

In Windows, there are at least three supported alternatives for packet capturing, which are discussed in the following sections.

Probe 1.9.2.0

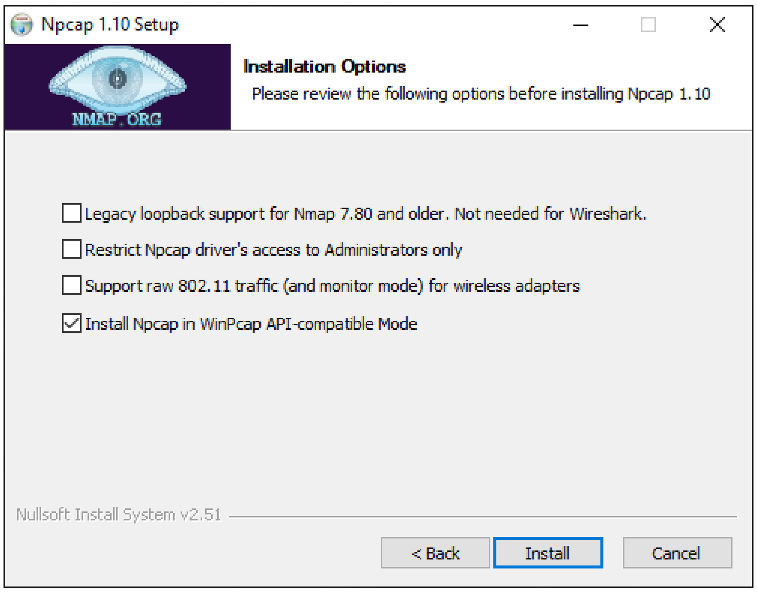
Only Npcap is supported from this version forward.

5.3.1.1. Npcap

Npcap is the recommended packet capturing library to use with Qosium. It is originally based on WinPcap, but it is under continuous development, unlike WinPcap. Npcap uses *NDIS* 6.x, so it should support also those *NIC*'s that WinPcap no longer does. Please bear in mind that in contrast to WinPcap, Npcap's free use is not unlimited. Thus, please check the license conditions of Npcap if you are considering using it.

Download Npcap

If you choose Npcap, the older versions of Qosium (Probe version before 1.9.2.0 and Scope version before 2.8.1.5) require it to be installed in WinPcap-compatible mode. See below what needs to be checked in the installation wizard. The other options shown in the figure are up to the needs – they are neither limited nor required by Qosium.


Npcap 1.10 Setup

Installation Options

Please review the following options before installing Npcap 1.10

- [ ] Legacy loopback support for Nmap 7.80 and older. Not needed for Wireshark.
- [ ] Restrict Npcap driver's access to Administrators only
- [ ] Support raw 802.11 traffic (and monitor mode) for wireless adapters
- [✓] Install Npcap in WinPcap API-compatible Mode

Nullsoft Install System v2.51

If you need to make modifications to the driver's parameters, restart the driver, e.g., from the command prompt (with superuser rights) to enable the changes.

To shut down:

```
net stop npcap
```

To start:

```
net start npcap
```

5.3.1.2. Win10Pcap

Another alternative for a packet capturing driver for Windows is Win10Pcap. Like Npcap, this is also based on WinPcap but has the support for the NDIS 6.x driver model. Win10Pcap works directly with the older versions of Qosium. Win10Pcap is fully free to use since it is under the GPLv2 license. It is, however, likely that the driver is no longer developed: the newest version is from Oct. 8, 2015. In any case, it is still newer than the original WinPcap and seems to work.

One technical downside is that it is not confirmed whether the different timestamp modes are supported or not in Win10Pcap. At least, with tests carried out, we haven't yet been successful in modifying the timestamp mode.

Download Win10Pcap

5.3.1.3. WinPcap

WinPcap (version 4.1.3) is one option for the older versions of Qosium. Despite the fact that it has not been updated for years, and uses the nowadays deprecated NDIS 5.x, it still works with most of the system configurations at least in Windows 10. Thus, if your NIC works with WinPcap, there is no imminent reason to stop using it.

Download WinPcap

5.3.1.4. Parameterization: Timestamping Mode

Npcap and WinPcap support several timestamp modes. With the default mode 0, the absolute accuracy is "a one-shot accuracy", since system performance counters are used for providing high timestamp resolution, but the absolute time is not synchronized. This means that once the driver is turned on, the absolute time is once got from the system clock, but after this, system performance counters will be used for updating the time. Hence, the absolute time begins to drift, and the accuracy of Qosium's delay measurement gets worse as a function of measurement time. However, the resolution is high – even better than in Linux – leading to highly accurate jitter calculation.

If you are interested in one-way delay measurements, it is recommended to change the timestamp mode. One option supported by all known versions of Npcap and WinPcap is 2, in which case, the system clock is used for timestamping packets. This relieves the one-shot accuracy problem but has a downside that it makes the accuracy of jitter calculation worse. This is thanks to the Windows' system clock resolution that is typically only 1 ms. In fact, the default Windows' system clock resolution is as bad as 15.6 ms, but when Qosium is in use, it increases the resolution to 1 ms. For most of the practical applications this 1 ms resolution is already enough, but of course, it is very far from the µs-level resolution that can be reached with timestamp mode 0.

The newer versions of Npcap support also timestamp mode 4. In this, the system clock is used for absolute timing, but system performance counters are used to calculate the accurate time between the sparse clock ticks. If you can use a new Npcap, this is the recommended mode, since it provides highly accurate timestamps without losing the absolute timing synchronization.

If you allow, Qosium's installation wizard detects automatically the installed Pcap version and selects timestamp mode 4 when available and mode 2 otherwise. If not, here are the instructions on how to do it manually. The timestamping mode can be changed in the registry. Prior to this, the packet capturing driver must be stopped.

1. Stop the measurement first (and all SW using Pcap, such as Wireshark in addition to Qosium)
2. Open the command prompt and stop the packet capturing driver as explained earlier in the context of Npcap
3. Open *Registry Editor* and navigate to
   WinPcap: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NPF`
   Npcap (below 0.9985): `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\npf\Parameters`
   Npcap (version 0.9985 and higher):
   `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\npcap\Parameters`
4. Locate the parameter **TimestampMode**. If it does not exist, create it (REG_DWORD).
5. Change/set the mode to the desired value
6. Start the packet capturing driver

Probe 1.9.2.0
From this version forward, manual timestamp mode parameterization is no longer needed. Qosium Probe is

itself capable of selecting always the best available mode.

5.3.2. Clock Synchronization in Windows

Windows has an inbuilt NTP client, so there is no need to install another NTP software. However, the standard settings in the NTP client are only meant for rough system timing. These settings can be modified in order to get better results.

5.3.2.1. Introduction

System clock synchronization in Windows is carried out by its own *NTP* client. In the Services-list it can be found with the name Windows Time and as a driver, it is called *W32Time*. The default parameters of the Windows NTP client are not at their best for accurate measurement purposes. By tweaking the parameters, typical NTP absolute clock accuracy of some milliseconds can be reached. This requires a high-quality network connection with low jitter to the NTP server. If synchronization is carried out over wireless, such as WiFi, the accuracy will be worse.

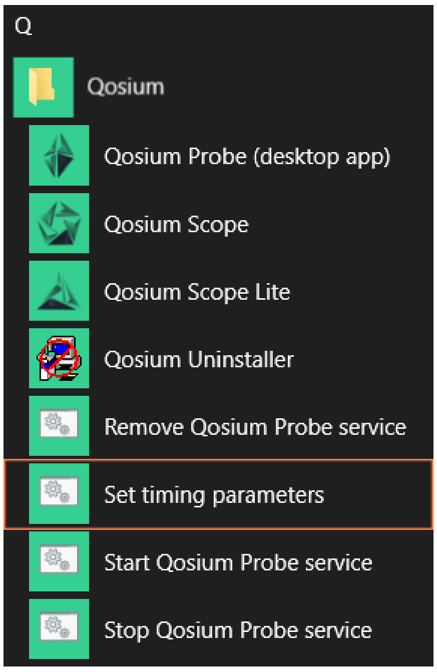
With Qosium's installation package, a set of Windows NTP parameters is also given. By using these, typically a reasonable accuracy for measurement is reached. Those are discussed in the next section.

If the goal is to perform highly accurate delay measurements, other clock synchronization methods are recommended. By using an external *GNSS* clock with Qosium, it is possible to reach about 50 µs absolute timing accuracy for delay measurements with 1 µs jitter resolution (at its best). This, however, requires a custom synchronization driver, which is currently at an experimental state. If you are interested in this, please contact Kaitotek's support.

The W32Time driver in the newer Windows 10 based systems supports also *PTP*. This leads to much better accuracy when compared to NTP. If you are interested in this, contact Kaitotek's support for instructions.

5.3.2.2. NTP Client: Automatic Configuration

Qosium's installation wizard will ask at the end permission to tweak the Windows' NTP client settings. If you did not allow this, it can be done later from the shortcut. Remember, you need to run the script with superuser rights.


Q
Qosium
Qosium Probe (desktop app)
Qosium Scope
Qosium Scope Lite
Qosium Uninstaller
Remove Qosium Probe service
Set timing parameters
Start Qosium Probe service
Stop Qosium Probe service

5.3.2.3. NTP Client: Manual Configuration

Windows NTP client parameters can also be set manually. Qosium's installation package contains a registry file that can be taken into use easily. It is found at the Qosium's installation folder with name `win_timing_conf_<code>.reg`, where the `<code>` part is dependent on the distribution. Before activating the new settings, you can also edit them by a text editor. Alternatively, you can set the values directly with Registry Editor.

To activate the parameters from the file:

1. Open Windows Command Prompt with superuser (system administrator) rights
2. Stop the W32Time service by typing: `net stop w32time`
3. Go to the Qosium's installation folder by using, e.g., File Explorer
4. Double-click the file `win_timing_conf_<code>.reg` and allow it to be entered to Registry
5. At the Command Prompt, start the W32Time service by typing: `net start w32time`

Please note that it might take some minutes before the synchronization takes effect.

⚠ There might also be other timing clients installed to the system. In this case, in order to use the Windows' own NTP client, find the other clients and shut them down.

5.3.2.4. NTP Server Configuration

Windows NTP can also be used in server mode to distribute time to NTP clients. The NTP server is often, however, turned off by default. To check and turn on the NTP server, do the following:

1. Open Windows Registry Editor tool
2. Go to: Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\TimeProviders\NtpServer
3. Check that the value **Enabled** is set to 1. If not, change it.
4. Close Registry Editor

If you made changes, you need to restart the Windows Time service. This can be done in many ways, e.g., by using Command Prompt:

1. Open Windows Command Prompt with superuser rights
2. Stop the W32Time service by typing: `net stop w32time`
3. Start the W32Time service by typing: `net start w32time`

5.4. Installation on Debian-Based Systems

Qosium Probe can be installed by using Debian package management system. This works on Debian, Ubuntu, Raspbian (Raspberry Pi), and other Debian-based Linux distributions. Installation can be done in terminal or by using package manager applications like the Software Center and Synaptic.

5.4.1. Preparations

First, sign in to your account and access your downloads page. Then download Qosium Probe for the target machine. The package is typically named as `QosiumProbe_<version_details>.deb`, where the `<version_details>` part depends on version to another and may contain many identifiers.

5.4.2. Package Pre-checks (Optional)

The Debian package can be verified by opening Terminal and running:

```
dpkg --info QosiumProbe_<version_details>.deb
```

To see what will be installed and where to, run:

```
dpkg --contents QosiumProbe_<version_details>.deb
```

5.4.3. Installation

To install a fresh or upgrade an existing Qosium Probe in a machine, open Terminal and run:

```
sudo dpkg -i  QosiumProbe_<version_details>.deb
```

It is also possible to use *apt* for installation. This method installs automatically all dependencies. Installation using apt-get happens the following way:

```
sudo apt-get install ./QosiumProbe_<version_details>.deb
```

The installation process asks all the relevant details, e.g., whether or not to install Qosium Probe as a Systemd service, etc.

After installation, you can check which version of Qosium Probe is installed by running:

```
dpkg -l qosiumprobe
```

5.4.4. Uninstallation

To remove Qosium Probe from the device, run:

```
sudo dpkg -r qosiumprobe
```

5.4.5. GNSS Setup

In Linux, GNSS positioning is enabled by gpsd and GNSS-based time synchronization is enabled by ntp or chrony. The following guide expects that you have a compatible GNSS receiver.

5.4.5.1. Prerequisites

Using *GNSS* for timing is a two-phased process. First, to get coarse clock synchronization within the correct second, the *full time* is taken from the NMEA messages fed by the GNSS device. Then, the *PPS* signal is used to fine-tune clock synchronization within one second, reaching even microsecond-level accuracy. You could also take the full time from another source, like that provided with *NTP*, but since GNSS also delivers the full time, it is convenient to use that.

To start with, you need a compatible GNSS unit. Any GNSS with PPS output that works in Linux should apply. However, if you did not purchase the GNSS unit from Kaitotek, modifications to the configuration may be needed to match with the wiring of the GNSS unit.

The following packages need to be installed (Ubuntu):

- gpsd
- ntp or chrony
- setserial
- pps-tools (optional, recommended for debugging problems)
- gpsd-clients (optional, recommended for debugging problems)

You need ntp or chrony with PPS clock support. By default, all newer ntp and chrony implementations should support PPS. Bear in mind that ntp and chrony with GNSS/PPS are used just as tools to adjust the clock, not to control the clock with NTP.

This guide assumes that kernel level PPS is supported. If you have a modern distribution this should not be a problem, but for example some old CentOS distributions do not have kernel level PPS support. In this case special *shmpps* application with NMEA(20) and SHM(28) NTP drivers and special configuration might help you to get application level PPS support.

⚠ In order to reach a good result, your machine should have an integrated serial port or at least a PCI-based serial port adapter for the PPS signal. If, e.g., a USB serial port adapter is in use, the synchronization accuracy will be much worse.

5.4.5.2. Step-by-Step Instructions for ntp

This section gives the instructions when you are using ntp. If you are using chrony, see the next section.

This section assumes that the serial port location is */dev/ttyS0* (an integrated serial port for PPS signal) and USB port is */dev/ttyUSB0* (for NMEA message reception). Port numbers may vary depending on your system, so please check and change them accordingly.

To install the required packages, type:

```
sudo apt-get install gpsd gpsd-clients ntp pps-tools setserial
```

Connect the USB and serial cables. It might be useful to check in which port the USB device is attached:

```
dmesg
```

To start *gpsd* type:

```
sudo service gpsd start
```

Then, check that GNSS reception is working:

```
xgps
```

You should see satellites appearing. It might take few minutes initially (as long as 12 minutes in the worst case). Please wait until the fix is got. If nothing happens, exit the xgps and check that gpsd is using the correct USB port by typing:

```
sudo dpkg-reconfigure gpsd
sudo service gpsd restart
```

Sometimes you need to tell gpsd manually the device you intend to use. To do this, modify the configuration file at */etc/default/gpsd*. There, add your device(s) to the device list, e.g.,

```
…
DEVICES=”/dev/ttyUSB0”
…
```

When the GNSS basic reception is working fine, enable the serial driver for PPS (18). In case there are several serial ports, please modify the port number */dev/ttyS<port>* to match the correct one.

```
sudo ldattach 18 /dev/ttyS0
```

/dev/pps0 device should appear. If there are several */dev/pps<port>* devices or you want to ensure that the PPS is working type:

```
sudo ppstest /dev/pps0
```

If you see output appearing every second, the PPS works. Press *Ctrl+C* to stop. If you don’t see anything after `ok, found 1 source(s), now start fetching data...`, please check the device is properly connected and port number is correctly selected.

Enabling the serial port **low_latency** mode can improve jitter:

```
sudo setserial /dev/ttyS0 low_latency
```

Configure NTP by editing */etc/ntp.conf*.

Disable the default servers by commenting them out, if you want to use only GPS time. Example:

```
# Use servers from the NTP Pool Project. Approved by Ubuntu Technical Board
# on 2011-02-08 (LP: #104525). See http://www.pool.ntp.org/join.html for
# more information.
#server 0.ubuntu.pool.ntp.org
#server 1.ubuntu.pool.ntp.org
#server 2.ubuntu.pool.ntp.org
#server 3.ubuntu.pool.ntp.org
```

Add the following lines:

```
#PPS
server 127.127.22.0 minpoll 4 maxpoll 4
fudge 127.127.22.0 flag2 0
#SHM
server 127.127.28.0 prefer
fudge 127.127.28.0 time1 0.55 refid GPS
```

Again, if the */dev/pps<port>* device number is not 0, you should modify the number in PPS lines accordingly. Example (PPS device */dev/pps1*):

```
#PPS
server 127.127.22.1 minpoll 4 maxpoll 4
fudge 127.127.22.1 flag2 0
```

Start the NTP

```
sudo service ntp start
```

5.4.5.2.1. Checking the Status

Wait for the clock to be synchronized. To check the status type:

```
ntpq -p
```

Typical outputs are presented next.

1. The clock is correctly synchronized (see the marks o and *):

   ```
   remote           refid           st t when poll reach   delay   offset  jitter
   ==============================================================================
   oPPS(1)          .PPS.            0 l   11   16  377    0.000   -0.011   0.019
   *SHM(0)          .GPS.            0 l   42   64  377    0.000    6.741  15.645
   ```

2. Not (yet) synchronized. Please wait or recheck the configuration. Also, if the clock is off too much, NTP could refuse synching. In this case, do initial synching manually.

   ```
   remote           refid           st t when poll reach   delay   offset  jitter
   ==============================================================================
   PPS(1)          .PPS.           0 l    -   16    0    0.000    0.000   0.000
   SHM(0)          .GPS.           0 l    -   64    0    0.000    0.000   0.000
   ```

3. SHM only synchronized. Please wait for the PPS sync or recheck the PPS configuration.

   ```
   remote           refid           st t when poll reach   delay   offset  jitter
   ==============================================================================
   PPS(1)          .PPS.           0 l    -   16    0    0.000    0.000   0.000
   *SHM(0)         .GPS.           0 l    3   64    1    0.000  -14.525   0.000
   ```

4. Clock sources dropped as falseticker due to poor accuracy. Please check that the GPS signal is strong enough and restart NTP.

   ```
   remote           refid           st t when poll reach   delay   offset  jitter
   ==============================================================================
   xPPS(1)          .PPS.            0 l    -   16    0    0.000    0.000   0.000
   xSHM(0)          .GPS.            0 l    -   64    0    0.000    0.000   0.000
   ```

5. No PPS clock. Please check that PPS (22) is enabled in *ntp.conf* and that NTP has PPS driver support.

```
remote           refid           st t when poll reach   delay   offset  jitter
==============================================================================
*SHM(0)          .GPS.            0 l    3   64    1    0.000  -14.525   0.000
```

6. NTP not running:

```
ntpq: read: Connection refused
```

5.4.5.3. Step-by-Step Instructions for chrony

This section gives the GNSS positioning setup and timing synchronization instructions when you are using chrony.

In the following instructions, it is assumed that the PPS signal is connected to the device's integrated serial port at */dev/ttyS0*, and NMEA messages are received through a USB-serial converter via */dev/ttyUSB0*. Port numbers may vary depending on your system, so please check and change them accordingly.

If you have ntp installed and you wish to start using chrony, remove ntp first. To install the required packages, type:

```
sudo apt-get install gpsd gpsd-clients chrony pps-tools setserial
```

Just in case, if gpds was already installed, and is running, stop it:

```
sudo systemctl stop gpsd
sudo systemctl stop gpsd.socket
```

Set the NMEA message reception speed correctly according to your settings, e.g., 4800 bauds:

```
sudo stty -F /dev/ttyUSB0 4800
```

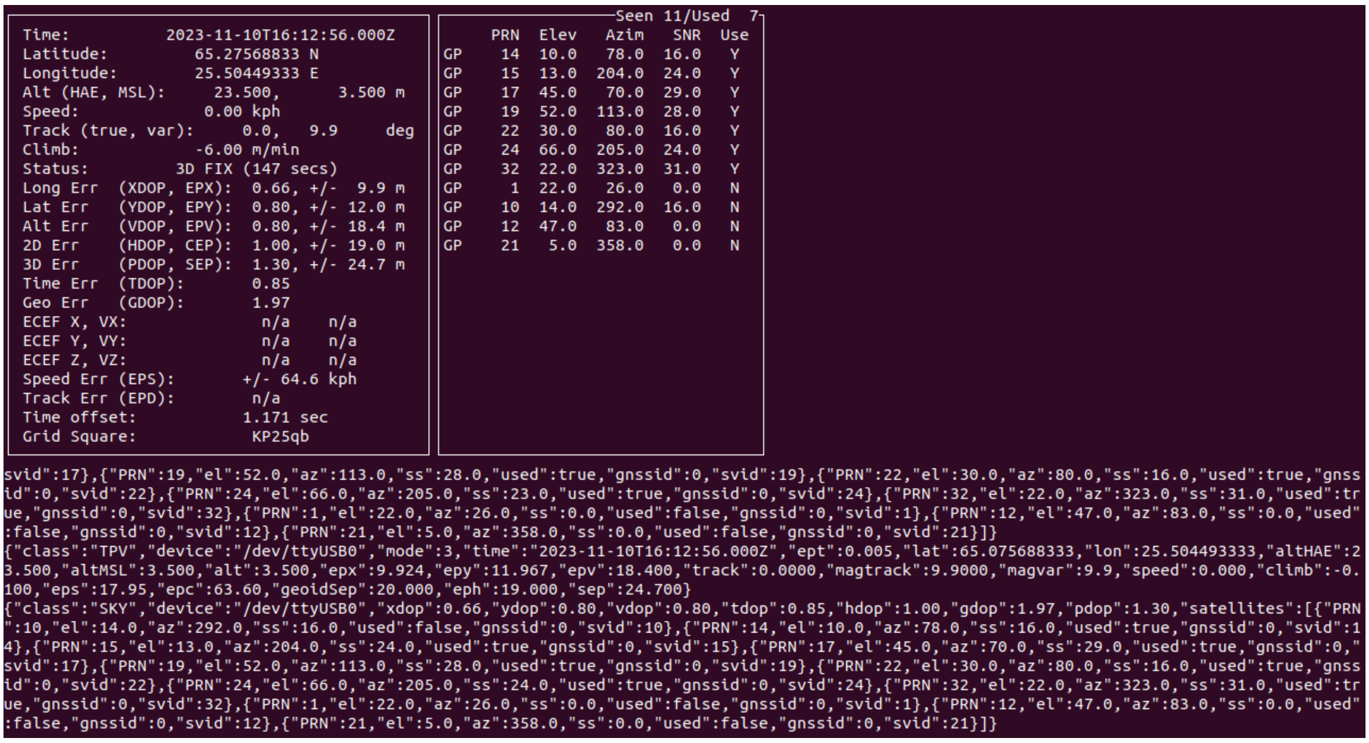
Start gpds manually:

```
sudo gpsd -n /dev/ttyUSB0
```

Now you should be receiving the NMEA message data. Check it:

```
cgps
```

You should see something like the following:



```
                                                        ┌Seen 11/Used  7┐
 Time:          2023-11-10T16:12:56.000Z       PRN  Elev   Azim   SNR  Use
 Latitude:         65.27568833 N          GP    14  10.0   78.0  16.0   Y
 Longitude:        25.50449333 E          GP    15  13.0  204.0  24.0   Y
 Alt (HAE, MSL):     23.500,      3.500 m GP    17  45.0   70.0  29.0   Y
 Speed:              0.00 kph             GP    19  52.0  113.0  28.0   Y
 Track (true, var):     0.0,   9.9     deg GP   22  30.0   80.0  16.0   Y
 Climb:            -6.00 m/min            GP    24  66.0  205.0  24.0   Y
 Status:          3D FIX (147 secs)       GP    32  22.0  323.0  31.0   Y
 Long Err  (XDOP, EPX):  0.66, +/-  9.9 m GP     1  22.0   26.0   0.0   N
 Lat Err   (YDOP, EPY):  0.80, +/- 12.0 m GP    10  14.0  292.0  16.0   N
 Alt Err   (VDOP, EPV):  0.80, +/- 18.4 m GP    12  47.0   83.0   0.0   N
 2D Err    (HDOP, CEP):  1.00, +/- 19.0 m GP    21   5.0  358.0   0.0   N
 3D Err    (PDOP, SEP):  1.30, +/- 24.7 m
 Time Err  (TDOP):       0.85
 Geo Err   (GDOP):       1.97
 ECEF X, VX:              n/a    n/a
 ECEF Y, VY:              n/a    n/a
 ECEF Z, VZ:              n/a    n/a
 Speed Err (EPS):       +/- 64.6 kph
 Track Err (EPD):        n/a
 Time offset:           1.171 sec
 Grid Square:            KP25qb
svid":17},{"PRN":19,"el":52.0,"az":113.0,"ss":28.0,"used":true,"gnssid":0,"svid":19},{"PRN":22,"el":30.0,"az":80.0,"ss":16.0,"used":true,"gnss
id":0,"svid":22},{"PRN":24,"el":66.0,"az":205.0,"ss":23.0,"used":true,"gnssid":0,"svid":24},{"PRN":32,"el":22.0,"az":323.0,"ss":31.0,"used":tr
ue,"gnssid":0,"svid":32},{"PRN":1,"el":22.0,"az":26.0,"ss":0.0,"used":false,"gnssid":0,"svid":1},{"PRN":12,"el":47.0,"az":83.0,"ss":0.0,"used"
:false,"gnssid":0,"svid":12},{"PRN":21,"el":5.0,"az":358.0,"ss":0.0,"used":false,"gnssid":0,"svid":21}]}
{"class":"TPV","device":"/dev/ttyUSB0","mode":3,"time":"2023-11-10T16:12:56.000Z","ept":0.005,"lat":65.075688333,"lon":25.504493333,"altHAE":2
3.500,"altMSL":3.500,"alt":3.500,"epx":9.924,"epy":11.967,"epv":18.400,"track":0.0000,"magtrack":9.9000,"magvar":9.9,"speed":0.000,"climb":-0.
100,"eps":17.95,"epc":63.60,"geoidSep":20.000,"eph":19.000,"sep":24.700}
{"class":"SKY","device":"/dev/ttyUSB0","xdop":0.66,"ydop":0.80,"vdop":0.80,"tdop":0.85,"hdop":1.00,"gdop":1.97,"pdop":1.30,"satellites":[{"PRN
":10,"el":14.0,"az":292.0,"ss":16.0,"used":false,"gnssid":0,"svid":10},{"PRN":14,"el":10.0,"az":78.0,"ss":16.0,"used":true,"gnssid":0,"svid":1
4},{"PRN":15,"el":13.0,"az":204.0,"ss":24.0,"used":true,"gnssid":0,"svid":15},{"PRN":17,"el":45.0,"az":70.0,"ss":29.0,"used":true,"gnssid":0,"
svid":17},{"PRN":19,"el":52.0,"az":113.0,"ss":28.0,"used":true,"gnssid":0,"svid":19},{"PRN":22,"el":30.0,"az":80.0,"ss":16.0,"used":true,"gnss
id":0,"svid":22},{"PRN":24,"el":66.0,"az":205.0,"ss":24.0,"used":true,"gnssid":0,"svid":24},{"PRN":32,"el":22.0,"az":323.0,"ss":31.0,"used":tr
ue,"gnssid":0,"svid":32},{"PRN":1,"el":22.0,"az":26.0,"ss":0.0,"used":false,"gnssid":0,"svid":1},{"PRN":12,"el":47.0,"az":83.0,"ss":0.0,"used"
:false,"gnssid":0,"svid":12},{"PRN":21,"el":5.0,"az":358.0,"ss":0.0,"used":false,"gnssid":0,"svid":21}]}
```

The messages in the lower part of the terminal show the NMEA data. The upper-right box lists the found satellites, and the upper-left box shows the data extracted. If you don't see any data, check the connections and the previous steps.

If you do get data, check that the Status in the upper-left box shows *fix* like in the example figure. If you don't have fix yet, wait for it. It will take some time to synchronize with GNSS (as long as 12 minutes in the worst case during the cold start). After you get fix, you also get the location and the time information shown in the upper part of the box.

Now, you have successfully set up the GNSS positioning for your device. When parameterized, Qosium Probe can read location directly from gpsd. Next, we will configure clock synchronization and start with PPS. Attach a line discipline to the PPS serial port to create a PPS source:

```
sudo ldattach 18 /dev/ttyS0
```

The PPS source will appear as a new device, typically: */dev/pps0*. However, sometimes there might already be PPS devices in the device list. The newly created PPS source will be the last of the PPS devices shown in the device list. To check the device list, type:

```
ls /dev
```

To verify that you are receiving the PPS data, e.g., for */dev/pps0*, type:

```
sudo ppstest /dev/pps0
```

When it works, you see the timestamps for the rising and lower edges of the PPS pulse arriving once per second. If you don't, check the connections and the previous steps.

Enabling the serial port **low_latency** mode can improve jitter:

```
sudo setserial /dev/ttyS0 low_latency
```

Now we are ready to configure chrony. Open the configuration file */etc/chrony/chrony.conf* for editing and add there the following lines:

```
refclock PPS /dev/pps0 lock NMEA
refclock SHM 0 offset 0.5 delay 0.2 refid NMEA noselect
```

The first line ties the reference clock to a PPS source at */dev/pps0* and locks it to another reference lock, called *NMEA*, providing the full time. The second line parameterizes the NTP shared memory driver, *SHM*, capable of receiving timing samples from another processes, like *gpsd* now. The first parameter is the shared memory segment being 0, typically. The rest of the parameters are:

- **offset** (seconds): specifies a correction if there is a delay in the time source. Now, we only need to hit the correct second, so setting 0.5 seconds here decreases the likelihood of locking to an incorrect second.
- **delay** (seconds): sets the NTP delay of the source, relating to the source selection algorithm.
- **refid**: specifies the reference ID of the refclock. We have now set here *NMEA* for convenience, but it is not a reserved name. You can set there any four-ASCII character ID you wish, but remember to refer to the same ID in the PPS reference clock.
- **noselect**: means that this source is never selected as the dominating source for clock synchronization.

Start the chrony service:

```
sudo systemctl start chrony
```

Now, everything should be operational. You can check the situation with instructions:

```
chronyc sources -v
```

When synchronization is working, you should see something like the following:

```
210 Number of sources = 2

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#* PPS0                          0   4   377    17   +589ns[+1251ns] +/-  486ns
#? NMEA                          0   4   377    15    +41ms[  +41ms] +/-  104ms
```

As seen, there is * next to PPS, meaning it is now in use and working. The ? in the NMEA line only means that it is not in use, but as there are samples received, PPS has initially used it to fetch the full time. Otherwise, PPS would not claim to be synchronized. You might also see some NTP servers below, depending on your chrony configuration, but they are not in use either because of our chrony settings. If you do not get sync for PPS after a while, check your cables, settings, and the earlier steps.

5.4.6. Clock Synchronization in Linux

This section assists you in clock synchronization setup on Linux-based systems.

5.4.6.1. Timestamping Accuracy

In Linux, the **timestamp mode** parameter, like in Windows, does not exist. This is not a problem, since the timestamping process in Linux has already good enough resolution. Also, Linux systems have typically *NTP* running with parameters suiting also to measurements, while, of course, it is still likely that the parameters are not optimal, and tuning might be useful. Especially, select as close NTP server (in terms of network delay) as possible. In addition, check that the *minimum and maximum poll values* are small enough. How to tweak the NTP parameters can change from one distribution to another.

Besides NTP, *PTP* is typically also available, enabling better accuracy for the system clock synchronization. In good conditions (a lightly loaded LAN), PTP can yield a microsecond-level accuracy. If you are aiming for microsecond-level accuracy, and fixed connections for PTP are not possible, GNSS clock synchronization is recommended.

5.4.6.2. PTP in Linux

If *GNSS* clock source is not available, PTP is the second best option to synchronize the machines where you run Qosium Probe. In the simplest case, PTP works in a way that there is a Master device and then Slave devices around the network synchronize to that. Thus, if you have a server in the network, which already has an accurate system time, and is accessible, set that as the Master, and let the other machines synchronize into that. If, however, you have only two devices communicating directly, you can select either one of them as the Master.

A good PTP solution is *PTPd*. Setting a Master service with *PTPd* is done as follows:

```
ptpd --masteronly --interface <network interface over which PTP synch messages will be
sent>
```

PTP Slave service is set as:

```
ptpd --slaveonly --interface <network interface over which PTP synch messages will be
sent>
```

In addition, it is useful to use `--verbose` argument to see that Slave finds a Master and also to see the observed time drift. There’s no need to switch the system timesyncd service off. Everything should start working directly.

PTP messages are sent by default as multicast. If you want to use unicast mode instead, use the following arguments in both the Master and the Slave: `--unicast` and `--unicast-destinations`. So, in the case of unicast, you need to tell the Master by parameters who is allowed to connect. If a particular Slave is not on the Master’s list, synchronization will not work.

Sometimes PTP is desired to be tied into a *PTP domain*. This is done with the switch `--domain`. Remember that if the Master uses a certain domain, you need to use the same domain in the Slave.

In addition to command line parameters, PTPd can be configured by using a configuration file.

5.5. Installation on Android

Qosium Probe can be installed on Android devices, such as smartphones and tablets. For Probe to be able to capture traffic, it needs to be run with superuser rights. Thus, you need to root the device or have another way to install software as a system service. This article explains rooting and installation process of Qosium Probe for Android.

5.5.1. Rooting

If you have a way to install Qosium Probe as a system service or you already have superuser rights (root permissions) to the Android device, you can skip this part of the instruction.

The first step in Qosium installation is to acquire superuser permissions in Android device. The so called *rooting* is always device specific. Some manufacturers provide support for unlocking their devices, while others do not. In both cases, it is advisable to search reliable instructions and carefully follow them.

⚠ Kaitotek cannot and will not take any responsibility on the possible damage caused by rooting procedures. However, we have experience on a set of devices and can give recommendations or instructions managing those devices.

⚠ Rooting usually requires unlocking bootloader, which leads to voiding the warranty on the phone! It is advisable to dedicate a specific device for Qosium installation and not to use personal devices.

5.5.2. Installation

Qosium installer comes as an Android APK package. The installation follows the normal process that comes to the packages outside the Google Play store. There are a couple of different methods for performing the installation. The tested methods are using a specific APK Installer application (available at Google Play store) and by using `adb install` command (adb is part of the Android SDK).

The following instructions make use of the APK Installer application:

1. Download the Qosium Probe’s installation package (typically named as `QosiumProbe-<version_information>.apk` to the Android device for example, from a memory card, or with a browser from you Kaitotek account
2. Open APK Installer and install the just downloaded package
3. Locate **QosiumProbe** application from the application menu and start it
4. If previously installed Qosium Probe agent process is running on background, close it by tapping the **Stop** button
5. If Qosium Probe agent was previously installed remove it by tapping **Uninstall** button
6. Execute the Qosium Probe agent installation procedure by tapping **Install** button

5.6. Installation on Red Hat Based Operating Systems

Qosium Probe installation package is provided as a tar.gz package and the installation happens in command line interface. The same installation method works on CentOS, Red Hat, and other operating systems that are similar to Red Hat.

5.6.1. Preparations

First, sign in to your account and access your downloads page. Then download Qosium products for the target machine. The package is typically named as `QosiumProbe_<version_details>.tar.gz`, where the `<version_details>` part depends on version to another and may contain many identifiers.

5.6.2. Installation

First, extract the tar.gz package by opening a *Terminal*:

```
tar -zxvf QosiumProbe_<version_details>.tar.gz
```

Then, go into the extracted directory by typing:

```
cd QosiumProbe
```

Run the install script with superuser rights:

```
./install-QosiumProbe.sh
```

The installation procedure asks whether you want to set Qosium Probe as a Systemd service or to be started manually.

The installed Qosium Probe files can be found at `/opt/QosiumProbe/`.

5.6.3. Uninstall

The uninstall script is located in the same extracted folder where the *install-QosiumProbe.sh* was. To uninstall Qosium Probe, run the following in Terminal with superuser privileges:

```
uninstall-QosiumProbe.sh
```

5.7. OpenWRT

Qosium Probe installation package is provided as an ipk package to be managed with the opkg package manager.

5.7.1. Preparations

First, sign in to your account and access your downloads page. Then download Qosium products for the target machine. The package is typically named as `QosiumProbe_<version_details>.ipk`, where the `<version_details>` part includes the Qosium Probe version and other delivery-specific identifiers.

5.7.2. Installation

Installation happens through the *opkg* package manager. Transfer the install file to the device and run:

```
opkg install QosiumProbe_<version_details>.ipk
```

The installation procedure asks whether you want to set Qosium Probe as a background service (creates a *procd* init script to `/etc/init.d/`) or to be started manually. You can also set a unique *Service ID* for the Probe.

The installed Qosium Probe files can be found at `/opt/QosiumProbe/`.

5.7.2.1. Install from tarball

If you wish to get the installer as a tarball file, the installation happens in the following way:

First, move the tarball file to the device and extract the compressed tar file:

```
tar -zxvf QosiumProbe_<version_details>.tar.gz
```

Then, go into the extracted directory by typing:

```
cd QosiumProbe
```

Run the install script with superuser privileges:

```
./install-QosiumProbe.sh
```

5.7.3. Uninstall

Uninstallation happens with *opkg*:

```
opkg remove QosiumProbe
```

5.7.3.1. Uninstall with tarball installation

If you installed Probe from a tar.gz file, the uninstall script is located in the same extracted folder where the *install-QosiumProbe.sh* was. To uninstall Qosium Probe, run the following in command line with superuser privileges:

```
uninstall-QosiumProbe.sh
```

5.8. Installation on Other Operating Systems

Qosium Probe can be installed to a large number of different operating systems. Below you can see instruction for some of them. Kaitotek will provide you with more instructions regarding these operating systems upon delivery.

5.8.1. MacOS

Qosium Probe installer is provided as a DMG, mountable disk image, file. Installation happens over a graphical window by drag and dropping Qosium Probe to *Applications* directory.

5.8.2. Equipment of Axis Communications©

Qosium Probe is provided as an official Axis software installation package. The installation is carried out by using the standard software installation mechanisms of Axis.

5.8.3. Other Linux Systems & FreeBSD

Qosium Probe is provided as a *tar.gz* package, typically named as `QosiumProbe_<version_details>.tar.gz`, where the `<version_details>` part depends on version to another and may contain many identifiers.

First, extract the tar.gz package in a command line interface:

```
tar -zxvf QosiumProbe_<version_details>.tar.gz
```

Then, go into the extracted directory by typing:

```
cd QosiumProbe
```

Run the install script as superuser:

```
./install-QosiumProbe.sh
```

The Qosium Probe files can then be found from `/opt/QosiumProbe/`.

6. Parameterizing Probe

A Qosium measurement is parameterized by using a measurement controller, such as Qosium Scope. In most of the cases, you just install Qosium Probe as a system service once and start measuring without any modifications to the Probe itself. However, Qosium Probe has also operational parameters that in some measurement scenarios need attention. This article describes the available parameters and how to set them to the Probe.

6.1. Editing Probe’s Parameters

The parameters of Qosium Probe can be edited via `QosiumProbe.ini`, which is located in the installation directory of Probe. The file follows format

```
parameter_name parameter_value
```

i.e., a single space between the parameter name and its value.

These are the operating parameters of Probe and cannot be set by a measurement controller. Qosium Probe reads the parameters at startup. Thus, if you make changes to the parameters, run the Probe down, make the changes, and then run the Probe up.

Most of the parameters are already set to appropriate default values, which work well in most cases

6.2. Setting Up the Server

Qosium Probe shows outside as a server that accepts incoming QMCP connections. The address of the server can be defined with the parameter **server_address**. Server port (TCP) can be defined with the parameter **server_port**. By default, connections via all local IP addresses are accepted to port 8177.

6.2.1. server_address

Defines the IP address to which Probe is bound to.

- Values:
  - `0` All addresses – Sets the general address 0.0.0.0 which accepts connections from all the local interfaces.
  - `1` Local address – Sets the local loopback address 127.0.0.1.

  - `<IP address>` – Interface IP address
    - Set the specific (interface) IP address to which the Probe is bound to.
    - Format: the standard dotted format (e.g., 192.168.0.109).
- Default: `0`

To bind the Probe only to a certain interface, use its IP address.

6.2.2. server_port

Defines the port (TCP only, currently) of Qosium Probe to which the server is bound to.

- Values:
  - `0` Port 8177 is used
  - `<TCP port>` – Your specified port at range `1 - 65534`
- Default: `0`

If the Probe is behind a firewall and you wish to have access from outside, you have to open this port to the firewall.

If the selected port is already booked, the Probe cannot be bound there.

6.2.3. service_id

Defines one part of the Qosium's measurement fingerprint. This parameter provides a way to uniquely identify a Probe. The use of this parameter is voluntary and up to the user. You can specify here a number based on your own needs and rules. The set parameter value will appear in the Average Results and can be used, e.g., to classify results in Qosium Storage.

This parameter has also a very important role in *Central server registration* functionality explained later.

- Values: `0` - `4294967294`
- Default: `0`

6.3. Limiting Access to Probe

If you wish to limit who can control Qosium, it can be done from an IP address basis. You can select a network or even individual IP addresses from which the Qosium Probe accepts control. A typical example is that Qosium's control is limited inside the corporate network. By default, there are no limitations. There are two parameters in *QosiumProbe.ini* that allow controlling the access and they are explained next.

6.3.1. secu_ctrl_addr_mode

This parameter turns the control limiting feature on/off and determines the way how the control addressing is defined.

- Values:
  - `0` Function is off – Control is not limited
  - `1` Address mode – Each allowed controller address is given separately (the next parameter)

  - `2` Masking mode – An address mask will be used instead of separate addresses (the next parameter)
- Default: `0`

6.3.2. secu_ctrl_addresses

This parameter defines either the network or individual addresses that are allowed to control the Probe based on the mode (**secu_ctrl_addr_mode**).

- Values:
  - If **secu_ctrl_addr_mode** = 1
    - Feed the addressess one by one separated by ;.
    - Example: `192.168.0.101;192.168.1.3;192.169.1.78`
  - If **secu_ctrl_addr_mode** = 2
    - Feed the allowed network in format: `<network>;<mask>`
    - Example: `192.168.0.0;255.255.255.0`

6.4. Positioning

Positioning support is an integral part of Qosium, but the direct support for GNSS receivers is an optional feature, and not all the builds support it. The log at Probe's startup shows the supported features listing also whether GNSS is supported or not. Regardless of GNSS support location can be still fed to Qosium Probe via UDP messages or manually by measurement controllers.

6.4.1. location_mode

This parameter is used to turn on/off the automatic location information collection, defining also the way how the collection is carried out.

- Values:
  - `0` Off – No automatic location information collection
  - `1` GNSS with TSIP
    - Meant for Windows based systems only with a special time-stamping driver.
    - Position is acquired from TSIP messages directly via serial port interface.
    - This mode requires a Probe built with GNSS support.
  - `2` GNSS with NMEA
    - Meant for Windows based systems only with a special time-stamping driver.
    - Position is acquired from NMEA messages directly via serial port interface.
    - This mode requires a Probe built with GNSS support.
  - `3` GPSD API
    - This is meant for Linux based system only.
    - Position is acquired from Linux's GNSS daemon (gpsd).
    - This mode requires a Probe built with GNSS support.
  - `4` NMEA data from UDP
    - Acquires position via NMEA messages that are carried over UDP.
    - When this is set, the parameter **location_nmea_udp_port** defines the port which is used to listen the

UDP NMEA messages.

- Default: `0`

Manually positioning Probe is still possible via Qosium Scope, regardless of the selected location mode.

6.4.2. location_nmea_udp_port

When parameter **location_mode** is set to use *NMEA data from UDP*, this parameter will define the reception port for the UDP messages carrying the NMEA data. When using this, remember to activate at least the GGA message transmission from your external GNSS receiver.

- Values:
  - `1 - 65534`
- Default: `7777`

6.5. Physical Layer Information Collection Parameters

Qosium is able to collect some physical layer information statistics directly from the network device. In practice, this is about radio interface statistics, such as signal strength and SINR. This information is attached as a part of the measurement results, being valuable especially during field measurements.

There are many ways to gather the physical layer information. In the case, you are measuring a WLAN interface, the information is collected automatically. The typical statistics got are RSSI and Base Station MAC address. Some WLAN interfaces might let you also get SINR. The WLAN statistics are currently available only for Linux-based systems. Please note that only the WLAN client end will yield the statistics.

Another method to gather these statistics is to use AT commands. This method is typically used with cellular modems. There are two requirements: Qosium Probe needs to run on the same device with the wireless modem, and the modem needs to support AT commands. Once gathering is successful, the collected information acts as a property of Qosium Probe similar to location and collection takes place in the background. Thus, all measurements using the particular Probe will get the same information. If your modem supports AT commands, but it is not supported by Qosium, please ask us how to proceed. We include support to new AT commands from time to time based on customer need.

The third method to gather physical layer information is to feed it to Qosium Probe from an external source. For example, if you have a modem that is running elsewhere or, e.g., does not support AT commands, you can (or ask us to) implement an external statistic collection functionality and feed the collected information to Qosium Probe. The information feeding is carried out using a specific QMCP message over UDP. If you are interested in this, please ask us for details.

If you have multiple cellular modems in a device, set up multiple Qosium Probes, each per modem, to collect their radio level statistics.

6.5.1. phy_collection_mode

This parameter turns the physical layers information collection on/off and sets the collection mode.

- Values:
  - `0` Off – Physical layer information collection is not active
  - `1` – AT commands are used (Linux only!)

    - `2` – External source (QMCP over UDP) is used
- Default: `0`

6.5.2. phy_ext_udp_reception_port

This parameter is meaningful when phy_collection_mode is set to *External source*. In this case, the parameter defines the TCP port where to bind the QMCP external PHY information reception server. Thus, use this port to feed the results to Probe.

- Values: Your specified port at range `1 - 65534`
- Default: `8181`

6.5.3. phy_at_dev_address

Defines the modem address to where the physical layers information collection AT commands will connect.

- Values:
    - `<Device>` Address – Given in string format, example: `"/dev/ttyUSB2"`

6.5.4. phy_at_signal_cmd

When physical layer information fetching by AT commands is activated, this parameter defines the command to get the signal information. In practice, the modem is fed with command `AT+' to which it replies in a certain way.

- Values:
    - `CSQ` – A generally used command to gather signal strength (RSSI) supported by many devices
    - `QCSQ`
        - Yields a wider set of signal related information than the previous command (e.g., RSSI, RSRP, RSRQ, SINR).
        - This is supported at least by some Quectel LTE-modules.
    - `QENG`
        - This is supported at least by RG50xQ&RM5xxQ Series Quectel modules to fetch extended signal information like QCSQ.
        - This returns also network information at the same time.
- Default: `CSQ`

6.5.5. phy_at_cellid _cmd

This parameter is very similar to the previous, but instead of signal information, cell information is collected.

- Value: `CREG`
    - A generally used command to gather cell information of a cellular network
    - Typically returns the active Cell ID

Kaitotek can easily add more commands for different devices. Thus, if you have something in mind, contact Kaitotek's support.

6.6. Central Server Registration

In large Qosium setups, it is feasible to use a central server to collect information on the available measurement points in the network. A unique identifier can be defined for each device being a big help, e.g., when dynamic IP addressing is used. This enables you to detect at a quick glance which devices are in the network and what is their current IP address. A missing device in the list can be an indication of a network problem. The list of registered Probes can be fetched by, e.g., Qosium Scope.

Any Qosium Probe can act as such a central server. Thus, just put Qosium Probe running to the desired location that you wish to act as the Central server. Then parameterize the other Probe's in the network as follows.

When using Central server registration, the parameter **service_id** has an important role. It is the one that is used to identify a Probe in the registration. A good way of operating is to have a separate list of the given IDs, i.e., which Service ID maps to what measurement point in the network.

⚠ Give each measurement point a unique Service ID. Currently, Qosium does not take into account the Service ID conflict!

6.6.1. central_server_address

Defines the Central server address where the registration is to be made. Thus, give here the IP address of your selected Central server.

- Values: `<IP address>` – Given in the standard dotted format (e.g., `192.168.0.109`).
- Default: `0`

6.6.2. central_server_port

Defines the Central server port where the registration is to be made.

- Values:
  - `0` – Port 8177 is used
  - `<TCP port>` – (e.g., `8901`)
- Default: `0`

6.6.3. reg_lifetime

This parameter is used, first, to turn on/off the registration feature, and second, to set the registration lifetime. It has a direct relation to the registration interval since it is carried out periodically.

- Values:
  - `0` – Registration is off
  - `1 - 4294967294`
- Default: `0`
- Unit: `seconds`

ⓘ Set this value large enough to avoid excess control traffic but small enough to remain the reactivity of the registration process. A good compromise is typically a couple of minutes.

6.7. Packet Capturing

The Pcap-based packet capturing libraries operate as drivers at the kernel level. When packet capturing is activated from an application, such as Qosium, Pcap reserves a fixed amount of memory as a capture buffer regardless of the number of captured packets. This fixed allocation is made separately for each independent simultaneous capturing process. Thus, in the case of Qosium, each separate measurement controller, when connected and measuring in a Probe, causes this same reservation. Qosium Probe's parameter **pcap_cap_buff_size sets** the size of this allocation.

Another parameter is also related to this: **max_pcap_mem_reservation**. It defines how much memory, as total per Probe, can be reserved for the Pcap processes. Thus, this parameter, together with **pcap_cap_buff_size**, determines how many clients a Probe can serve simultaneously. The purpose of **max_pcap_mem_reservation** is to avoid Qosium of consuming too much memory in the device. After all, a measurement should never hinder the operation of the measurement target.

Setting **pcap_cap_buff_size** is not always straightforward. If it is too small, Pcap starts to drop packets when measuring high-rate traffic stream. The Pcap drops will cause false packet loss and *negative packet loss* (the statistic *Sent info not found*). Luckily, the Pcap drops themselves are also collected as statistics, so detecting this situation is easy. If, on the other hand, **pcap_cap_buff_size** is set to an unnecessarily large value, the device's memory capacity gets wasted for nothing. Also, a large **pcap_cap_buff_size** will decrease the maximum number of allowed simultaneous clients per Probe.

There are certain rules of thumb on how to set the **pcap_cap_buff_size**. If the measured traffic load is sufficiently small (< 10 Mbit/s), setting 10 MB as the reserved buffer size should be enough. When measuring very small traffic loads (« 1 Mbit/s), even 1 MB could be enough. When measuring high traffic loads (> 100 Mbit/s), it is recommended to set 100 MB or more. However, even these coarse rules do not hold in all the cases because the joint performance of packet capturing and Qosium's internal processes is dependent on the processing power of the platform. If the platform is a typical desktop PC, a business laptop, or a powerful smartphone, then the said rules should hold quite nicely. If, however, the platform is, e.g., a resource-limited IoT device, paradoxically, you need a larger Pcap buffer. Then again, when Qosium is run in a high-power server platform or in a state-of-the-art desktop PC, less Pcap buffer is required.

Another dimension is how to set the **max_pcap_mem_reservation** parameter. In this, first, consider how many clients you wish to serve with the Probe. Second, examine how much free memory there is in the device where the Probe is installed. If the Probe is running in a powerful PC with plenty of memory, this is typically not a problem, but just set, e.g., 4 GB of memory to this, being well enough for many typical measurement cases. In performance-limited devices, consider free memory as the memory that is free while the device is operating its normal routines. For example, if the device where you run the Probe is a network video camera, let the camera stream its content with maximum resolution and then observe how much memory is free in the device. Based on this information, you can set **max_pcap_mem_reservation**. Remember also that all Qosium measurement threads themselves will consume some memory. A typical measurement thread measuring medium rate traffic takes about 10 MB of memory. For example, by setting **pcap_cap_buff_size** to 40 MB and **max_pcap_mem_reservation** to 400 MB, you can fit 10 clients in Qosium Probe. When running all 10 clients, they will take that 400 MB of memory in the Pcap driver level, but the Qosium measurement will take about 500 MB in total. Hence, **max_pcap_mem_reservation** should be set according to the free memory with some clearance.

In conclusion, when operating Qosium Probe in a high-power device, setting the two parameters is quite easy. Then, when operating in performance-limited devices, setting the parameters will be balancing with the resources, needs, and capabilities. In these cases, it is typical that experiments are needed to set the **pcap_cap_buff_size** and **max_pcap_mem_reservation** optimally for the needs.

Yet another parameter related to packet capturing is **capture_snap_len**. It sets the maximum length for the captured packets. When this parameter value is smaller than the length of the packet to be captured, the packet will be cut to this size before it is delivered from Pcap to Qosium. Qosium, in its normal operation, only glances inside the packet content, so no notable performance benefits are got by cutting the packets. Instead, if a packet is cut too short, it might have a negative effect on the QoS calculation process. In addition, when taking full Pcap captures, the cut bytes will be missing in the output capture file. Therefore, if you have no special needs to cut the packets, leave them as they are.

⚠ If the parameter max_pcap_mem_reservation is set carelessly, Qosium can, in the worst case, jam the whole device!

This can happen if **max_pcap_mem_reservation** is set higher than the maximum amount of memory in the device and too many users measure simultaneously in a Qosium Probe. The total memory reservation of the measurement process will drain all the available memory, potentially halting the device.

6.7.1. pcap_cap_buff_size

This parameter defines the size of the packet capturing buffer at Pcap. Its role in defining the success and efficiency of the packet capturing process is very important.

- Value range: `1 - 2147`
- Default: platform dependent
- Unit: `MB`

6.7.2. max_pcap_mem_reservation

This parameter sets the maximum joint Pcap buffer memory allocation of all the measurements. Together with the parameter pcap_cap_buff_size, the maximum number of simultaneous measurements is defined.

- Value range: `1 - 4294967294`
- Default: platform dependent
- Unit: `MB`

6.7.3. capture_snap_len

This parameter is used to limit the capture length of packets.

- Values:
  - `0` Off – The packets are not cut
  - `1 - 4294967294` – Your specified maximum length
- Default: `0`
- Unit: `B`

6.8. Other Parameters

This category contains some additional parameters that might be sometimes useful.

6.8.1. hide_console

If Qosium Probe is not intended to be run as a service, but still needs to be run without a visible log window, this is the parameter to use. This parameter is active only in Windows-based platforms.

- Values:

  - `0` Off – The log window is visible
  - `1` On – Qosium Probe is run at the background without a visible window
- Default: `0`

6.8.2. force_to_single_cpu

Qosium Probe is a multi-thread software. Every controller connection is run in a separate thread each constituting multiple threads. Thus, a multi-core platform is a natural environment for running Qosium Probe. Sometimes, however, there might be a need to limit Qosium Probe's computational resource usage. This parameter can be used to limit Qosium Probe to operate only in a single CPU / processor core.

- Values:
  - `0` Off – The CPU / core affinity is free
  - `1` On – Qosium Probe is forced to operate on a single CPU / core (the first one)
- Default: `0`

7. Using Qosium Probe

Qosium Probe is not directly operated as such since Qosium measurements are controlled fully by a measurement controller. However, there are certain operative functionalities, like license activation and launching, that are discussed in this section.

7.1. License and Activation

Qosium Probe needs a license to run. The license information is locally stored in a license file located in Probe's installation folder. The license can be pre-activated, like, for example, in the academic license. In this case, Qosium usage can be started right after the installation.

In installation-limited licenses, each Qosium Probe needs to be activated online. The activation is carried out simply by using Kaitotek's License server as follows:

- Launch the Probe if not already running
- Verify that the device where Qosium Probe is installed has Internet access
- Open a measurement controller, like Qosium Scope, and connect Qosium Probe
- Qosium Probe will now activate itself at Kaitotek's License server.

A successful activation is typically fast. There appears some information of the successful activation while the controller just connects the Probe as usual. If something goes wrong, error information is delivered via the controller. An activated Qosium Probe will be fixed to the device. However, the installation can also be moved to another device.

ⓘ If your license requires online activation, but it is impossible to organize an Internet connection to the device, contact Kaitotek's support. There are also other alternatives for the activation.

7.2. Launching Probe

Once Qosium Probe has been parameterized, it is time to launch it. Typically, Qosium Probe is set to run as a system service. In this way, it always starts with the platform, being ready for measurements. This also has the advantage that running Qosium measurements does not require superuser rights in the platform.

Qosium Probe can also be run as a normal application. This can be useful in some special cases since Probe's log window will be visible. superuser rights, however, are needed when launching the Qosium Probe application.

The way of launching Qosium Probe is dependent on the platform. These are discussed in the following subsections:

- Windows based systems
- Linux based systems
- Android based systems

Only a single Probe is required per measurement point.

Did you try to launch Qosium Probe as an application, but it failed because of a reserved port? If so, check that is Qosium Probe already running as a system service.

7.3. Installation Transfer

If you have an installation-limited license, it is possible to transfer an activated Qosium Probe to another device. In practice, Qosium Probes can always be installed without limits, but in installation-limited license models, only a certain number of them can be in use. Thus, transferring Qosium Probe from one device to another is, in practice, transferring the license.

License transfer is carried out as follows:

- Verify that both of the devices have an Internet connection
- Go to the device from where the Qosium Probe is to be removed
- Open command line (with administrative rights in Windows)
- Start Qosium Probe with a special argument:
  - Windows: `QosiumProbe licunreg`
  - Linux: `sudo <path>/QosiumProbe licunreg`
- The given command will try to release the activated license. Check the printed log that did it succeed. If so, the license has been disactivated and the installation-right returned to the Kaitotek's License server.
- Now go the device where the installation will be transferred to
- Install Qosium Probe there, if not already installed.
- Activate the Qosium Probe installation as explained earlier

7.4. Log

Qosium Probe writes a high-level log to standard output when it is running. Typically, there is no need to observe this log. In some rare cases, e.g., when troubleshooting license issues, you may need to check the log. When you launch Qosium Probe directly as an application from the command line, the log will be printed there to be observed directly. If Qosium Probe runs as a system service, you need to access the system log.

For example, if you are running Qosium Probe as a system service in Ubuntu, and you wish to see the last 50 lines of Qosium Probe's log, it is done in terminal as follows:

```
sudo systemctl status -n 50 QosiumProbe
```

7.5. Launching Probe in Windows

Qosium Probe is typically run as a Windows service. However, it can also be run as a normal desktop application. This section describes how to run Probe on Windows, and what options are available.

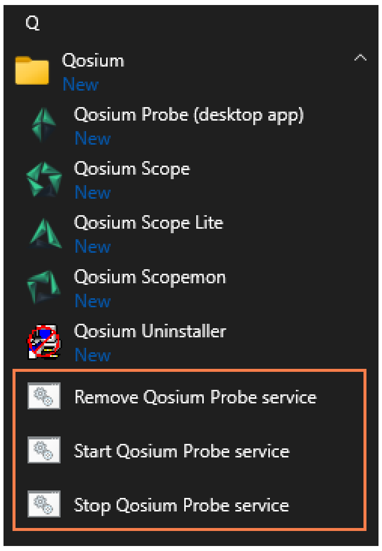
7.5.1. Introduction

Qosium Probe can be launched either as an application or service in Windows. Using shortcuts created by Qosium installer is the most convenient choice. Nonetheless, Probe can be also launched directly from the installation folder.

Launching Probe requires administrative rights on Windows

7.5.2. Launching as Windows Service

Qosium Probe is typically run as a *Windows service*. This is convenient since Probe will then always start with the operating system. Starting and stopping the service is also possible via the shortcuts:


Q
Qosium
New
Qosium Probe (desktop app)
New
Qosium Scope
New
Qosium Scope Lite
New
Qosium Scopemon
New
Qosium Uninstaller
New
Remove Qosium Probe service
Start Qosium Probe service
Stop Qosium Probe service

The shortcuts are as follows:

- **Start Qosium Probe service** - Install Qosium Probe as a Windows service and start it
- **Stop Qosium Probe service** - Stop Qosium Probe service (if active)
- **Remove Qosium Probe service** - Uninstall Qosium Probe service (if installed)

The same can also be configured directly from the command line. To do this, start *Command Prompt*, change directory to Qosium installation folder, and execute **QosiumProbe.exe** with arguments, as demonstrated below.

Install Qosium Probe as a Windows service:

```
QosiumProbe.exe install
```

Start Qosium Probe service:

```
QosiumProbe.exe start
```

Stop Qosium Probe service:

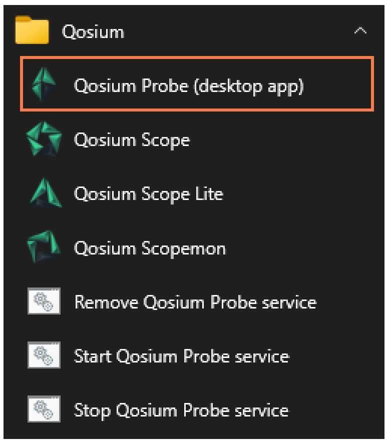
```
QosiumProbe.exe stop
```

Uninstall Qosium Probe service:

```
QosiumProbe.exe uninstall
```

7.5.3. Launching as Windows Application

In addition to running as a service, Qosium Probe can be launched the same way as any other Windows application. If an installation package was used to install the software, access Windows start menu and use the shortcut **Qosium Probe (desktop app)**.


Qosium
Qosium Probe (desktop app)
Qosium Scope
Qosium Scope Lite
Qosium Scopemon
Remove Qosium Probe service
Start Qosium Probe service
Stop Qosium Probe service

If the shortcut is not available, locate the installation folder and simply execute **QosiumProbe.exe**. That can be done either in the *File Explorer* or *Command Prompt*.

7.6. Launching Probe on Linux

This section goes through starting of Qosium Probe on Linux based systems. The launching of Qosium Probe depends on whether it is installed as a system service or as to be started manually.

7.6.1. Manual Launching

In command line (Terminal), run the following command (use *sudo* in front of the command if you are not a superuser):

```
/opt/QosiumProbe/bin/QosiumProbe
```

Probe can be set in the background by pressing *Ctrl-Z* and typing `bg`.

7.6.2. Qosium Probe Installed as Service

7.6.2.1. Systemd Service (Ubuntu, Debian, Raspbian, CentOS, Red Hat, etc.)

On Debian-based and Red Hat based operating systems, Qosium Probe can be installed as a Systemd service. After that, controlling of Qosium Probe happens with *systemctl* commands. Qosium Probe is launched automatically upon installation and device startup. Omit *sudo* from the commands below if you are superuser and sudo is not supported by operating system.

Start:

```
sudo systemctl start QosiumProbe
```

Stop:

```
sudo systemctl stop QosiumProbe
```

Request status:

```
sudo systemctl status QosiumProbe
```

Disable Qosium Probe from starting automatically upon reboot:

```
sudo systemctl disable QosiumProbe
```

Enable Qosium Probe to start automatically upon reboot:

```
sudo systemctl enable QosiumProbe
```

7.6.2.2. Procd Init Scripts (OpenWRT)

On OpenWRT-based systems, Qosium Probe can be installed to start to background automatically. Qosium Probe is launched automatically upon installation and device startup. Run the following example commands with superuser privileges.

Start:

```
/etc/init.d/qosium_probe start
```

Stop:

```
/etc/init.d/qosium_probe stop
```

Disable Qosium Probe from starting automatically upon reboot:

```
/etc/init.d/qosium_probe disable
```

Enable Qosium Probe to start automatically upon reboot:

```
/etc/init.d/qosium_probe enable
```

7.7. Launching Probe in Android

Qosium Probe agent is started with the same Qosium Probe Android Launcher that was used to install it.

1. Locate the Qosium Probe application from application menu and start it.
2. Start the Qosium Probe agent by tapping the *Start* button.
3. Optionally, you can view the output of Qosium Probe agent by selecting the *Show Log* option.

An easy way of engineering the connection between Android Qosium Probe and other Qosium components, is to use the Mobile Wi-Fi hotspot feature of Android. After enabling Mobile hotspot, the Qosium measurement controllers can attach to the Wi-Fi exposed by the Android device and connect to the Qosium Probe through it.

8. Glossary

General Package

Qosium General Package contains the basic Qosium components required for measurements and monitoring.

Network Driver Interface Specification

NDIS is the API for the network interface cards used by Windows-based systems.

Wikipedia article on Network Driver Interface Specification

Network Interface Card

A piece of hardware which offers a device a networking interface.

Network Time Protocol

A very common protocol for synchronizing the clocks of devices across a network.

Global Navigation Satellite System

A general term under which all the different global satellite navigation systems (e.g., GPS, GLONASS, Galileo, BDS) fall.

Precision Time Protocol

A protocol for synchronizing the clocks of devices across a network. The reached synchronization accuracy is typically considerably better than with NTP.

Pulse per second

A square-like electrical signal that is used in accurate clock synchronization