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.

Table of Contents

1. Editing Probe's Parameters 3
2. Setting Up the Server 3
  2.1. server_address 3
  2.2. server_port 3
  2.3. service_id 4
3. Limiting Access to Probe 4
  3.1. secu_ctrl_addr_mode 4
  3.2. secu_ctrl_addresses 4
4. Positioning 5
  4.1. location_mode 5
  4.2. location_nmea_udp_port 5
5. Physical Layer Information Collection Parameters 6
  5.1. phy_collection_mode 6
  5.2. phy_ext_udp_reception_port 6
  5.3. phy_at_dev_address 6
  5.4. phy_at_signal_cmd 7
  5.5. phy_at_cellid _cmd 7
6. Central Server Registration 7
  6.1. central_server_address 8
  6.2. central_server_port 8
  6.3. reg_lifetime 8
7. Packet Capturing 8
  7.1. pcap_cap_buff_size 10
  7.2. max_pcap_mem_reservation 10
  7.3. capture_snap_len 10
8. Other Parameters 10
  8.1. hide_console 10
  8.2. force_to_single_cpu 10

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

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.

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.

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.

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`

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.

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`

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`

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.

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.

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`

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.

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`

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`

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"`

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`

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. 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!

6.1. 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`

6.2. 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`

6.3. 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.

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.

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`

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`

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`

8. Other Parameters

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

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`

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`