# Qosium Scopemon

*Qosium Scopemon is the key to continuous and automated monitoring of critical applications and services. This sibling of Qosium Scope is designed for automatic monitoring instead of targeted measurements.*
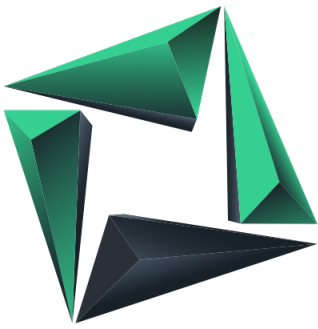
# Table of Contents

# 1. Smart Monitoring

Qosium Scopemon is a measurement controller targeted for automated QoS monitoring. Like always with Qosium, the service/application to be monitored can be everything without limits on network technologies throughout the measurement path. Qosium Scopemon implements all measurement features of [Qosium Probe](#).

Scopemon currently offers three different ways of monitoring:

- Constant:
  - Monitoring is continuously ongoing when Qosium Scopemon is running.
  - Useful when a network path, device, etc., is to be monitored all the time
- Scheduled:
  - Monitoring is activated based on a predetermined schedule, e.g., a certain time of the day.
  - Useful, e.g., when there are critical periods that only need to be monitored
- Flow-based:
  - Optimized monitoring: triggered when the traffic specified as interesting is flowing in the network
  - Useful for session-based sporadic applications/services (e.g., VoIP calls)

The commissioning of Scopemon is straightforward. After installation, the parameterization includes configuring interesting traffic types (applications/services) and measurement paths. Once Scopemon is started, its operation if fully autonomous. It is common to set Qosium Scopemon to start during the boot process of the host machine. This ensures that the monitoring of critical services/applications takes place whenever the network device is powered on.

Qosium Scopemon makes it simple for you to monitor multi-user large-scale networks. The results can be stored in a persistent database from which a real-time situation awareness dashboard of the network can be created.

Qosium takes monitoring accuracy to the next level, which is required, especially for *critical communications systems*, where a decrease of quality can seize production, delay deliveries, or even cause damage and injuries.

# 2. Startup

Qosium Scopemon can be run in various environments. Scopemon is started simply by running the executable `QosiumScopemon`. This will also launch the [GUI](#), where the measurement can be monitored locally. If GUI is parameterized off, Scopemon will just run and perform its tasks in the background. Scopemon is configured by editing [the configuration file](#).

# 3. Scopemon User Interface

The user interface of Scopemon displays results and provides some adjustments for the ongoing measurement. Sometimes there is no need for such local results monitoring. In these cases, the GUI can be left out, and Scopemon handles everything in the background.

## 3.1. Overview

When the Qosium Scopemon's *GUI* is enabled, the default interface consists of 2 windows: *Main window* and *Control Panel window*. The main window is primarily used to monitor the ongoing measurement, while the control panel window hosts the log, measurement on/off toggle, and information about the Scopemon version.
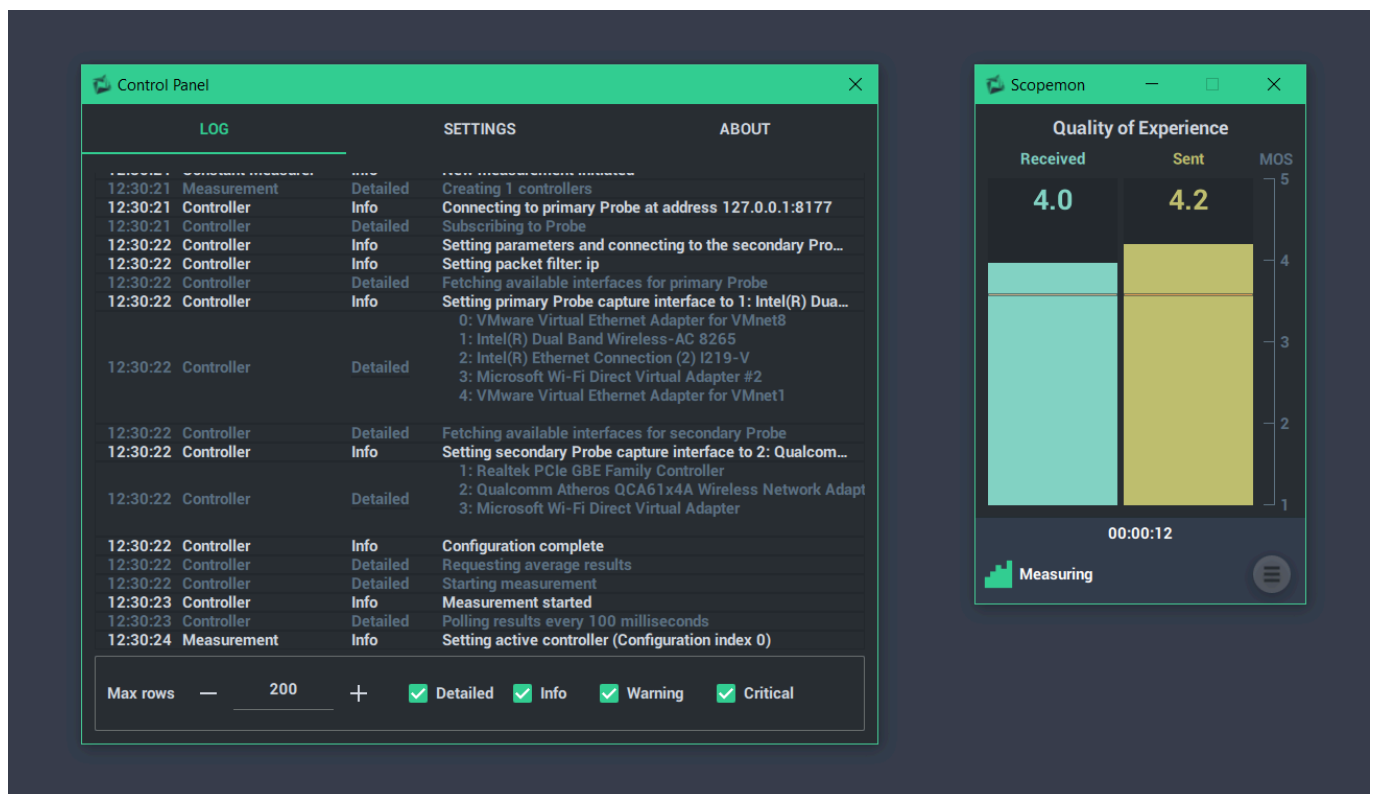


> ℹ The user interface of Scopemon is highly customizable. Therefore, your distribution of Scopemon may have a different appearance, but the basic functionality is nonetheless the same.

## 3.2. Main Window

The main window of Qosium Scopemon consists of two elements:

1. **Charts** (top) - One or more charts that visualize given measurement results
2. **Toolbar** (bottom) - This panel displays current measurement duration and status. The menu button shows/hides the *Control Panel Window*
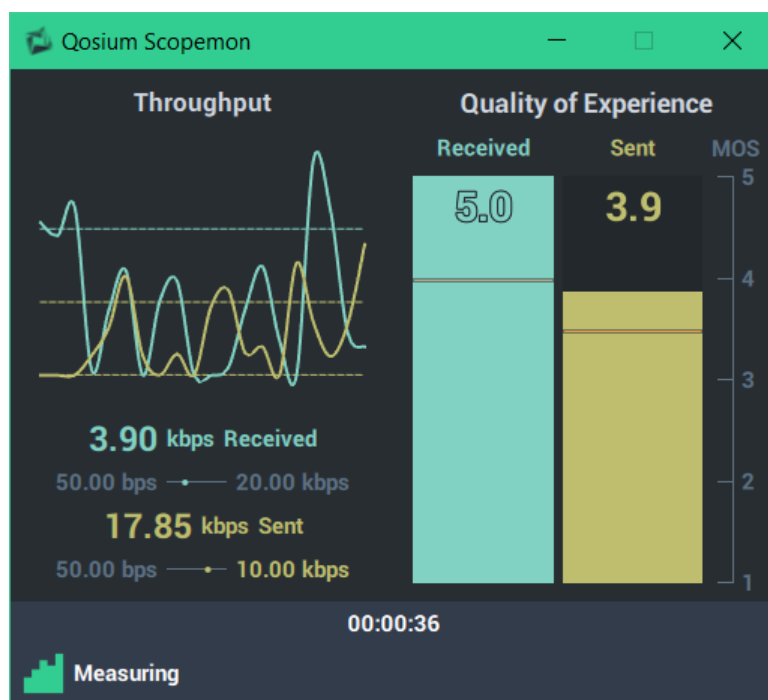
## 3.2.1. Warnings

Scopemon displays some warnings in the main window. These types of warnings usually require user action, be it a misconfiguration or a non-responsive Probe. If Scopemon can deduce that the warning has outlived its usefulness, it will be removed from this list. Some warnings, such as configuration errors, persist until Scopemon is restarted.

Scopemon displays an icon at the bottom of the window (orange exclamation mark inside a triangle) in case of one or more active warnings. Hover over the icon to see the warnings:
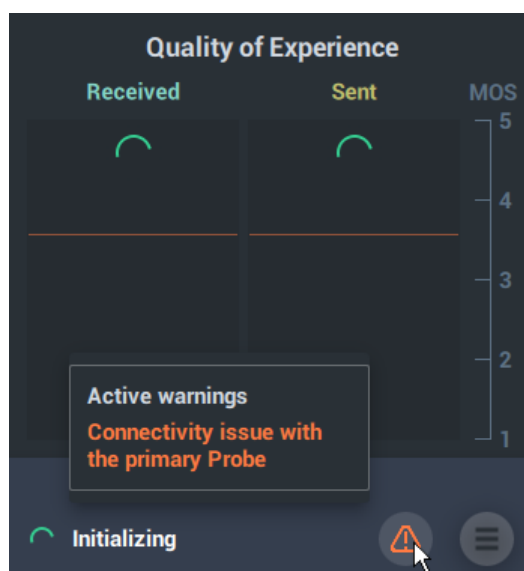


A warning can also be used as a trigger for external alerts in a network management solution.

## 3.3. Control Panel Window

The control panel provides a few pages with miscellaneous purposes. See each page in the following

section for more information.

## 3.3.1. Log Page

The log page displays a list of events. It reports errors, warnings, and other anomalies in the operation of Scopemon.



This page has a few settings available:

- **Max rows** - The maximum number of log messages Scopemon will keep in memory
- **Detailed** - Show/Hide messages related to operational details
- **Info** - Show/Hide messages related main operation
- **Warning** - Show/Hide warning messages
- **Critical** - Show/Hide critical messages (require actions, typically)

## 3.3.2. Settings Page

On the settings page, the measurement can be turned on/off without closing Scopemon. To learn how to adjust Scopemon settings, see Scopemon Configuration.
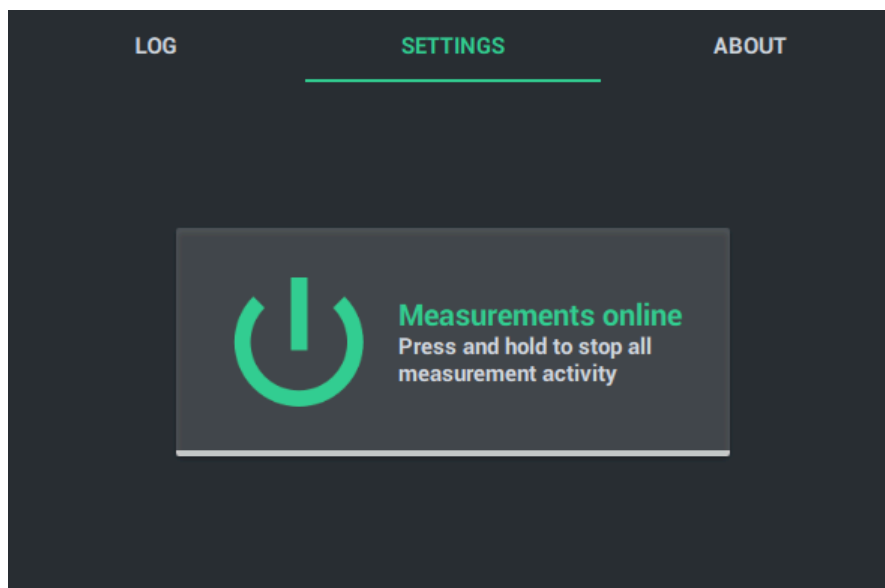
⚠️ Using this functionality does not reload the settings from the configuration file. To reload, restart Scopemon instead.

### 3.3.3. About Page

The about page displays technical information about Scopemon, such as version information and additional license clauses.
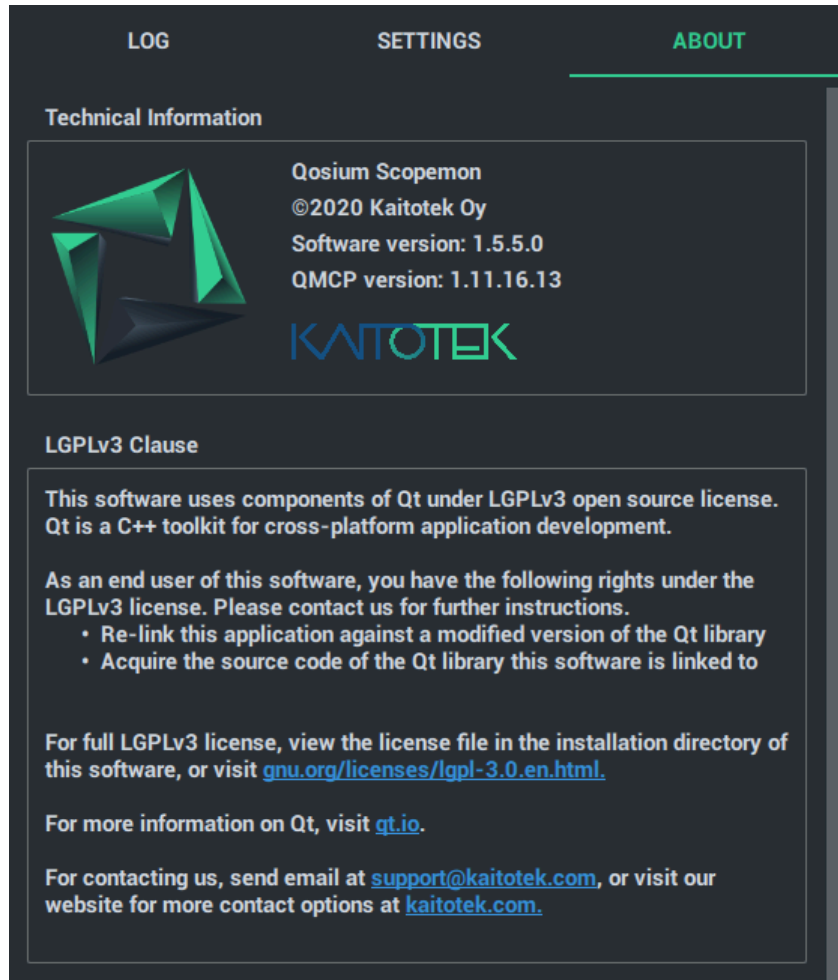
# 4. Scopemon Configuration

Configuration defines how Scopemon connects to Probes, what network traffic is measured, and how results are handled. Scopemon always requires configuration when deployed on a new computer. Familiarizing with parameters, their purpose, and use cases is therefore essential in efficient use of Scopemon.

For a detailed description of each parameter and how to use them, please refer to Scopemon Parameter Reference.

## 4.1. Command Line Parameters

Scopemon 1.5.9.0
Scopemon accepts a few command line parameters. These parameters can be inspected by typing `QosiumScopemon.exe -h` in *command prompt* / *terminal*:

```
C:\Scopemon>QosiumScopemon.exe -h

Qosium Scopemon

Options:
  -?, -h, --help          Displays this help.
  -v, --version           Displays version information.
  -c, --config <FILENAME>  Override the default configuration filename
```

## 4.2. Configuration File

Scopemon is configured by editing the configuration file. By default, this is a file named `config.ini` located in Scopemon installation directory. It is possible to use another file by launching Scopemon with `-c` `<FILENAME>` parameter, where `<FILENAME>` is the absolute or relative path to the configuration file.

The configuration file follows basic *ini* configuration notation. All parameters belong to a *section*, and are defined by using `=`:

```
[SectionName]
parameter1=value
parameter2=value
```

- Each parameter has a list/range of allowed values. Consult each parameter individually for allowed values
- Using an invalid value for a parameter results in a warning/error/undesired behavior
- If a parameter is omitted, the default value will be used instead

## 4.3. Editing & Saving

The configuration file can be edited with any text editor software which supports basic text files. When Scopemon is started, the configuration file is loaded automatically.

A typical configuration editing process is as follows:

- Close running Scopemon instance
- Open the configuration file for editing
- After editing, save changes and close the configuration file
- Restart Scopemon

> After editing configuration file, Scopemon needs to be restarted for the new changes to take effect.

## 4.4. Scopemon Example Configuration

This section presents an example of how to configure Scopemon. It should give an insight into what the configuration file looks like and how it is structured. Most parameters are already set to default values, which are appropriate for most use cases. However, several parameters are almost always required.

### 4.4.1. Configuration File Contents

The configuration file used in this example is presented below. We define only a subset of parameters and leave the rest to their default values.

```
[Application]
admin_tools=true
measurer=1

[ThroughputChart]
received_upper_threshold=10000000
received_upper_alert=true

[ScheduledMeasurer]
schedule_rules\size=2
schedule_rules\1\start_time=8:00:00
schedule_rules\1\end_time=10:00:00
schedule_rules\2\start_time=18:00:00
schedule_rules\2\end_time=23:00:00

[Measurement]
primary_probe_hostname=127.0.0.1
primary_probe_interface_index=2
use_secondary_probe=true
secondary_probe_hostname=192.168.1.236
secondary_probe_interface_index=3
packet_filter=udp port 6889 or udp port 6890
packet_filter_mode=220
write_average_results=true
write_filename_suffix=MyMeasurement
write_path=E:/Measurement Results/
```

Let's go through this configuration, piece by piece, in the next sections.

## 4.4.2. Application Group

Application Group defines the general settings for Scopemon. The most important step in this group is to select the measurer, which controls the measurements autonomously.

```
[Application]
admin_tools=true
measurer=1
```

First, we select a **scheduled measurer** by setting `measurer=1`. We'll need to configure it later in its own group. Let's also select `admin_tools=true` to enable the control panel window and some other extra privileges. It's recommended to turn on the admin tools during setup and testing.

## 4.4.3. ThroughputChart

Let's assume that we want an alert notification each time the received traffic exceeds 10 Mbps. We set the upper threshold by typing `received_upper_threshold=10000000`. This alone does not enable alerts; it only expresses that 10 Mbps is the upper boundary to our range of interest. To enable the alert, we type `received_upper_alert=true`.

```
[ThroughputChart]
received_upper_threshold=10000000
received_upper_alert=true
```

## 4.4.4. ScheduledMeasurer Group

Since we selected the scheduled measurer in our application group, we need to configure it. Let's assume that we want automatic measurements between 8.00-10.00 and 18.00-23.00. We define 2 rules by

assigning `schedule_rules\size=2`. We issue the `start_time` and `end_time` fields to appropriate values for both rules.

```
[ScheduledMeasurer]
schedule_rules\size=2
schedule_rules\1\start_time=8:00:00
schedule_rules\1\end_time=10:00:00
schedule_rules\2\start_time=18:00:00
schedule_rules\2\end_time=23:00:00
```

## 4.4.5. Measurement Group

Regardless of which measurer is selected, the measurement group is always relevant when defining the actual measurement.

```
[Measurement]
primary_probe_hostname=127.0.0.1
primary_probe_interface_index=2
use_secondary_probe=true
secondary_probe_hostname=192.168.1.236
secondary_probe_interface_index=3
packet_filter=udp port 6889 or udp port 6890
packet_filter_mode=220
write_average_results=true
write_filename_suffix=MyMeasurement
write_path=E:/Measurement Results/
```

Let's assume that we want to perform a two-point measurement by using two Probes. Our first Probe is running on a local computer, so we define the hostname as `primary_probe_hostname=127.0.0.1`. The interface we want to use for capturing has an index of 2. Therefore we define `primary_probe_interface_index=2`. The same settings must be used for the second Probe, in addition to `use_secondary_probe=true`.

But we don't want to capture any traffic, but just certain UDP traffic. For a custom packet filter, we need to set two parameters. First, by typing `packet_filter_mode=220`, we tell Scopemon that we want to use our own packet filter instead of an auto-generated one. Then we type `packet_filter=udp port 6889 or udp port 6890` to capture only UDP traffic going through port 6889 or 6890.

Finally, we write the average results to disk by issuing `write_average_results=true`. The last parameters define the filename suffix and the write directory.

### 4.4.5.1. Determining Interfaces

The capture interface index identifies the network interface which is used in the measurement setup. Qosium uses internal indexing for interfaces, which is not the same as the order your OS is displaying. For both the primary and the secondary Probe, capture indexes need to be defined manually. Defining capture interface indexes correctly is vital in configuration, and ignoring these indexes is one reason why measurement configuration fails.

There is no direct way to deduce the index for each interface that Qosium uses. To find out the indexes of interfaces, follow these steps:

- Configure Scopemon as usual
- Assign any number to interfaces indexes, or omit them from the configuration file
- Set `measurer` to `0` so that measurement starts immediately
- Set `admin_tools` to `true` to access the log page in the control panel window

After following the steps above, launch Scopemon. The log window displays the available interfaces (Make sure **Info** and **Detailed** messages are enabled):



If, for example, we want to use `Intel(R) Dual Band Wireless-AC 8265` (index 1) as the primary Probe capture interface and `Qualcomm Atheros QCA61x4A Wireless Network Adapter` (index 2) as the secondary capture interface, we would update the configuration file with the following information:

```
[Measurement]
primary_probe_interface_index=1
secondary_probe_interface_index=2
```

## 4.5. Scopemon Parameter Reference

Scopemon is configured via parameters. This reference lists all available parameters, default values, allowed values, and use examples.

### 4.5.1. Introduction

The reference is organized in sections. Each section corresponds to a group in the configuration file.

### 4.5.2. Application Group

The application group contains all general parameters related to the operation of Scopemon.

#### 4.5.2.1. admin_tools

When set to true, Log and Settings windows are available, and the user is given elevated privileges.

- Values:
  - `true` - Admin tools are enabled
  - `false` - Admin tools are disabled
- Default: `true`

**Example**

```
[Application]
admin_tools=true
```

## 4.5.2.2. measurer

Determine the type of measurer. The measurer is responsible for starting and stopping automatic measurements.

- Values:
  - `0` Constant Measurer - Always-on measurement
  - `1` Scheduled Measurer- Measure within specific time period
  - `2` Flow Monitor Measurer - Measure specific traffic flow
- Default: `0`

**Example**

To switch to scheduled measurer, defined this parameter as:

```
[Application]
measurer=1
```

## 4.5.2.3. start_offline

Determine if Scopemon should start in an offline state. Measurement functionality must be turned on manually from the settings window.

- Values:
  - `true` - Application starts with offline mode
  - `false` - Application starts with online mode
- Default: `false`

> If admin_tools are disabled, this option has no effect, and Scopemon is started in online mode

**Example**

```
[Application]
start_offline=true
```

## 4.5.2.4. stay_on_top

When set to true, Scopemon stays on top of other windows even when not active. This is especially useful when performing other tasks on the device while simultaneously keeping an eye on Scopemon GUI.

- Values:
  - `true` - Scopemon stays on top of other windows even when not active
  - `false` - Scopemon recedes behind other windows when they become active
- Default: `false`

### Example

```
[Application]
stay_on_top=true
```

## 4.5.2.5. taskbar_alert

When set to true, Scopemon gives a taskbar notification when one or more visualization charts yield an alert. The visual appearance of the taskbar notification depends on the operating system. This feature is activated only if the Scopemon window is minimized while the alert is given.

- Values:
  - `true` - Scopemon gives a taskbar notification while minimized when an alert is raised
  - `false` - Scopemon gives no taskbar notifications
- Default: `false`

### Example

```
[Application]
taskbar_alert=true
```

## 4.5.3. Log Group

Log group contains settings related to the log window and writing the log to one or multiple files.

## 4.5.3.1. write_file_datecode_format

When writing multiple log files, this date code defines the format used to determine the date part of the log filename. This parameter also implicitly determines the file-swapping frequency since a new log file is created as soon as this date code changes.

- Type: `string`
- Default: `yyyyMMdd`

### Example
To save log files every hour, define the following parameters as:

```
[Log]
write_log_file=true
write_filename_format={DATE}_hourly.log
write_file_datecode_format=yyyyMMdd-hh
```

## 4.5.3.2. write_file_dir

Determines the directory where the log files are saved. By default, the log files are saved to `C:\Users\<user>\AppData\Local\Kaitotek\Qosium Scopemon\logs` in Windows.

- Type: `string`

> ⚠️ Use forward slashes / or double backslashes \\ as directory separators

**Example**

To save log files to directory `E:\ScopemonLogs\`, define this parameters as:

```
[Log]
write_file_dir=E:\\ScopemonLogs
```

### 4.5.3.3. write_filename_format

Determines the filename format of the log file or multiple log files.

- Type: `string`
- Default: `{DATE}_scopemon.log`

> ⚠️ Use forward slashes / or double backslashes \\ as directory separators.

**Example**

To save log files to directory `E:\ScopemonLogs\`, define this parameters as:

```
[Log]
write_file_dir=E:\\ScopemonLogs
```

### 4.5.3.4. write_mode_append

Determines whether the log file is opened in *overwrite* or *append* mode when using single file mode. This option has no effect when writing multiple log files, as they are always opened in *append* mode.

- Values:
  - `true` - File is opened in append mode, and new log entries are added at the end of the file
  - `false` - The log file contents will be erased each time Scopemon is started
- Default: `false`

> ⚠️ When using append mode, the file size grows indefinitely. It is advisable to set this to false in production environments or to make sure the file size won't become a problem by other means.

**Example**

To preserve log messages from previous runs, set this parameter as:

```
[Log]
write_mode_append=true
```

### 4.5.3.5. write_log_file

Determines whether Scopemon writes the log into file(s). By default, Scopemon does not produce log files.

- Values:
  - `true` - Log file(s) are written
  - `false` - Log files won't be written
- Default: `false`

## Example

To enable log writing, define this parameter as:

```
[Log]
write_log_file=true
```

## 4.5.4. FlowMonitorMeasurer Group

Flow Monitor Measurer has several parameters, which define how it detects the target traffic flow.

### 4.5.4.1. flowmap_interval

Determines how often the flow map is collected. A lower value consumes more resources but is more responsive.

- Unit: `milliseconds`
- Precision: `integer`
- Minimum: `50`
- Default: `1000`

> 📝 Setting this value below `200` is not advisable in production environments or regular use.

## Example

To make Scopemon collect flow results twice per second (i.e., every 500 ms), define this parameter as:

```
[FlowMonitorMeasurer]
flowmap_interval=500
```

### 4.5.4.2. flow_timeout

The duration in which a flow can remain inactive before the measurement is stopped. Setting this value too low may result in duplicate measurement for a single traffic flow. Setting this value too high results in measurement carrying unnecessarily long after the flow has ended.

- Unit: `seconds`
- Precision: `integer`
- Minimum: `1`
- Default: `10`

## Example

To set the flow timeout to 5 seconds, define this parameter as:

```
[FlowMonitorMeasurer]
flow_timeout=5
```

### 4.5.4.3. operation_mode

Flow monitoring can be set to operate either in single flow or multi-flow mode. In single flow mode, Scopemon expects to detect a single flow at a time and create a measurement for each flow. The detection of more than 1 flow results in a warning. In multi-flow mode, Scopemon carries out a measurement as long

as one or more flows are detected.

- Values:
  - `0` Multi-flow mode - Detect and measure multiple flows
  - `1` Single flow mode - Detect and measure a single flow
- Default: `0`

### Example
To switch to single flow mode, define this parameter as:

```
[FlowMonitorMeasurer]
operation_mode=1
```

## 4.5.4.4. probe_hostname

The hostname of the Probe which is used for flow monitoring. This can be omitted if the Probe is located on the same device where Scopemon is used.

- Type: `string`
- Default: `127.0.0.1`

### Example
If the Probe is installed in another device at IP address *192.168.1.43*, define this parameter as:

```
[FlowMonitorMeasurer]
probe_hostname=192.168.1.43
```

## 4.5.4.5. probe_interface_index

Capture interface for flow monitoring. Typically `0` is the default OS interface.

- Precision: `integer`
- Minimum: `0`
- Default: `0`

### Example
If the capture interface index is 2, define this parameter as:

```
[FlowMonitorMeasurer]
probe_interface_index=2
```

## 4.5.4.6. probe_port

The port number of the local Probe. This can be typically omitted unless the port where Probe serves control connections has been changed in Probe configuration.

- Precision: `integer`
- Minimum: `0`
- Maximum: `65535`
- Default: `8177`

**Example**

If Probe is configured to serve control connections on port 9776, define this parameter as:

```
[FlowMonitorMeasurer]
probe_port=9776
```

## 4.5.4.7. packet_filter

Packet filter is one of the most important parameters, as it defines which traffic flows are measured. The packet filter needs to be strict enough so that no irrelevant flows are captured. Otherwise the application is unable to select a flow for monitoring and throws out a flow unambiguity warning. For more information, see Packet Filters in Qosium.

- Type: `string`
- Default: `ip`

It is important to understand that when a flow is found, the filter defined here will also become the measurement filter when its mode is manual.

**Example**

To enable monitoring only for UDP traffic going through ports 6889 or 6890, define this parameter as:

```
[FlowMonitorMeasurer]
packet_filter=udp port 6889 or udp port 6890
```

## 4.5.4.8. reconnect_interval

If a connection cannot be established to the local Probe, Scopemon waits for a duration specified by this parameter and then attempts to reconnect.

- Unit: `milliseconds`
- Precision: `integer`
- Minimum: `0`
- Default: `1000`

**Example**

To attempt a reconnection after 500 milliseconds, define this parameter as:

```
[FlowMonitorMeasurer]
reconnect_interval=500
```

## 4.5.4.9. use_promiscuous_mode

Promiscuous mode allows the detection of incoming traffic that is not directed to the selected network interface. This scenario is common when capturing mirrored traffic, e.g., from a switch.

- Values:
  - `true` - Allow detection of all incoming traffic
  - `false` - Allow detection of incoming traffic destined only for this interface
- Default: `true`

### Example

To disable detection of traffic not designated to the network interface, define this parameter as:

```
[FlowMonitorMeasurer]
use_promiscuous_mode=false
```

## 4.5.4.10. use_remote_probe

By default, Flow Monitor Measurer carries out measurements by using two Probes. With this setting, it's possible to disable the use of remote Probe entirely.

- Values:
  - `true` - Perform a two-point measurement
  - `false` - Perform a single-point measurement
- Default: `true`

⚠️ Single-point measurement significantly limits the number of available measurement result types

### Example

To perform a single-point measurement using the local Probe only, define this parameter as:

```
[FlowMonitorMeasurer]
use_remote_probe=false
```

## 4.5.4.11. user_id

User ID can be used to identify a controller, i.e., the Qosium Scopemon instance in this case. You can set this freely. The set value will appear in the results, where it can be used as a parameter to find results. Thus, you can use this as you wish as an identifier for your measurement, e.g., in a large-scale measurement setup. The User ID here applies only for the flow monitor measurement.

- Precision: `integer`
- Minimum: `0`
- Maximum: '4294967295'
- Default: `0`

### Example

To set an id of 6 for this client, define this parameter as:

```
[FlowMonitorMeasurer]
user_id=6
```

## 4.5.4.12. write_date_code_format

Date code format governs the frequency of file creation when `write_multiple_files`. Whenever Scopemon detects a change in the date code, it automatically triggers new result files. A timestamp with this date code is then appended to the filename.

- Type: `string`
- Default: `yyyyMMdd`

### Example

To write results every hour, define this parameter as:

```
[FlowMonitorMeasurer]
write_multiple_files=true
write_date_code_format=yyyyMMdd-hh
```

## 4.5.4.13. write_filename_suffix

File suffix string when forming a filename for measurement result files.

- Type: `string`
- Default: Empty

> This option has no impact when *write_flowmap_to_file* is set to *false*

### Example

If defined for example as "MyMeasurement", the resulting filename is "flows_MyMeasurement.txt".

```
[FlowMonitorMeasurer]
write_flowmap_to_file=true
write_filename_suffix=MyMeasurement
```

## 4.5.4.14. write_flowmap_to_file

When true, flow measurement results are written to file.

- Values:
  - `true` - Results are written to file
  - `false` - Results are not written to file
- Default: `false`

### Example

```
[FlowMonitorMeasurer]
write_flowmap_to_file=true
```

## 4.5.4.15. write_multiple_files

When true, flow results are written to multiple files. By default, one file is created for each day. For configuring multiple file writing frequency, see write_date_code_format.

- Values:
  - `true` - Results are written to multiple files
  - `false` - All results are written into a single file
- Default: `false`

> 📄 This option has no impact when *write_flowmap_to_file* is set to *false*

## Example

```
[FlowMonitorMeasurer]
write_flowmap_to_file=true
write_multiple_files=true
```

### 4.5.4.16. write_path

Set to override the path where measurement result files are stored. Use `/` as the directory separator.

- Type: `string`
- Default: Scopemon root directory

> 📄 This option has no impact when *write_flowmap_to_file* is set to *false*

## Example

```
[FlowMonitorMeasurer]
write_flowmap_to_file=true
write_path=c:/temp
```

## 4.5.5. Measurement Group

Scopemon is configured via parameters. This reference lists all available parameters, default values, allowed values, and use examples for the measurement.

### 4.5.5.1. General

#### 4.5.5.1.1. measurement_description

Verbose description of the measurement. This value is written to results files as metadata.

- Type: `string`
- Default: `[empty]`

**Example**
To name this measurement "My measurement", define this parameter as:

```
[Measurement]
measurement_description=My measurement
```

#### 4.5.5.1.2. measurement_start_offset

Artificially delay the start of the measurement by the given time. If the value is 0, the measurement starts as soon as possible once triggered.

- Unit: `milliseconds`
- Precision: `integer`
- Minimum: `0`

- Default: 0

## Example

To delay the start of the measurement by 1 second, define this parameter as:

```
[Measurement]
measurement_start_offset=1000
```

> 📑 This feature is used in particular scenarios and is rarely needed

### 4.5.5.1.3. reconnect_interval

If a connection cannot be established to the primary Probe, Scopemon waits for a duration specified by this parameter and then attempts to reconnect.

- Unit: `milliseconds`
- Precision: `integer`
- Minimum: 0
- Default: `1000`

## Example

To attempt a reconnection after 500 milliseconds, define this parameter as:

```
[Measurement]
reconnect_interval=500
```

### 4.5.5.1.4. robust_mode_max_cbd

This parameter defines the maximum connection break duration allowed in the QMCP connections related to the measurement session. If a connection break is longer than the defined value, the measurement session will end to a timeout error.

- Unit: `minutes`
- Precision: `integer`
- Minimum: `1`
- Default: `10`

## Example

To allow connection breaks of maximum of three minutes, define this parameter as:

```
[Measurement]
robust_mode_max_cbd=3
```

### 4.5.5.1.5. user_id

User ID can be used to identify a controller, i.e., the Qosium Scopemon instance in this case. You can set this freely. The set value will appear in the results, where it can be used as a parameter to find results. Thus, you can use this as you wish as an identifier for your measurement, devices, etc., in a large-scale measurement setup.

- Precision: `integer`

- Minimum: `0`
- Maximum: `4294967295`
- Default: `0`

### Example

To set an id of 6 for this client, define this parameter as:

```
[Measurement]
user_id=6
```

## 4.5.5.2. Topology

### 4.5.5.2.1. primary_probe_hostname

The hostname (or directly the IPv4 address) of the primary Probe. This can be omitted if the Probe is located on the same device where Scopemon is used.

- Type: `string`
- Default: `127.0.0.1`

### Example

If the primary Probe is installed in another device at *myhost.home*, define this parameter as:

```
[Measurement]
primary_probe_hostname=myhost.home
```

### 4.5.5.2.2. primary_probe_port

The port number of the primary Probe. This can be typically omitted unless the port where Probe serves control connections has been changed in Probe configuration.

- Precision: `integer`
- Minimum: `0`
- Maximum: `65535`
- Default: `8177`

### Example

If Probe is configured to serve control connections on port 9776, define this parameter as:

```
[Measurement]
primary_probe_port=9776
```

### 4.5.5.2.3. use_secondary_probe

By default, measurement is performed with one Probe. With this setting, it's possible to set a two-point measurement.

- Values:
  - `true` - Perform a two-point measurement
  - `false` - Perform a single-point measurement
- Default: `false`

## Example

To perform a two-point measurement using the Probe at 192.168.1.14 and interface #4, set as follows:

```
[Measurement]
use_secondary_probe=false
secondary_probe_hostname=192.168.1.14
secondary_probe_interface_index=4
```

ⓘ If secondary Probe is not used, i.e., the measurement is a single-point one, all the secondary Probe parameters are just ignored.

⚠ Single-point measurement significantly limits the number of available measurement result types.

### 4.5.5.2.4. secondary_probe_hostname

The hostname (or directly the IPv4 address) of the secondary Probe.

- Type: `string`
- Default: `127.0.0.1`

## Example

If a secondary Probe is installed in another device at IP address *192.168.1.43*, define this parameter as:

```
[Measurement]
secondary_probe_hostname=192.168.1.43
```

📄 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.2.5. secondary_probe_port

The port number of the secondary Probe. This can be typically omitted unless the port where Probe serves control connections has been changed in Probe configuration.

- Precision: `integer`
- Minimum: `0`
- Maximum: `65535`
- Default: `8177`

## Example

If Probe is configured to serve control connections on port 9778, define this parameter as:

```
[Measurement]
secondary_probe_port=9778
```

📄 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.2.6. primary_probe_placement

The topological placement of the primary Probe.

- Values:
  - `10` Measurement end-point - Probe is in either one of the endpoints of the measured traffic. In other words, the device Probe is installed to is either sending or receiving the measured network traffic.
  - `100` Within measured path - Probe is not located at either one of the end-points but instead resides somewhere along the path where the measured traffic traverses.
  - `200` Off-path - Probe is not located within the measurement path at all. This is the case, e.g., when the Probe is located in a separate device where the traffic to be measured is mirrored.
- Default: `10`

### 4.5.5.2.7. primary_probe_interface_index

This parameter defines the capture interface of the primary Probe to be used in the measurement. The interface numbering in a device is unique per Qosium installation. Thus, once Qosium Probe is installed on a device, a particular NIC will always have the same interface index. The numbering can change if Qosium Probe is removed and reinstalled.

- Precision: `integer`
- Minimum: `0`
- Default: `0`

#### Example
If desired the capture interface has index of 2, define this parameter as:

```
[Measurement]
primary_probe_interface_index=2
```

> (i) To see the available interfaces and their indices start Scopemon and check its log.

### 4.5.5.2.8. secondary_probe_interface_index

This parameter defines the capture interface of the secondary Probe to be used in the measurement.

- Precision: `integer`
- Minimum: `0`
- Default: `0`

#### Example
If the index of the desired capture interface is 2, define this parameter as:

```
[Measurement]
secondary_probe_interface_index=2
```

> This parameter has no effect when the secondary Probe is disabled

### 4.5.5.2.9. secondary_probe_placement

The topological placement of the secondary Probe.

- Values:

  - `10` Measurement end-point - Probe is in either one of the endpoints of the measured traffic. In other words, the device Probe is installed to is either sending or receiving the measured network traffic
  - `100` Within measured path - Probe is not located at either one of the end-points but instead resides somewhere along the path where the measured traffic traverses
  - `200` Off-path - Probe is not located within the measurement path at all. This is the case, e.g., when the Probe is located in a separate device where the traffic to be measured is mirrored.

- Default: `10`

> This parameter has no effect when the secondary Probe is disabled

### 4.5.5.2.10. nat_between_probes

Qosium needs to be aware if a *NAT* occurs between Probes. If this is the case, enable this parameter.

- Values:

  - `true` - There's a NAT occurring between Probes
  - `false` - No NAT is occurring between Probes

- Default: `false`

> This parameter has no effect when the secondary Probe is disabled

## 4.5.5.3. Filtering

Packet filter is one of the most important parameters, as it defines which traffic is measured. The packet filter needs to be strict enough so that no irrelevant traffic is captured. Otherwise, the results may not be useful.

### 4.5.5.3.1. packet_filter_mode

This parameter determines the mode in which packets are filtered. In most cases, the selection is between *Automatic*, which generates an automatic filter, or *Manual*, which allows the use of a manual filter defined in [packet_filter](). For more information, see [Packet Filters in Qosium]().

- Values:

  - `220` Manual - Packet filter is defined manually in [packet_filter](). In a two-point measurement, this filter is used in the secondary Probe as well.
  - `240` Automatic - Generates automatically a filter, which includes all IP traffic between the hosts (a two-point measurement) or the measurement point's own IP traffic (a single-point measurement). The parameter [packet_filter]() will be ignored.
  - `231` Automatic for secondary (strict) - This mode is meant for cases where a *NAT* is between the measurement points in a two-point measurement. The filter is set manually for the primary Probe, but Qosium generates an automatic filter for the secondary Probe. The generated filter will be [strict](), focusing on a single flow, so define the primary Probe filter to be strict as well.
  - `233` Automatic for secondary (light) - This mode is similar to the previous, but now a [loose]() automatic

filter is generated for the secondary Probe. A *loose filter* includes only addresses, so all traffic traveling between these addresses will be included. Remember to define the primary Probe's manual filter to be loose as well.

- Default: `240`

### Example

```
[Measurement]
packet_filter_mode=231
```

⚠️ Only end-point placements of Probes allow the Packet filter to be calculated automatically.

### 4.5.5.3.2. packet_filter

When packet_filter_mode is Manual, use this parameter to define the filter. In addition, when using the *NAT* automatic filtering modes, the primary Probe filter is defined here.

- Type: `string`
- Default: `ip`

For more information, see Packet Filters in Qosium.

### Example

To enable monitoring only for *UDP* traffic going through ports 6889 or 6890, define this parameter as:

```
[Measurement]
packet_filter=udp port 6889 or udp port 6890
```

⚠️ If you are running Scopemon in Flow Monitor Measurer mode, the manual filter defined here will be overruled by the filter defined under FlowMonitorMeasurer.

### 4.5.5.4. Senders

Sometimes it is not clear which way the traffic is traveling in the network. In these cases, you need to tell it to Qosium by defining senders manually. See Direction of Traffic and Senders under concepts section for more information what the senders mean.

In a single-point measurement, you need to define the senders manually if Probe's placement is *Off-path*.

In a two-point measurement, you need to define the senders manually in the following cases:

- If both Probes are *Off-path*, you need to define senders for both manually.
- If one Probe is *Off-path* and the other is *Within path*, you need to define the senders manually for the *Off-path* Probe.

Otherwise, Qosium defines the senders automatically, and the following parameters in this category are ignored.

### 4.5.5.4.1. primary_probe_senders_pure_mac_method_enabled

Determines whether the pure MAC method is used for defining primary Probe senders. When enabled, the senders are defined only based on MAC addresses and no other senders settings are required for the

primary Probe.

- Values:
    - `true` - Pure MAC method is used
    - `false` - Pure MAC method is not used
- Default: `false`

> ⚠️ This mode works only if the measured traffic contains Ethernet-like MAC addresses.

### 4.5.5.4.2. primary_probe_senders_eth_mode

The Ethernet senders mode of the primary Probe.

- Values:
    - `0` Auto-search - Use the Ethernet addresses of the device's interfaces as senders.
    - `249` Manual - Input sender addresses manually.
    - `250` Inverse definition - The senders are defined according to the senders of the secondary Probe.
    - `252` Mask - Define the senders manually by using a mask instead of individual addresses.
- Default: `0`

#### Example

```
[Measurement]
primary_probe_senders_eth_mode=249
```

### 4.5.5.4.3. primary_probe_senders_eth_address_list

The manual Ethernet senders list of the local Probe. Used only when primary_probe_senders_eth_mode is set to *manual* or *mask* mode.

#### Example (Manual mode)

```
[Measurement]
primary_probe_senders_eth_address_list/size=2
primary_probe_senders_eth_address_list/1/address=12:34:56:78:9a:bc
primary_probe_senders_eth_address_list/2/address=12:34:56:78:9a:bd
```

### 4.5.5.4.4. primary_probe_senders_ipv4_mode

The IPv4 senders mode of the primary Probe.

- Values:
    - `0` Auto-search - Use the IPv4 addresses of the device's interfaces as senders.
    - `249` Manual - Input sender addresses manually.
    - `250` Inverse definition - The senders are defined according to the senders of the secondary Probe.
    - `252` Mask - Define the senders manually by using a network mask instead of individual addresses.
- Default: `0`

#### Example

```
[Measurement]
primary_probe_senders_ipv4_mode=252
```

> ⚠ When using the Inverse definition or Pure MAC method, the Packet filter cannot be calculated automatically.

### 4.5.5.4.5. primary_probe_senders_ipv4_address_list

The manual IPv4 senders list of the local Probe. Used only when primary_probe_senders_ipv4_mode is set to *Manual* or *Mask* mode.

**Example (Mask mode)**

```
[Measurement]
primary_probe_senders_ipv4_address_list/size=1
primary_probe_senders_ipv4_address_list/1/address=192.168.1.0
primary_probe_senders_ipv4_address_list/1/value=255.255.255.0
```

### 4.5.5.4.6. primary_probe_senders_ipv6_mode

The IPv6 senders mode of the primary Probe.

- Values:
  - `0` Auto-search - Use the IPv6 addresses of the device's interfaces as senders.
  - `249` Manual - Input sender addresses manually.
  - `250` Inverse definition - The senders are defined according to the senders of the secondary Probe.
  - `252` Mask - Define the senders manually by using a network mask instead of individual addresses.
- Default: `0`

**Example**

```
[Measurement]
primary_probe_senders_ipv6_mode=249
```

> ⚠ When using the Inverse definition or Pure MAC method, the Packet filter cannot be calculated automatically.

### 4.5.5.4.7. primary_probe_senders_ipv6_address_list

The manual IPv6 senders list of the local Probe. Used only when primary_probe_senders_ipv6_mode is set to *Manual* or *Mask* mode.

**Example (Manual mode)**

```
[Measurement]
primary_probe_senders_ipv6_address_list/size=1
primary_probe_senders_ipv6_address_list/1/address=fe80:12ab:c839:8df9::1
```

### 4.5.5.4.8. secondary_probe_senders_pure_mac_method_enabled

Determines whether the pure MAC method is used for defining secondary Probe senders. When enabled, the senders are defined only based on MAC addresses and no other senders settings are required for the

secondary Probe.

- Values:
  - `true` - Pure MAC method is used
  - `false` - Pure MAC method is not used
- Default: `false`

> ⚠ This mode works only if the measured traffic contains Ethernet-like MAC addresses.

> 📋 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.4.9. secondary_probe_senders_eth_mode

The Ethernet senders mode of the secondary Probe.

- Values:
  - `0` Auto-search - Use the Ethernet addresses of the device's interfaces as senders.
  - `249` Manual - Input sender addresses manually.
  - `250` Inverse definition - The senders are defined according to the senders of the secondary Probe.
  - `252` Mask - Define the senders manually by using a network mask instead of individual addresses.
- Default: `250`

**Example**

```
[Measurement]
secondary_probe_senders_eth_mode=252
```

> 📋 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.4.10. secondary_probe_senders_eth_address_list

The manual Ethernet senders list of the secondary Probe. Used only when secondary_probe_senders_eth_mode is set to *manual* or *mask*.

**Example (Mask)**

```
[Measurement]
secondary_probe_senders_eth_address_list/size=1
secondary_probe_senders_eth_address_list/1/address=12:34:56:78:9a:00
secondary_probe_senders_eth_address_list/1/value=ff:ff:ff:ff:ff:00
```

> 📋 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.4.11. secondary_probe_senders_ipv4_mode

The IPv4 senders mode of the secondary Probe.

- Values:

- `0` Auto-search - Use the IPv4 addresses of the device's interfaces as senders.
  - `249` Manual - Input sender addresses manually.
  - `250` Inverse definition - The senders are defined according to the senders of the secondary Probe.
  - `252` Mask - Define the senders manually by using a network mask instead of individual addresses.
- Default: `250`

### Example

```
[Measurement]
secondary_probe_senders_ipv4_mode=249
```

⚠️ When using the Inverse definition or Pure MAC method, the Packet filter cannot be calculated automatically.

📝 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.4.12. secondary_probe_senders_ipv4_address_list

The manual IPv4 senders list of the secondary Probe. Used only when secondary_probe_senders_ipv4_mode is set to *Manual* or *Mask*.

### Example (Manual)

```
[Measurement]
secondary_probe_senders_ipv4_address_list/size=3
secondary_probe_senders_ipv4_address_list/1/address=10.0.0.1
secondary_probe_senders_ipv4_address_list/2/address=10.0.0.4
secondary_probe_senders_ipv4_address_list/3/address=10.0.0.12
```

📝 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.4.13. secondary_probe_senders_ipv6_mode

The IPv6 senders mode of the secondary Probe.

- Values:
  - `0` Auto-search - Use the IPv6 addresses of the device's interfaces as senders.
  - `249` Manual - Input sender addresses manually.
  - `250` Inverse definition - The senders are defined according to the senders of the secondary Probe.
  - `252` Mask - Define the senders manually by using a network mask instead of individual addresses.
- Default: `250`

### Example

```
[Measurement]
secondary_probe_senders_ipv6_mode=252
```

⚠️ When using the Inverse definition or Pure MAC method, the Packet filter cannot be calculated automatically.

📝 This parameter has no effect when the secondary Probe is disabled

### 4.5.5.4.14. secondary_probe_senders_ipv6_address_list

The manual IPv6 senders list of the secondary Probe. Used only when secondary_probe_senders_ipv6_mode is set to *Manual* or *Mask*.

**Example (Mask)**

```
[Measurement]
secondary_probe_senders_ipv6_address_list/size=1
secondary_probe_senders_ipv6_address_list/1/address=fe80:1234:5678::0
secondary_probe_senders_ipv6_address_list/1/value=ffff:ffff:ffff::0
```

📝 This parameter has no effect when the secondary Probe is disabled

## 4.5.5.5. Measurement Details

### 4.5.5.5.1. averaging_interval

Determines how often quality is measured for the ongoing measurement. Lower value gives more detailed results and consumes more resources. A higher value gives smoother values.

- Unit: `milliseconds`
- Precision: `integer`
- Minimum: `50`
- Default: `1000`

ℹ️ If you are seeking packet-level resolution for the statistics, do not try to do it by decreasing Averaging interval. Instead, consider using Packet QoS Statistics

**Example**
To make Scopemon collect quality results twice per second (i.e., every 500 ms), define this parameter as:

```
[Measurement]
averaging_interval=500
```

### 4.5.5.5.2. packet_id_method

This parameter defines how Probes identify packets during a measurement. The mode Automatic is the recommended one.

See Qosium Scope's Packet Identification Method for more details of this parameter.

- Values:
  - `10` Automatic - Qosium selects the method from the options below based on the measurement scenario.

- 50 IPv4 ID Field - Qosium uses the *Identification field* in the IPv4 header for packet identification.
- 60 RTP Sequence Number - Qosium uses the *Sequence number field* in the *RTP* header for packet identification.
- 100 Payload-Based ID - Qosium calculates the identification based on the packet payload. If a packet has no payload, *IP4 ID Field*, when present, is used.
- 110 Extended Payload-Based ID - Qosium calculates the identification based on the packet payload, including some parts of the transport layer header.
- 120 Pure Payload-Based ID - This is a very similar method with *Payload-Based ID*, but packets without payload are just ignored from QoS calculation.
- 200 NAT bypasser + Payload based ID - Operates as *Payload-Based ID* but with NAT bypasser functionality enabled.
- 210 NAT Bypasser + Pure Payload Based ID - Operates as *Pure Payload-Based ID* but with NAT bypasser functionality enabled.

- Default: 10

### Example

```
[Measurement]
packet_id_method=50
```

### 4.5.5.5.3. packet_loss_timer

This parameter defines how long to wait for a packet before considering it lost. Thus, selecting a small value may cause delayed packets to be considered lost even though they would later arrive at the destination. The Automatic setup is recommended for most use cases.

- Unit: `milliseconds`
- Precision: `integer`
- Special value: 0 - Automatic
- Minimum: 1
- Default: 0

### Example
To allow packet delay up to 2 seconds, define this parameter as:

```
[Measurement]
packet_loss_timer=2000
```

> ⓘ   Keep this at least on the same level as the Averaging Interval, unless you are using a long Averaging Interval (> 5 s)

.

### 4.5.5.5.4. pk_delay_threshold

Packets with a delay above this threshold are counted in [QoS Statistics: Th. ex. delay pkts](QoS Statistics: Th. ex. delay pkts).

- Precision: `integer`
- Unit: `microseconds`
- Minimum: 0

- Default: `100000`

### Example

To count packets that have a delay of 500 ms (500000 μs), define this parameter as:

```
[Measurement]
pk_delay_threshold=500000
```

### 4.5.5.5.5. pk_jitter_threshold

Packets with a jitter above this threshold are counted in [QoS Statistics: Th. ex. jitter pkts](#).

- Precision: `integer`
- Unit: `microseconds`
- Minimum: `0`
- Default: `100000`

### Example

To count packets that have a jitter of 100 ms (100000 μs), define this parameter as:

```
[Measurement]
pk_jitter_threshold=100000
```

### 4.5.5.5.6. use_promiscuous_mode

Promiscuous mode allows the detection of incoming traffic that is not directed to the selected network interface. This scenario is common when capturing mirrored traffic, e.g., from a switch.

- Values:
  - `true` - Allow detection of all incoming traffic
  - `false` - Allow detection of incoming traffic destined only for this interface
- Default: `true`

### Example

To disable detection of traffic not designated to the network interface, define this parameter as:

```
[Measurement]
use_promiscuous_mode=false
```

## 4.5.5.6. QoE Averaging

When using QoE methods, it is recommended to use averaging because it resembles better how humans perceive connection quality.

### 4.5.5.6.1. use_qoe_swa

Enable or disable sliding window averaging (SWA) for quality estimates.

- Values:
  - `true` - Enable SWA
  - `false` - Disable SWA

- Default: `true`

### Example

```
[Measurement]
use_qoe_swa=true
```

### 4.5.5.6.2. use_qoe_wma

Enable or disable weighted moving averaging (WMA) for quality estimates.

- Values:
  - `true` - Enable WMA
  - `false` - Disable WMA
- Default: `true`

### Example

```
[Measurement]
use_qoe_wma=false
```

### 4.5.5.6.3. qoe_swa_window_size

SWA window size.

- Precision: `Unsigned integer`
- Unit: `Averaging samples`
- Minimum: `0`
- Default: `5`

### Example

```
[Measurement]
qoe_swa_window_size=2
```

### 4.5.5.6.4. qoe_wma_weight_newest

- Weight of the newest sample in WMA.
- Precision: Real number
  - Minimum: `0.0`
  - Default: `0.5`

### Example

```
[Measurement]
qoe_wma_weight_newest=1.0
```

### 4.5.5.7. Results

### 4.5.5.7.1. get_secondary_probe_average_results

Determines whether single-point average statistics are gathered from the secondary Probe.

- Values:
  - `true` - Gather average results from the secondary Probe
  - `false` - Do not gather average results from the secondary Probe
- Default: `false`

> 📑 This setting is relevant only when write_average_results is enabled.

### 4.5.5.7.2. use_results_distribution

Enable or disable result distribution directly from primary Probe to external result receivers.

- Values:
  - `true` - Enable result distribution
  - `false` - Disable result distribution
- Default: `false`

### 4.5.5.7.3. results_distribution_destinations

Qosium Probe can send measurement results to additional receivers during measurement. These receivers must be running the Qosium server, such as Qosium Storage.

- Type: `Array`
- Fields:
  - `address` The IPv4 address of the receiver
  - `port` The port number of the receiver

**Example**

To send Qosium results to destinations *127.0.0.1:7700* and *192.168.1.3:7710*, define this parameter as:

```
[Measurement]
use_results_distribution=true
results_distribution_destinations/size=2
results_distribution_destinations/1/address=127.0.0.1
results_distribution_destinations/1/port=7700
results_distribution_destinations/2/address=192.168.1.3
results_distribution_destinations/2/port=7710
```

### 4.5.5.7.4. write_absolute_results

When true, Packet QoS measurement results are written to file. Two files are generated, and the filenames have format *pk_qosDL[suffix].txt* and *pk_qosUL[suffix].txt*, and new measurements are appended to the files.

- Values:
  - `true` - Results are written to files
  - `false` - Results are not written to files
- Default: `false`

**Example**

```
[Measurement]
write_absolute_results=true
```

### 4.5.5.7.5. write_average_results

When true, average measurement results are written to file. The filename has the format
"averages_[suffix].txt", and new measurements are appended to the file.

- Values:
  - `true` - Results are written to file
  - `false` - Results are not written to file
- Default: `false`

**Example**

```
[Measurement]
write_average_results=true
```

### 4.5.5.7.6. write_date_code_format

Date code format governs the frequency of file creation when `write_multiple_files`. Whenever Scopemon
detects a change in the date code, it automatically triggers new result files. A timestamp with this date
code is then appended to the filename.

- Type: `string`
- Default: `yyyyMMdd`

**Example**
To write results every hour, define this parameter as:

```
[Measurement]
write_multiple_files=true
write_date_code_format=yyyyMMdd-hh
```

### 4.5.5.7.7. write_filename_suffix

File suffix string when forming a filename for measurement result files.

- Type: `string`
- Default: Empty

**Example**
If defined for example as "test", filenames will begin with the suffix and underscore, e.g. *averages_test.txt*.

```
[Measurement]
write_average_results=true
write_filename_suffix=test
```

> 📄 This settings has effect only when write_absolute_results, write_average_results, write_flow_results, and/or write_packet_results is set to true.

### 4.5.5.7.8. write_flow_results

When true, flow measurement results are written to file. The filename has the format "flows_[suffix].txt", and new measurements are appended to the file. This data only contains flow map detected during the measurement, which starts later than the actual flow. Therefore expect the reported flow duration to be shorter. Typically only one flow should be visible.

- Values:
  - `true` - Results are written to file
  - `false` - Results are not written to file
- Default: `false`

#### Example

```
[Measurement]
write_flow_results=true
```

### 4.5.5.7.9. write_multiple_files

When true, measurement results are written to multiple files. By default, one file is created for each day. For configuring multiple file writing frequency, see write_date_code_format.

- Values:
  - `true` - Results are written to multiple files
  - `false` - All results are written into a single file
- Default: `false`

#### Example

```
[Measurement]
write_average_results=true
write_flows=true
write_multiple_files=true
```

> 📄 This settings has effect only when write_absolute_results, write_average_results, write_flow_results, and/or write_packet_results is set to true.

### 4.5.5.7.10. write_packet_results

When true, packet measurement results are written to file. The filename has the format "pk_info[suffix].txt", and new measurements are appended to the file.

- Values:
  - `true` - Results are written to file
  - `false` - Results are not written to file
- Default: `false`

## Example

```
[Measurement]
write_packet_results=true
```

### 4.5.5.7.11. write_path

Set to override the path where measurement result files are stored. Use `/` as the directory separator.

- Type: `string`
- Default: Scopemon root directory

## Example

```
[Measurement]
write_path=c:/temp
```

> 📝 This settings has effect only when write_absolute_results, write_average_results, write_flow_results, and/or write_packet_results is set to true.

## 4.5.5.8. Measurement Group (GQoSM)

Scopemon is configured via parameters. This reference lists all available parameters, default values, allowed values, and examples for GQoSM configuration.

### 4.5.5.8.1. qoe_gqosm_cbl_form

The form factor for connection break length.

- Precision: `float`
- Minimum: `0.0`
- Maximum: `1.0`
- Default: `0.3`

> 📝 Ignored if use_qoe_gqosm_cbl is false

## Example

```
[Measurement]
use_qoe_gqosm_cbl=true
qoe_gqosm_cbl_form=0.4
```

### 4.5.5.8.2. qoe_gqosm_cbl_threshold

Bad performance limit for connection break length.

- Unit: `packets`
- Precision: `integer`
- Minimum: `0`
- Default: `5`

> 📋 Ignored if use_qoe_gqosm_cbl is false

### Example

```
[Measurement]
use_qoe_gqosm_cbl=true
qoe_gqosm_cbl_threshold=2
```

### 4.5.5.8.3. qoe_gqosm_delay_form

The form factor for the delay.

- Precision: `float`
- Minimum: `0.0`
- Maximum: `1.0`
- Default: `0.3`

> 📋 Ignored if use_qoe_gqosm_delay is false

### Example

```
[Measurement]
use_qoe_gqosm_delay=true
qoe_gqosm_delay_form=0.4
```

### 4.5.5.8.4. qoe_gqosm_delay_threshold

Bad performance limit for the delay.

- Unit: `seconds`
- Precision: `float`
- Minimum: `0.0`
- Default: `0.3`

> 📋 Ignored if use_qoe_gqosm_delay is false

### Example

```
[Measurement]
use_qoe_gqosm_delay=true
qoe_gqosm_delay_threshold=0.5
```

### 4.5.5.8.5. qoe_gqosm_jitter_form

The form factor for jitter.

- Precision: `float`
- Minimum: `0.0`
- Maximum: `1.0`

- Default: `0.3`

> 📋 Ignored if use_qoe_gqosm_jitter is false

### Example

```
[Measurement]
use_qoe_gqosm_jitter=true
qoe_gqosm_jitter_form=0.1
```

## 4.5.5.8.6. qoe_gqosm_jitter_threshold

Bad performance limit for jitter.

- Unit: `seconds`
- Precision: `float`
- Minimum: `0.0`
- Default: `0.1`

> 📋 Ignored if use_qoe_gqosm_jitter is false

### Example

```
[Measurement]
use_qoe_gqosm_jitter=true
qoe_gqosm_jitter_threshold=0.07
```

## 4.5.5.8.7. qoe_gqosm_packet_loss_form

The form factor for packet loss.

- Precision: `float`
- Minimum: `0.0`
- Maximum: `1.0`
- Default: `0.3`

> 📋 Ignored if use_qoe_gqosm_packet_loss is false

### Example

```
[Measurement]
use_qoe_gqosm_packet_loss=true
qoe_gqosm_packet_loss_form=0.4
```

## 4.5.5.8.8. qoe_gqosm_packet_loss_threshold

Bad performance limit for packet loss.

- Unit: `fraction`

- Precision: `float`
- Minimum: `0.0`
- Maximum: `1.0`
- Default: `0.03`

> 📝 Ignored if use_qoe_gqosm_packet_loss is false

## Example

```
[Measurement]
use_qoe_gqosm_packet_loss=true
qoe_gqosm_packet_loss_threshold=0.08
```

### 4.5.5.8.9. use_qoe_gqosm

Determines if GQoSM quality model calculations are performed. When false, GQoSM results are not available in results.

- Values:
  - `true` - GQoSM quality model is calculated
  - `false` - GQoSM quality model is not calculated
- Default: `true`

## Example

```
[Measurement]
use_qoe_gqosm=true
```

### 4.5.5.8.10. use_qoe_gqosm_cbl

Determines if connection break length parameter should be used in quality assessment.

- Values:
  - `true` - Parameter is used
  - `false` - Parameter is not used
- Default: `false`

## Example

```
[Measurement]
use_qoe_gqosm_cbl=true
```

### 4.5.5.8.11. use_qoe_gqosm_delay

Determines if delay parameter should be used in quality assessment.

- Values:
  - `true` - Parameter is used
  - `false` - Parameter is not used

- Default: `true`

### Example

```
[Measurement]
use_qoe_gqosm_delay=false
```

### 4.5.5.8.12. use_qoe_gqosm_jitter

Determines if jitter parameter should be used in quality assessment.

- Values:
    - `true` - Parameter is used
    - `false` - Parameter is not used
- Default: `true`

### Example

```
[Measurement]
use_qoe_gqosm_jitter=false
```

### 4.5.5.8.13. use_qoe_gqosm_packet_loss

Determines if packet loss parameter should be used in quality assessment.

- Values:
    - `true` - Parameter is used
    - `false` - Parameter is not used
- Default: `true`

### Example

```
[Measurement]
use_qoe_gqosm_packet_loss=false
```

## 4.5.5.9. Measurement Group (PSQA)

Scopemon is configured via parameters. This reference lists all available parameters, default values, allowed values, and examples for PSQA configuration.

### 4.5.5.9.1. qoe_psqa_codec_list

The audio codec for VoIP listening mode.

- Values:
    - `0` PCM - PCM codec is used
    - `1` GSM - GSM codec is used
- Default: `0`

### Example

```
[Measurement]
qoe_psqa_codec_list=1
```

### 4.5.5.9.2. qoe_psqa_fec_conv

Forward error correction for VoIP conversational mode.

- Values:
  - 0 Off - FEC is off
  - 10 Low - FEC is low
  - 11 High - FEC is high
- Default: 0

#### Example

```
[Measurement]
qoe_psqa_fec_conv=0
```

### 4.5.5.9.3. qoe_psqa_fec_list

Forward error correction offset for VoIP listening mode.

- Values:
  - 0 Off - FEC is off
  - 1 FEC 1 - FEC offset = 1.0
  - 2 FEC 2 - FEC offset = 2.0
  - 3 FEC 3 - FEC offset = 3.0
- Default: 0

#### Example

```
[Measurement]
qoe_psqa_fec_list=2
```

### 4.5.5.9.4. qoe_psqa_mode

The neural network model of PSQA.

- Values:
  - 1 VoIP listening - PSQA is used for VoIP in listening mode
  - 2 VoIP conversational - PSQA is used for VoIP in conversational mode
  - 3 Video AV - Video in streaming audiovisual mode
  - 4 Video AV MLP - Video in streaming audiovisual mode (MPL)
- Default: 1

#### Example

```
[Measurement]
qoe_psqa_mode=1
```

### 4.5.5.9.5. qoe_psqa_pi

Packetization interval for VoIP listening mode.

- Values:
  - 0 20 ms - Packetization interval is 20 ms
  - 1 40 ms - Packetization interval is 40 ms
  - 2 80 ms - Packetization interval is 80 ms
- Default: 0

**Example**

```
[Measurement]
qoe_psqa_pi=2
```

### 4.5.5.9.6. qoe_psqa_speex_rate

Speex codec data rate for VoIP conversational mode.

- Values:
  - 10 2.4 - Speex rate: 2.4 kbit/s
  - 11 4.0 - Speex rate: 4.0 kbit/s
  - 12 6.0 - Speex rate: 6.0 kbit/s
  - 13 8.0 - Speex rate: 8.0 kbit/s
  - 14 11.2 - Speex rate: 11.2 kbit/s
  - 15 14.2 - Speex rate: 14.2 kbit/s
  - 16 18.4 - Speex rate: 18.4 kbit/s
  - 17 24.8 - Speex rate: 24.8 kbit/s
- Default: 10

**Example**

```
[Measurement]
qoe_psqa_codec_conv=11
```

### 4.5.5.9.7. qoe_psqa_video_resolution

The resolution of the video content.

- Values:
  - 0 480p - Video resolution is 480p
  - 1 720p - Video resolution is 720p
  - 2 1080p - Video resolution is 1080p
- Default: 0

### Example

```
[Measurement]
qoe_psqa_video_resolution=1
```

## 4.5.5.9.8. qoe_psqa_video_motion

The amount of motion in the video content.

- Values:
    - `0` Low - Low motion (e.g., news)
    - `1` Moderate - Moderate motion (e.g., typical TV shows)
    - `2` High - High motion (e.g., sports)
- Default: `0`

### Example

```
[Measurement]
qoe_psqa_video_motion=1
```

## 4.5.5.9.9. qoe_psqa_video_ec

The error concealment mode of the video content (valid only for the Video AV MLP mode).

- Values:
    - `0` Off - Error concealment is off
    - `1` On - Error concealment is on
- Default: `0`

### Example

```
[Measurement]
qoe_psqa_video_ec=1
```

## 4.5.5.9.10. qoe_psqa_video_cmq

Calculated movement quantity of the video content (valid only for the Video AV MLP mode).

- Unit: `percentage`
- Precision: `float`
- Minimum: `0.0`
- Maximum: `100.0`
- Default: `0.0`

### Example

```
[Measurement]
qoe_psqa_video_cmq=1.15
```

### 4.5.5.9.11. use_qoe_psqa

Determines if PSQA quality model calculations are performed. When false, PSQA results are not available in results.

- Values:
  - `true` - PSQA quality model is calculated
  - `false` - PSQA quality model is not calculated
- Default: `false`

#### Example

```
[Measurement]
use_qoe_psqa=true
```

## 4.5.6. QoEChart Group

QoEChart contains configuration options for Quality of Experience visualization.

### 4.5.6.1. mode

Select whether the chart uses GQoSM or PSQA as the source data.

- Values:
  - `0` GQoSM - Use GQoSM as the source model
  - `1` PSQA- Use PSQA as the source model
- Default: `0`

#### Example
To use PSQA as the source model, define this parameter as:

```
[QoEChart]
mode=1
```

### 4.5.6.2. received_lower_threshold

Defines the lower threshold for received quality.

- Unit: `MOS`
- Precision: `float`
- Minimum: `1.0`
- Maximum: `5.0`
- Default: `0.0`

#### Example
To set the lower threshold to 2.0 MOS, define this parameter as:

```
[QoEChart]
received_lower_threshold=2.0
```

### 4.5.6.3. received_lower_alert

When enabled, an alert is emitted whenever the received quality falls below `received_lower_threshold`.

- Values:
    - `true` - An alert is emitted
    - `false` - No alert is emitted
- Default: `false`

### 4.5.6.4. sent_lower_threshold

Defines the lower threshold for sent quality.

- Unit: `MOS`
- Precision: `float`
- Minimum: `1.0`
- Maximum: `5.0`
- Default: `0.0`

**Example**

To set the lower threshold to 2.0 MOS, define this parameter as:

```
[QoEChart]
sent_lower_threshold=2.0
```

### 4.5.6.5. sent_lower_alert

When enabled, an alert is emitted whenever the sent quality falls lower than `sent_lower_threshold`.

- Values:
    - `true` - An alert is emitted
    - `false` - No alert is emitted
- Default: `false`

### 4.5.6.6. visible

Determines if this visualizer is displayed in the GUI.

- Values:
    - `true` - Visualizer is displayed
    - `false` - Visualizer is not displayed
- Default: `false`

**Example**

To display the visualizer, define this parameter as:

```
[QoEChart]
visible=true
```

## 4.5.7. ScheduledMeasurer Group

Scopemon is configured via parameters. This reference lists all available parameters, default values, allowed values, and examples for measurer ScheduledMeasurer.

### 4.5.7.1. schedule_rules

Schedule rules determine when a measurement commences. When the current system time meets all rule requirements, measurement is started. The measurement persists until the current system clock time no longer meets the set of rules.

Schedule rules is a list of rule entries. Scopemon supports a time range rule, which defines a timespan when a measurement should be performed.

- Type: `array`
- Fields:

  - `start_time` The time when the schedule is turned on
  - `end_time` The time when the schedule is turned off

**Example**
To run measurement every day between 8am and 9am, and between 2pm and 9:30pm, define the list as:

```
[ScheduledMeasurer]
schedule_rules\size=2
schedule_rules\1\start_time=8:00:00
schedule_rules\1\end_time=9:00:00
schedule_rules\2\start_time=14:00:00
schedule_rules\2\end_time=21:30:00
```

## 4.5.8. ThroughputChart Group

ThroughputChart contains configuration options for throughput visualization.

### 4.5.8.1. received_lower_threshold

Defines the lower threshold for received traffic.

- Unit: `bits per second`
- Precision: `integer`
- Minimum: `0`
- Default: `0`

**Example**
To set the lower threshold to 1 kbps (= 1000 bps), define this parameter as:

```
[ThroughputChart]
received_lower_threshold=1000
```

### 4.5.8.2. received_lower_alert

When enabled, an alert is emitted whenever the received traffic load falls lower than `received_lower_threshold`.

- Values:
  - `true` - An alert is emitted
  - `false` - No alert is emitted
- Default: `false`

**Example**

To emit an alert when received traffic falls below 1 kbps, define this parameter as:

```
[ThroughputChart]
received_lower_threshold=1000
received_lower_alert=true
```

### 4.5.8.3. received_upper_threshold

Defines the upper threshold for received traffic.

- Unit: `bits per second`
- Precision: `integer`
- Minimum: `0`
- Default: `1000000`

**Example**

To set the upper threshold to 5 Mbps (= 5000000 bps), define this parameter as:

```
[ThroughputChart]
received_upper_threshold=5000000
```

### 4.5.8.4. received_upper_alert

When enabled, an alert is emitted whenever the received traffic load exceeds `received_upper_threshold`.

- Values:
  - `true` - An alert is emitted
  - `false` - No alert is emitted
- Default: `false`

**Example**

To emit an alert when received traffic exceeds 5 Mbps, define this parameter as:

```
[ThroughputChart]
received_upper_threshold=5000000
received_upper_alert=true
```

### 4.5.8.5. sent_lower_threshold

Defines the lower threshold for sent traffic.

- Unit: `bits per second`
- Precision: `integer`
- Minimum: `0`
- Default: `0`

**Example**

To set the lower threshold to 1 kbps (= 1000 bps), define this parameter as:

```
[ThroughputChart]
sent_lower_threshold=1000
```

### 4.5.8.6. sent_lower_alert

When enabled, an alert is emitted whenever the sent traffic load falls lower than `sent_lower_threshold`.

- Values:
  - `true` - An alert is emitted
  - `false` - No alert is emitted
- Default: `false`

**Example**

To emit an alert when sent traffic falls below 1 kbps, define this parameter as:

```
[ThroughputChart]
sent_lower_threshold=1000
sent_lower_alert=true
```

### 4.5.8.7. sent_upper_threshold

Defines the upper threshold for sent traffic.

- Unit: `bits per second`
- Precision: `integer`
- Minimum: `0`
- Default: `1000000`

**Example**

To set the upper threshold to 5 Mbps (= 5000000 bps), define this parameter as:

```
[ThroughputChart]
sent_upper_threshold=5000000
```

### 4.5.8.8. sent_upper_alert

When enabled, an alert is emitted whenever the sent traffic load exceeds `sent_upper_threshold`.

- Values:
  - `true` - An alert is emitted

- `false` - No alert is emitted
- Default: `false`

### Example
To emit an alert when sent traffic exceeds 5 Mbps, define this parameter as:

```
[ThroughputChart]
sent_upper_threshold=5000000
sent_upper_alert=true
```

## 4.5.8.9. visible
Determines if this visualizer is displayed in the GUI.

- Values:
  - `true` - Visualizer is displayed
  - `false` - Visualizer is not displayed
- Default: `false`

### Example
To display the visualizer, define this parameter as:

```
[ThroughputChart]
visible=true
```

# 5. Glossary

## Graphical User Interface

*User interface which features graphical elements, such as windows, buttons, dialogs, images, and controls.*

## Network Address Translation

*A technique for remapping an IP address space*

[Wikipedia article on Network Address Translation](#)

## User Datagram Protocol

*A simple, fast, unreliable transport protocol.*

## Real-time Transport Protocol

*A transport protocol for applications with real-time constraints, such as video streams, VoIP, and remote control.*