Measurement Group

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

Table of Contents

1. General ... 4
1.1. measurement_description ... 4
1.2. measurement_start_offset ... 4
1.3. reconnect_interval ... 4
1.4. robust_mode_max_cbd ... 5
1.5. user_id ... 5
2. Topology ... 5
2.1. primary_probe_hostname ... 5
2.2. primary_probe_port ... 6
2.3. use_secondary_probe ... 6
2.4. secondary_probe_hostname ... 6
2.5. secondary_probe_port ... 7
2.6. primary_probe_placement ... 7
2.7. primary_probe_interface_index ... 7
2.8. secondary_probe_interface_index ... 8
2.9. secondary_probe_placement ... 8
2.10. nat_between_probes ... 9
3. Filtering ... 9
3.1. packet_filter_mode ... 9
3.2. packet_filter ... 10
4. Senders ... 10
4.1. primary_probe_senders_pure_mac_method_enabled ... 10
4.2. primary_probe_senders_eth_mode ... 11
4.3. primary_probe_senders_eth_address_list ... 11
4.4. primary_probe_senders_ipv4_mode ... 11
4.5. primary_probe_senders_ipv4_address_list ... 12
4.6. primary_probe_senders_ipv6_mode ... 12
4.7. primary_probe_senders_ipv6_address_list ... 12
4.8. secondary_probe_senders_pure_mac_method_enabled ... 12
4.9. secondary_probe_senders_eth_mode ... 13
4.10. secondary_probe_senders_eth_address_list ... 13
4.11. secondary_probe_senders_ipv4_mode ... 13
4.12. secondary_probe_senders_ipv4_address_list ... 14
4.13. secondary_probe_senders_ipv6_mode ... 14
4.14. secondary_probe_senders_ipv6_address_list ... 15
5. Measurement Details ... 15
5.1. averaging_interval ... 15
5.2. packet_id_method ... 15
5.3. packet_loss_timer ... 16
5.4. pk_delay_threshold ... 16
5.5. pk_jitter_threshold ... 17
5.6. use_promiscuous_mode ... 17
6. QoE Averaging ... 17
6.1. use_qoe_swa ... 17
6.2. use_qoe_wma ... 18
6.3. qoe_swa_window_size ... 18
6.4. qoe_wma_weight_newest ... 18

7. Results 19
7.1. get_average_results 19
7.2. get_flow_results 19
7.3. get_packet_results 19
7.4. get_secondary_probe_average_results 19
7.5. use_results_distribution 20
7.6. results_distribution_destinations 20
7.7. write_average_results 20
7.8. write_flow_results 21
7.9. write_packet_results 21
7.10. write_absolute_results (deprecated) 21
7.11. write_multiple_files 22
7.12. write_date_code_format 22
7.13. write_filename_suffix 22
7.14. write_path 23
8. Glossary 24

1. General

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
```

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

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
```

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
```

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
```

2. Topology

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
```

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
```

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.

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

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

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`

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
```

To see the available interfaces and their indices start Scopemon and check its log.

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

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

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

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.

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.

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. 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.

4.1. 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.

4.2. 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
```

4.3. 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
```

4.4. 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. 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
```

4.6. 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.

4.7. 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
```

4.8. 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

4.9. 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

4.10. 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

4.11. 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

4.12. 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

4.13. 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

4.14. 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

5. Measurement Details

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`

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
```

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
```

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).

If your measured application has strict delay limits that you wish to take into account in the measurement, do not try to use this parameter to turn delayed packets into packet loss. Instead, use pk_delay_threshold parameter to calculate exactly the number of packets that experience higher delay than the set threshold.

5.4. pk_delay_threshold

Packets with a delay above this threshold are counted in [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
```

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
```

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
```

6. QoE Averaging

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

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
```

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
```

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
```

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
```

7. Results

7.1. get_average_results

Enable reception of average results during measurement.

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

This setting was introduced in Scopemon version of 1.6.0.0->.

7.2. get_flow_results

Enable reception of flow results during measurement.

- Values:
  - `true` - Gather flow results from the primary Probe
  - `false` - Do not gather flow results
- Default: `false`

This setting was introduced in Scopemon version of 1.6.0.0->.

7.3. get_packet_results

Enable reception of packet results during measurement. In a single-point measurement, the results include information of every packet matching the measurement filter. In a two-point measurement, also the QoS statistics (delay and jitter) are received for every single packet matching the measurement filter.

- Values:
  - `true` - Gather packet results
  - `false` - Do not gather packet results
- Default: `false`

This setting was introduced in Scopemon version of 1.6.0.0->. A lot of results data is received when measuring traffic with a high data rate.

7.4. get_secondary_probe_average_results

Enable reception of single-point average statistics from the secondary Probe during measurement.

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

7.5. 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`

7.6. 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
```

7.7. write_average_results

When true, average measurement results are written in a file. The filename has the format “averages_[suffix].txt”, and new measurements are appended to the file. This setting overrides get_average_results.

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

Example

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

7.8. write_flow_results

When true, flow measurement results are written in a file. The filename has the format “flows_[suffix].txt”; new measurements are appended to the file. This data only contains the flow map detected during the measurement. This setting overrides get_flow_results.

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

Example

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

7.9. write_packet_results

When true, packet measurement results are written in a file. In the single-point measurement scenario, the filename has the format “pk_info[suffix].txt”. In a two-point measurement, two files are generated, and the filenames have the format *pk_qosDL[suffix].txt* and *pk_qosUL[suffix].txt*. Results from new measurements are appended to the file. This setting overrides get_packet_results.

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

Example

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

In Scopemon versions of 1.6.0.0->, this setting also comprises the deprecated *write_absolute_results* .

7.10. write_absolute_results (deprecated)

When true, Packet QoS measurement results are written in a 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 in a file(s)
  - `false` - Results are not written in a file(s)

- Default: `false`

Example

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

This setting is deprecated in Scopemon versions of 1.6.0.0-> and is replaced by the setting *write_packet_results *.

7.11. 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 in multiple files
  - `false` - All results are written in 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.

7.12. 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:

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

7.13. 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.

7.14. 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.

8. Glossary

Network Address Translation

A technique for remapping an IP address space

Wikipedia article on Network Address Translation

User Datagram Protocol

A simple, fast, and connectionless transport protocol.

Real-time Transport Protocol

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