NAME

qof − Quality of Flow (yet another yet another flowmeter)

SYNOPSIS

```
qof     [--in LIBTRACE_URI] [--out OUTPUT_SPECIFIER]
        [--yaml CONFIG_FILE]
        [--filter BPF_FILTER]
        [--rotate ROTATE_DELAY] [--lock]
        [--stats INTERVAL]
        [--observation-domain DOMAIN_ID]
        [--ipfix TRANSPORT_PROTOCOL]
        [--ipfix-port PORT] [--tls] [--tls-ca CA_PEM_FILE]
        [--tls-cert CERT_PEM_FILE] [--tls-key KEY_PEM_FILE]
        [--template-refresh TEMPLATE_TIMEOUT]
        [--become-user UNPRIVILEGED_USER]
        [--become-group UNPRIVILEGED_GROUP]
        [--log LOG_SPECIFIER] [--loglevel LOG_LEVEL]
        [--verbose] [--version]
```

DESCRIPTION

qof is an IPFIX (**RFC 7011**) flow meter with an emphasis on passive TCP performance measurement. It is based on a fork of the **yaf** flow meter. It reads packet data using **libtrace**, which allows it to read from many dumpfile formats as well as a variety of capture devices, generates flows from these packets, and exports flow records via IPFIX over SCTP, TCP or UDP, or into **RFC 5655** IPFIX Files on the local file system.

Since **qof** is designed to be deployed on white-box sensors attached to local network segments or span ports at symmetric routing points, it supports bidirectional flow assembly natively. Biflow export is done via the export method specified in **RFC 5103** Bidirectional Flow Export using IPFIX. See the **OUTPUT** section below for information on this format.

As of this release, **qof** should be considered alpha quality software. The core is reasonably stable, having been based on **yaf**, but the command-line options, configuration interface, and output format may change wildly before a 1.0 release, and the correctness of the output has not yet been rigorously established by controlled testing.

OPTIONS

Input Options

These options control where **qof** will take its input from.

−−in *LIBTRACE_URI*

LIBTRACE_URI is a libtrace URI. If no scheme is given, assumes `pcapfile:` to read from a named pcap dumpfile. If not given, reads pcap dumpfiles from standard input.

−−filter *BPF_FILTER*

If present, enable Berkeley Packet Filtering (BPF) in **qof** with *FILTER_EXPRESSION* as the incoming traffic filter. The syntax of *FILTER_EXPRESSION* follows the expression format described in the ***tcpdump*** **(1)** man page.

Output Options

These options control where **qof** will send its output. **qof** can write flows to an IPFIX file or export flows to an IPFIX collector over SCTP, TCP, UDP, or Spread. By default, if no output options are given, **qof** writes an IPFIX file to standard output.

−−out *OUTPUT_SPECIFIER*

OUTPUT_SPECIFIER is an output specifier. If **−−ipfix** is present, the *OUTPUT_SPECIFIER* specifies the hostname or IP address of the collector to which the flows will be exported. Otherwise, *OUTPUT_SPECIFIER* is a filename in which the flows will be written; the string **−** may be used to write to standard output (the default).

−−ipfix *TRANSPORT_PROTOCOL*
If present, causes **qof** to operate as an IPFIX exporter, sending IPFIX Messages via the specified transport protocol to the collector (e.g., SiLK's rwflowpack or flowcap facilities) named in the *OUTPUT_SPECIFIER*. Valid *TRANSPORT_PROTOCOL* values are **tcp**, **udp** or **sctp**. **sctp** is only available if **qof** was built with SCTP support. UDP is not recommended, as it is not a reliable transport protocol, and cannot guarantee delivery of messages. As per the recommendations in RFC 5101, **qof** will retransmit templates three times within the template timeout period (configurable using **−−template−refresh**). Use the **−−ipfix−port**, **−−tls**, **−−tls−ca**, **−−tls−cert**, and **−−tls−key** options to further configure the connection to the IPFIX collector.

−−rotate *ROTATE_DELAY*
If present, causes **qof** to write output to multiple files, opening a new output file every *ROTATE_DELAY* seconds in the input data. Rotated files are named using the prefix given in the *OUTPUT_SPECIFIER*, followed by a suffix containing a timestamp in `YYYYMMDDhhmmss` format, a decimal serial number, and the file extension **.yaf**.

−−lock
Use lockfiles for concurrent file access protection on output files. This is recommended for interoperating with the Airframe filedaemon facility.

−−stats *INTERVAL*
If present, causes **qof** to export process statistics in an IPFIX Options Record inline in the output stream every *INTERVAL* seconds. If *INTERVAL* is set to zero, stats will not be exported; the default is not to export stats.

IPFIX Connection Options

These options are used to configure the connection to an IPFIX collector.

−−ipfix−port *PORT*
If **−−ipfix** is present, export flows to TCP, UDP, or SCTP port *PORT*. If not present, the default IPFIX port 4739 is used. If **−−tls** is also present, the default secure IPFIX port 4740 is used.

−−tls
If **−−ipfix** is present, use TLS to secure the connection to the IPFIX collector. Requires the *TRANSPORT_PROTOCOL* to be **tcp**, as DTLS over UDP or SCTP is not yet supported. Requires the **−−tls−ca**, **−−tls−cert**, and **−−tls−key** options to specify the X.509 certificate and TLS key information.

−−tls−ca *CA_PEM_FILE*
Use the Certificate Authority or Authorities in *CA_PEM_FILE* to verify the remote IPFIX Collecting Process' X.509 certificate. The connection to the Collecting Process will fail if its certificate was not signed by this CA (or by a certificate signed by this CA, recursively); this prevents export to unauthorized Collecting Processes. Required if **−−tls** is present.

−−tls−cert *CERT_PEM_FILE*
Use the X.509 certificate in *CERT_PEM_FILE* to identify this IPFIX Exporting Process. This certificate should contain the public part of the private key in *KEY_PEM_FILE*. Required if **−−tls** is present.

−−tls−key *KEY_PEM_FILE*
Use the private key in *KEY_PEM_FILE* for this IPFIX Exporting Process. This key should contain the private part of the public key in *CERT_PEM_FILE*. Required if **−−tls** is present. If the key is encrypted, the password must be present in the QOF_TLS_PASS environment variable.

−−observation−domain *ODID*
Set observation domain on exported IPFIX messages to *ODID*; the default is 0.

−−template−refresh *TEMPLATE_RTX_TIME*
Set UDP template refresh time in seconds. When set, every active template will be transmitted each *TEMPLATE_RTX_TIME* seconds. Defaults to 0 (no template retransmit) unless **−−ipfix udp** is present, in which case the default is 60 (1 minute).

Privilege Options

These options are used to cause **qof** to drop privileges when running as root for live capture purposes.

−−become−user *UNPRIVILEGED_USER*
: After opening inputs, drop privilege to the named user. Using **−−become−user** requires **qof** to be run as root or setuid root. This option will cause all files written by **qof** to be owned by the user *UNPRIVILEGED_USER* and the user's primary group; use **−−become−group** as well to change the group **qof** runs as for output purposes. If running as root for live capture purposes and **−−become−user** is not present, **qof** will warn that privilege is not being dropped. We highly recommend the use of this option, especially in production environments, for security purposes.

−−become−group *UNPRIVILEGED_GROUP*
: **−−become−group** can be used to change the group from the default of the user given in **−−become−user**. This option has no effect if given without the **−−become−user** option as well.

Logging Options

These options are used to specify how log messages are routed. **qof** can log to standard error, regular files, or the UNIX syslog facility.

−−log *LOG_SPECIFIER*
: Specifies destination for log messages. *LOG_SPECIFIER* can be a *syslog*(3) facility name, the special value **stderr** for standard error, or the *absolute* path to a file for file logging. The default log specifier is **stderr** if available, **user** otherwise.

−−loglevel *LOG_LEVEL*
: Specify minimum level for logged messages. In increasing levels of verbosity, the supported log levels are **quiet**, **error**, **critical**, **warning**, **message**, **info**, and **debug**. The default logging level is **warning**.

−−verbose
: Equivalent to **−−loglevel debug**.

−−version
: If present, print version and copyright information to standard error and exit.

CONFIGURATION FILE FORMAT

The behavior of the **qof** flow table and IPFIX export is defined by a YAML configuration file, the path to which is given on the **qof** command line by the **−−yaml** option. If no configuration file is given, it is as if an empty configuration file is given, and the default value for each of the configuration keys is taken.

The YAML configuration file is a mapping of configuration keys to values; the configuration keys are as follows:

template: *TEMPLATE_LIST*
: A list of Information Elements to export. The selection of Information Elements in this Template determines which **qof** features will be enabled; see the **OUTPUT** section for details. When no template is given, **qof** defaults to the following:

- flowStartMilliseconds
- flowEndMilliseconds
- octetDeltaCount
- reverseOctetDeltaCount
- packetDeltaCount
- reversePacketDeltaCount
- sourceIPv4Address
- destinationIPv4Address
- sourceIPv6Address

- destinationIPv6Address
- sourceTransportPort
- destinationTransportPort
- protocolIdentifier
- flowEndReason

See the **OUTPUT** section for a definition of each information element available here, and for the features enabled by each information element.

idle-timeout: *IDLE_TIMEOUT*
Set flow idle timeout in seconds. Flows are considered idle and flushed from the flow table if no packets are received for *IDLE_TIMEOUT* seconds. The default flow idle timeout is 30 seconds.

active-timeout: *ACTIVE_TIMEOUT*
Set flow active timeout in seconds. Any flow lasting longer than *ACTIVE_TIMEOUT* seconds will be flushed from the flow table. The default flow active timeout is 300 seconds (5 minutes).

max-flows: *FLOW_TABLE_MAX*
If present, limit the number of open flows in the flow table to *FLOW_TABLE_MAX* by prematurely expiring the flows with the least recently received packets; this is analogous to an adaptive idle timeout. This option is provided to limit **qof** resource usage when operating on data from large networks. By default, there is no flow table limit, and the flow table can grow to resource exhaustion.

max-frags: *FRAG_TABLE_MAX*
If present, limit the number of outstanding, not-yet reassembled fragments in the fragment table to *FRAG_TABLE_MAX* by prematurely expiring fragments from the table. This option is provided to limit **qof** resource usage when operating on data from very large networks or networks with abnormal fragmentation. The fragment table may exceed this limit slightly due to limits on how often **qof** prunes the fragment table (every 5 seconds). By default, there is no fragment table limit, and the fragment table can grow to resource exhaustion.

force-biflow: *FLAG*
If present and *FLAG* is anything except ‘‘0’’, export reverse Information Elements for all flows, even those with only one observed direction.

gre-decap: *FLAG*
If present and *FLAG* is anything except ‘‘0’’, attempt to decode GRE version 0 encapsulated packets. Flows will be created from packets within the GRE tunnels. Undecodeable GRE packets will be dropped. Without this option, GRE traffic is exported as IP protocol 47 flows. This option is presently experimental.

silk-compatible: *FLAG*
If present and *FLAG* is anything except ‘‘0’’, export flows in ‘‘SiLK mode’’. This flag must be used when exporting to SiLK for it to collect TCP flow information. This also introduces the following incompatibilities with standard IPFIX export:

- octetDeltaCount and reverseOctetDeltaCount are clamped to 32 bits. Any packet that would cause either of these counters to overflow 32 bits will cause the flow to close with flowEndReason 0x02 (active timeout), and will become the first packet of a new flow. This is analogous to forcing an active timeout when the octet counters overflow.
- The high-order bit of the flowEndReason IE is set on any flow created on a counter overflow, as above.
- The high-order bit of the flowEndReason IE is set on any flow created on an active timeout.

Since this changes the semantics of the exported flowEndReason IE, it should only be used when generating flows and exporting to rwflowpack, flowcap, or writing files for processing with rwipfix2silk.

OUTPUT

qof's output consists of an IPFIX message stream. **qof** uses a variety of templates for IPFIX data records, derived from a template supplied by the user in the YAML configuration file. This section describes the information elements used in those templates. For further information about IPFIX as implemented by **qof**, see RFC 7011, RFC 7012, RFC 5103, and RFC 5655.

Flow Records

qof assigns information element numbers to reverse flow elements in biflow capture based on the standard IPFIX PEN 29305. This applies only for information elements registered with IANA that do not have a reverse information element already defined. **qof** additionally uses information elements registered with CERT (PEN 6871) for features inherited from **yaf** and elements registered under the trammell.ch PEN (PEN 35566) for **qof** specific features. For enterprise-specific information elements, the method described in RFC 5103 is used to calculate their reverse element identifier: bit fourteen is set to one in the IE field, (e.g. 16384 + the forward IE number.)

qof supports the following Information Elements in flow records, which can appear in the **template:** list in the YAML configuration file; in the list below, features enabled and conditions for export for each Information Element are noted.

Note as well that if *any* reverse information element is specified in the **template:** list, **qof** will export biflows as in RFC 5103; otherwise, it will export biflows as two separate records.

For IANA IEs, see http://www.iana.org/assignments/ipfix for details.

octetDeltaCount IANA IE 1
Can be exported for all flows.

reverseOctetDeltaCount RFC 5103 (PEN 29305) IE 1
Can be exported for all biflows.

packetDeltaCount IANA IE 2
Can be exported for all flows.

reversePacketDeltaCount RFC 5103 (PEN 29305) IE 2
Can be exported for all biflows.

protocolIdentifier IANA IE 4
Flow key, can be exported for all flows.

tcpControlBits IANA IE 6
Can be exported for TCP flows.

reverseTcpControlBits RFC 5103 (PEN 29305) IE 6
Can be exported for TCP biflows.

sourceTransportPort IANA IE 7
Flow key, can be exported for all flows.

sourceIPv4Address IANA IE 8
Flow key, can be exported for IPv4 flows. If neither sourceIPv4Address nor destinationIPv4Address are present in the template, and either sourceIPv6Address or destinationIPv6Address are present, then sourceIPv4Address will be exported as an IPv6−mapped address in sourceIPv6Address.

ingressInterface IANA IE 10
Can be exported if ingress interface information is available; presently only supported with the (not yet documented) interface map.

destinationTransportPort IANA IE 11
Flow key, can be exported for all flows.

destinationIPv4Address IANA IE 12
Flow key, can be exported for IPv4 flows. If neither sourceIPv4Address nor destinationIPv4Address are present in the template, and either sourceIPv6Address or destinationIPv6Address are present, then destinationIPv4Address will be exported as an IPv6−mapped address in destinationIPv6Address.

egressInterface IANA IE 14
: Can be exported if egress interface information is available; presently only supported with the (not yet documented) interface map.

sourceIPv6Address IANA IE 27
: Flow key, can be exported for IPv6 flows, or for IPv4 flows if address mapping is enabled. If neither sourceIPv6Address nor destinationIPv6Address are present in the template, then IPv6 flows will not be measured.

destinationIPv6Address IANA IE 28
: Flow key, can be exported for IPv6 flows, or for IPv4 flows if address mapping is enabled. If neither sourceIPv6Address nor destinationIPv6Address are present in the template, then IPv6 flows will not be measured.

minimumTTL IANA IE 52
: Can be exported for all flows.

reverseMinimumTTL RFC 5103 (PEN 29305) IE 52
: Can be exported for all biflows.

maximumTTL IANA IE 53
: Can be exported for all flows.

reverseMaximumTTL RFC 5103 (PEN 29305) IE 53
: Can be exported for all biflows.

sourceMacAddress IANA IE 56
: Can be exported for all flows if MAC layer information available; enables MAC header parsing if selected.

vlanId IANA IE 58
: Can be exported for all flows if MAC layer information available; enables MAC header parsing if selected.

destinationMacAddress IANA IE 80
: Can be exported for all flows if MAC layer information available; enables MAC header parsing if selected.

flowEndReason IANA IE 136
: Can be exported for all flows.

flowId IANA IE 148
: Can be exported for all flows. flowIds are persistent across exports on active timeout, and can be used to link flows for continuation.

flowStartMilliseconds IANA IE 152
: Can be exported for all flows.

flowEndMilliseconds IANA IE 153
: Can be exported for all flows.

transportOctetDeltaCount IANA IE 401
: Can be exported for all flows.

reverseTransportOctetDeltaCount RFC 5103 (PEN 29305) IE 401
: Can be exported for all biflows.

transportPacketDeltaCount IANA IE 402
: Can be exported for all flows.

reverseTransportPacketDeltaCount RFC 5103 (PEN 29305) IE 402
: Can be exported for all biflows.

initialTCPFlags CERT (PEN 6871) IE 14
(type: unsigned8, semantics: flags) Flags on the first TCP packet in the forward direction; see tcpControlBits.

unionTCPFlags CERT (PEN 6871) IE 15
(type: unsigned8, semantics: flags) Union of the flags on all TCP packets after the first TCP packet in the forward direction; see tcpControlBits.

reverseInitialTCPFlags CERT (PEN 6871) IE 16398
(type: unsigned8, semantics: flags) Flags on the first TCP packet in the reverse direction; see tcpControlBits.

reverseUnionTCPFlags CERT (PEN 6871) IE 16399
(type: unsigned8, semantics: flags) Union of the flags on all TCP packets after the first TCP packet in the reverse direction; see tcpControlBits.

reverseFlowDeltaMilliseconds CERT (PEN 6871) IE 21
(type: signed32, semantics: quantity, units: milliseconds) Milliseconds between first packet in the forward direction and the first packet in the reverse direction.

tcpSequenceCount trammell.ch (PEN 35566) IE 1024
(type: unsigned64, semantics: deltaCounter, units: octets) The number of octets passed up to the application layer at the receiver, as measured by progress through TCP sequence number space observed at the Observation Point. Only exported for TCP flows. If present in the template, enables sequence number tracking.

reverseTcpSequenceCount trammell.ch (PEN 35566) IE 17408
(type: unsigned64, semantics: deltaCounter, units: octets) The number of octets passed up to the application layer at the receiver in the reverse direction, as measured by progress through TCP sequence number space observed at the Observation Point. Only exported for TCP biflows. If present in the template, enables sequence number tracking.

tcpRetransmitCount trammell.ch (PEN 35566) IE 1025
(type: unsigned64, semantics: deltaCounter, units: octets) The number of packets consisting of complete or partial retransmission, as measured by segments and/or portions of sequence number space observed at the Observation Point. Only exported for TCP flows. If present in the template, enables sequence number tracking.

reverseTcpRetransmitCount trammell.ch (PEN 35566) IE 17409
(type: unsigned64, semantics: deltaCounter, units: octets) The number of packets in the reverse direction consisting of complete or partial retransmission, as measured by segments and/or portions of sequence number space observed at the Observation Point. Only exported for TCP biflows. If present in the template, enables sequence number tracking.

maxTcpSequenceJump trammell.ch (PEN 35566) IE 1026
(type: unsigned64, semantics: quantity, units: octets) The maximum observed size of a sequence number jump, calculated by subtracting the sequence number of the first byte in a jump packet from the highest-seen sequence number on this Flow. Only exported for TCP flows. If present in the template, enables sequence number tracking.

reverseMaxTcpSequenceJump trammell.ch (PEN 35566) IE 17410
(type: unsigned64, semantics: quantity, units: octets) The maximum observed size of a sequence number jump in the reverse direction, calculated by subtracting the sequence number of the first byte in a jump packet from the highest-seen sequence number on the reverse direction of this Flow. Only exported for TCP biflows. If present in the template, enables sequence number tracking.

minTcpRttMilliseconds trammell.ch (PEN 35566) IE 1029
(type: unsigned32, semantics: quantity, units: milliseconds) The minimum observed TCP round trip time for this Flow, estimated from the sender's point of view, observed at the Observation Point. Derived from SYN-ACK as well as TSOPT VAL-ECR delays on both directions of the Flow. Only exported for TCP biflows. If present in the template, enables RTT measurement and TCP options

parsing.

lastTcpRttMilliseconds trammell.ch (PEN 35566) IE 1030
(type: unsigned32, semantics: quantity, units: milliseconds) The final smoothed observed TCP round trip time for this Flow, estimated from the sender's point of view, observed at the Observation Point. Derived from SYN-ACK as well as TSOPT VAL-ECR delays on both directions of the Flow. Only exported for TCP biflows. If present in the template, enables RTT measurement and TCP options parsing.

declaredTcpMss trammell.ch (PEN 35566) IE 1033
(type: unsigned16, semantics: quantity, units: octets) TCP MSS declared in TCP options. Only exported for TCP flows. If present in the template, enables TCP options parsing.

reverseDeclaredTcpMss trammell.ch (PEN 35566) IE 17417
(type: unsigned16, semantics: quantity, units: octets) TCP MSS declared in TCP options in the reverse direction of the Flow. Only exported for TCP biflows. If present in the template, enables TCP options parsing.

observedTcpMss trammell.ch (PEN 35566) IE 1034
(type: unsigned16, semantics: quantity, units: octets) Maximum size of payload in observed TCP segments. Only exported for TCP flows. If present in the template, enables TCP options parsing.

reverseObservedTcpMss trammell.ch (PEN 35566) IE 17418
(type: unsigned16, semantics: quantity, units: octets) Maximum size of payload in observed TCP segments in the reverse direction of the Flow. Only exported for TCP biflows. If present in the template, enables TCP options parsing.

tcpSequenceLossCount trammell.ch (PEN 35566) IE 1035
(type: unsigned64, semantics: deltaCounter, units: octets) Octets in TCP segments inferred from observing the sequence number series but not observed in the packet stream. A nonzero value generally indicates observation loss. Only exported for TCP flows. If present in the template, enables sequence number tracking.

reverseTcpSequenceLossCount trammell.ch (PEN 35566) IE 17419
(type: unsigned64, semantics: deltaCounter, units: octets) Octets in TCP segments inferred from observing the sequence number series but not observed in the packet stream in the reverse direction. A nonzero value generally indicates observation loss. Only exported for TCP biflows. If present in the template, enables sequence number tracking.

tcpSequenceJumpCount trammell.ch (PEN 35566) IE 1036
(type: unsigned64, semantics: deltaCounter, units: packets) Number of packets observed which cause a sequence number jump, calculated by subtracting the sequence number of the first byte in a jump packet from the highest-seen sequence number on this Flow. Counts multi-segment jumps in units of observed MSS, Only exported for TCP flows. If present in the template, enables sequence number tracking.

reverseTcpSequenceJumpCount trammell.ch (PEN 35566) IE 17420
(type: unsigned64, semantics: deltaCounter, units: packets) Number of packets observed in the reverse direction which cause a sequence number jump, calculated by subtracting the sequence number of the first byte in a jump packet from the highest-seen sequence number on this Flow. Counts multi-segment jumps in units of observed MSS. Only exported for TCP biflows. If present in the template, enables sequence number tracking.

tcpLossEventCount trammell.ch (PEN 35566) IE 1038
(type: unsigned64, semantics: deltaCounter, units: events) Count of RTT time windows during which a loss indication (retransmit or jump) was observed. Only exported for TCP flows. If present in the template, enables sequence number tracking, RTT tracking, and options parsing.

reverseTcpLossEventCount trammell.ch (PEN 35566) IE 17422
(type: unsigned64, semantics: deltaCounter, units: events) Count of RTT time windows during which a loss indication (retransmit or jump) was observed. Only exported for TCP biflows. If present in the

template, enables sequence number tracking, RTT tracking, and options parsing.

qofTcpCharacteristics trammell.ch (PEN 35566) IE 1039
(type: unsigned32, semantics: flags) Flags describing particular characteristics of the flow. 0x01 = *ECT*(0) present on at least one packet. 0x02 = *ECT*(1) present on at least one packet. 0x04 = CE present on at least one packet. 0x10 = TCP Timestamp present. 0x20 = At least one segment was selectively acknowledged. 0x40 = TCP Window Scale present. Only exported for TCP flows. If present in the template, enables options parsing.

reverseQofTcpCharacteristics trammell.ch (PEN 35566) IE 17423
(type: unsigned32, semantics: flags) Flags describing particular characteristics of the reverse direction of the flow. 0x01 = *ECT*(0) present on at least one packet. 0x02 = *ECT*(1) present on at least one packet. 0x04 = CE present on at least one packet. 0x10 = TCP Timestamp present. 0x20 = At least one segment was selectively acknowledged. 0x40 = TCP Window Scale present. Only exported for TCP biflows. If present in the template, enables options parsing.

minTcpRwin trammell.ch (PEN 35566) IE 1042
(type: unsigned32, semantics: quantity, units: octets) Minimum observed scaled receiver window. Only exported for TCP flows. If present in the template, enables TCP options parsing and receiver window tracking.

reverseMinTcpRwin trammell.ch (PEN 35566) IE 17426
(type: unsigned32, semantics: quantity, units: octets) Minimum observed scaled receiver window in the reverse direction. Only exported for TCP biflows. If present in the template, enables TCP options parsing and receiver window tracking.

meanTcpRwin trammell.ch (PEN 35566) IE 1043
(type: unsigned32, semantics: quantity, units: octets) Mean observed scaled receiver window. Only exported for TCP flows. If present in the template, enables TCP options parsing and receiver window tracking.

reverseMeanTcpRwin trammell.ch (PEN 35566) IE 17427
(type: unsigned32, semantics: quantity, units: octets) Mean observed scaled receiver window in the reverse direction. Only exported for TCP biflows. If present in the template, enables TCP options parsing and receiver window tracking.

maxTcpRwin trammell.ch (PEN 35566) IE 1044
(type: unsigned32, semantics: quantity, units: octets) Maximum observed scaled receiver window. Only exported for TCP flows. If present in the template, enables TCP options parsing and receiver window tracking.

reverseMaxTcpRwin trammell.ch (PEN 35566) IE 17428
(type: unsigned32, semantics: quantity, units: octets) Maximum observed scaled receiver window in the reverse direction. Only exported for TCP biflows. If present in the template, enables TCP options parsing and receiver window tracking.

tcpReceiverStallCount trammell.ch (PEN 35566) IE 1045
(type: unsigned32, semantics: quantity, units: events) Count of transitions of the receiver window from nonzero to zero. Only exported for TCP flows. If present in the template, enables TCP options parsing and receiver window tracking.

reverseTcpReceiverStallCount trammell.ch (PEN 35566) IE 17429
(type: unsigned32, semantics: quantity, units: events) Count of transitions of the receiver window from nonzero to zero in the reverse direction. Only exported for TCP biflows. If present in the template, enables TCP options parsing and receiver window tracking.

tcpRttSampleCount trammell.ch (PEN 35566) IE 1046
(type: unsigned32, semantics: quantity, units: events) Count of RTT samples from which the flow's RTT information is derived. If present in the template, enables TCP options parsing and RTT tracking.

tcpTimestampFrequency trammell.ch (PEN 35566) IE 1047
(type: unsigned32, semantics: quantity, units: Hz) Observed frequency of TCP Timestamps. Only exported for TCP flows on which the timestamp option is present. If present in the template, turns on options parsing and timestamp tracking.

reverseTcpTimestampFrequency trammell.ch (PEN 35566) IE 17431
(type: unsigned32, semantics: quantity, units: Hz) Observed frequency of TCP Timestamps in the reverse direction. Only exported for TCP biflows on which the timestamp option is present. If present in the template, turns on options parsing and timestamp tracking.

Statistics Option Template

qof will export information about its process periodically using IPFIX Options Template Record. This record gives information about the status of the flow and fragment table, as well as decoding information. This can be turned off using the **−−no−stats** option. The following Information Elements will be exported:

systemInitTimeMilliseconds IE 161, 8 octets, unsigned
The time in milliseconds of the last (re−)initialization of **qof**.

exportedFlowRecordTotalCount IE 42, 8 octets, unsigned
Total amount of exported flows from **qof** start time.

packetTotalCount IE 86, 8 octets, unsigned
Total amount of packets processed by **qof** from **qof** start time.

droppedPacketTotalCount IE 135, 8 octets, unsigned
Total amount of dropped packets according to statistics given by libpcap, libdag, or libpcapexpress.

ignoredPacketTotalCount IE 164, 8 octets, unsigned
Total amount of packets ignored by the **qof** packet decoder, such as unsupported packet types and incomplete headers, from **qof** start time.

notSentPacketTotalCount IE 167, 8 octets, unsigned
Total amount of packets rejected by **qof** because they were received out of sequence.

expiredFragmentCount CERT (PEN 6871) IE 100, 4 octets, unsigned
Total amount of fragments that have been expired since **qof** start time.

assembledFragmentCount CERT (PEN 6871) IE 101, 4 octets, unsigned
Total number of packets that been assembled from a series of fragments since **qof** start time.

flowTableFlushEventCount CERT (PEN 6871) IE 104, 4 octets, unsigned
Total number of times the **qof** flow table has been flushed since **qof** start time.

flowTablePeakCount CERT (PEN 6871) IE 105, 4 octets, unsigned
The maximum number of flows in the **qof** flow table at any one time since **qof** start time.

exporterIPv4Address IE 130, 4 octets, unsigned
The IPv4 Address of the **qof** flow sensor.

exportingProcessId IE 144, 4 octets, unsigned
Set the ID of the **qof** flow sensor by giving a value to **−−observation−domain**. The default is 0.

meanFlowRate CERT (PEN 6871) IE 102, 4 octets, unsigned
The mean flow rate of the **qof** flow sensor since **qof** start time, rounded to the nearest integer.

meanPacketRate CERT (PEN 6871) IE 103, 4 octets, unsigned
The mean packet rate of the **qof** flow sensor since **qof** start time, rounded to the nearest integer.

SIGNALS

qof responds to **SIGINT** or **SIGTERM** by terminating input processing, flushing any pending flows to the current output, and exiting. If **−−verbose** is given, **qof** responds to **SIGUSR1** by printing present flow and fragment table statistics to its log. All other signals are handled by the C runtimes in the default manner on the platform on which **qof** is currently operating.

EXAMPLES

To generate flows from an pcap file into an IPFIX file, using configuration defaults:

```
qof --in pcapfile:packets.pcap --out flows.ipfix
```

To capture flows from an interface and export them to files in the current directory rotated hourly, with a given configuration

```
qof --yaml qof-config.yaml --in int:eth0 --out eth0_capture --rotate
3600
```

To capture flows from an Endace DAG card and export them via IPFIX over TCP:

```
yaf --yaml qof-config.yaml --in dag:/dev/dag0 --ipfix tcp --out
my-collector.example.com
```

AUTHORS

Brian Trammell <brian@trammell.ch>

Based on YAF by Chris Inacio, Emily Sarneso, and the CERT Network Situational Awareness Group Engineering Team, <http://www.cert.org/netsa>.

SEE ALSO

libtrace **(3)**