NAME

nmsgtool − command line interface to libnmsg

SYNOPSIS

nmsgtool [*OPTIONS*]...

DESCRIPTION

nmsgtool is the command line interface to libnmsg, the reference implementation of the NMSG binary structured message interchange format.

The NMSG format is an efficient encoding of typed, structured data into payloads which are packed into containers which can be transmitted over the network or stored to disk. Each payload is associated with a specific message schema. Modules implementing a certain message schema along with functionality to convert between binary and presentation formats can be loaded at runtime by libnmsg. **nmsgtool** provides a command line interface to control the transmission, storage, creation, and conversion of NMSG payloads.

nmsgtool is a thin wrapper around libnmsg's I/O engine. In order to run, **nmsgtool** needs to open at least one input and at least one output. The supported input types are:

- Reading binary NMSG data from a file or socket. See the **−r**, **−l** options.
- Reassembled IP datagrams from a network interface or **pcap-savefile**(5) file. See the **−i**, **−p** options.
- Reading ASCII presentation form data from a file. See the **−f** option.

Once input data has been read or converted to NMSG format internally with the help of an external module (in the case of pcap or ASCII data), it is striped or mirror across one or more outputs. The supported output types are:

- Writing binary NMSG data to a file or socket. See the **−w**, **−s** options.
- Writing ASCII presentation form data to a file. See the **−o** option.

Reading or writing data in a non-NMSG format requires the use of an external module (called an "nmsgpb module") to convert to or from NMSG format. **nmsgtool** selects an nmsgpb module based on a vendor ID and message type. For input data, these fields must be set with the **−V** and **−T** options.

If one or more inputs have been specified but no outputs have been specified, an implicit output of presentation form data to stdout will be assumed.

See the nmsg developer documentation for details on the NMSG wire format, how to interface with the nmsg library, and how to extend nmsg with new message types.

OPTIONS

−h, **−−help**
: Display help text and exit.

−d, **−−debug**
: Increment debugging level. **−dd** is verbose and **−dddd** is very verbose.

−V *vendor*, **−−vendor** *vendor*
: Set the vendor field of generated NMSG payloads to the vendor identified by *vname*. This is a symbolic string whose allowed values are determined at runtime when nmsgpb modules are loaded.

−T *msgtype*, **−−msgtype** *msgtype*
: Set the message type field of generated NMSG payloads to the message type identified by *msgtype*. This is a symbol string whose allowed values are determined at runtime when nmsgpb modules are loaded.

−e *endline*, **−−endline** *endline*
: Set the string to be used for the end-of-line continuation separator. By default this value is set to the newline character. This option understands the escape sequences **'\n'** (newline), **'\t'** (tab), and **'\\'** (backslash). Setting this option to **' \\\n\t'** will emulate the line continuation behavior of **ncaptool**.

`-m` *mtu*, **`--mtu`** *mtu*
Set the "maximum transmission unit" for writes to datagram socket outputs. Note that this option is somewhat misnamed since it does not refer to protocol data units of IP but to NMSG containers, so the additional small but bounded overhead of the UDP and NMSG headers need to be taken into account. Since NMSG's UDP transport is tuned for operation on jumbo Ethernet, this value defaults to *8192*. To avoid IP fragmentation over standard Ethernet or Internet transport, this value should be lowered to around *1280* to *1400*.

This option does not limit the maximum size of an NMSG payload because NMSG containers can be segmented into multiple datagrams.

`-c` *count*, **`--count`** *count*
Stop the process or (with **`-k`** specified) reopen outputs after *count* payloads have been processed. Note that each output keeps its own payload output counter and this value refers to that per-output value and not to the sum of all output counters.

`-t` *secs*, **`--interval`** *secs*
Stop the process or (with **`-k`** specified) reopen outputs on a time interval modulo the *secs* parameter. For example, *-t 3600* means "on the hour".

`-k` *cmd*, **`--kicker`** *cmd*
Make **`-c`** and **`-k`** continuous. In this mode output file names are suffixed with a timestamp and **nmsgtool** runs continuously, rotating output files as payload counts or time intervals expire. *cmd* specifies the command to run on output files after rotation. If *cmd* is set to the empty string '', then no command is executed and only file rotation is performed.

`-b` *filter*, **`--bpf`** *filter*
Filter pcap inputs (**`-p`** or **`-i`**) with the BPF expression *filter*. Note that libnmsg's pcap input is designed for IP datagrams and not network frames, so the filter specified by **`-b`** will be munged internally into several forms in order to receive IP fragments and filter the reassembled IP datagrams. Internally, IPv4 and IPv6 packets are selected for processing, including those received in VLAN tagged frames.

`-r` *file*, **`--readnmsg`** *file*
Read NMSG payloads from a file.

`-f` *file*, **`--readpres`** *file*
Read presentation format data from a file and convert to NMSG payloads. This option is dependent on the **`-V`** and **`-T`** options being set in order to select a specific nmsgpb module to perform presentation format to NMSG payload conversion. Not all nmsgpb modules necessarily support this conversion method, in which case **nmsgtool** will print a "function not implemented" message.

`-l` *addr*/*port*, **`--readsock`** *addr*/*port*
Read NMSG payloads from a UDP socket. The *addr* parameter must be set to a valid system or broadcast IPv4 or IPv6 address and the *port* parameter may be a single port number or a range of ports in the form *port0..portN*.

`-C` *channel*, **`--readchan`** *channel*
Read NMSG payloads from one or more UDP sockets specified by an alias *channel*. **nmsgtool** will read aliases from the file named *`nmsgtool.chalias`* in the system configuration directory (usually *`/usr/local/etc`* or *`/etc`*). The format of this file is one alias per line with each line starting with the alias name *channel* followed by one or more whitespace delimited *address*/*port* entries (as would be parsed by the **`--readsock`** option).

For example, the following alias file would create two channels. Calling **nmsgtool** with *-C 123* would be equivalent to calling **nmsgtool** with *-l 192.0.2.1/8430* while *-C 234* would be equivalent to *-l 192.0.2.255/8430..8437 -l 192.0.2.255/9430*.

nmsgtool.chalias example

```
123 192.0.2.1/8430
234 192.0.2.255/8430..8437 192.0.2.255/9430
```
.fi

`-p` *file*, **`--readpcap`** *file*
Read IP packets from a **pcap-savefile**(5) file
file using the **pcap**(3)
library. These packets are then reassembled into datagrams
which are then passed to an nmsgpb module for conversion
into NMSG payloads. This option is dependent on the
`-V` and **`-T`** options being
set in order to select a specific nmsgpb module to perform
IP datagram to NMSG payload conversion. Not all nmsgpb
modules necessarily support this conversion method, in which
case **nmsgtool** will print a "function not
implemented" message.

`-i` *if*[+][,*snap*]
Read IP packets from a network interface
if using the **pcap**(3)
library. Reassembly is performed as described for
`--readpcap`. +
may be appended to the interface name to capture in
promiscuous mode. The capture length
snap may be set by appending
,*snap*. The default capture length
is *1518*. **`-V`** and
`-T` are required.

`-w` *file*, **`--writenmsg`** *file*
Write NMSG payloads to a file.

`-o` *file*, **`--writepres`** *file*
Write presentation format payloads to a
file. **`-V`** and **`-T`** are required.

`-s` *addr*/*port*[,*rate*[,*freq*]], **`--writesock`** *addr*/*port*[,*rate*[,*freq*]]
Write NMSG payloads to a UDP socket socket specified
by the system or broadcast IPv4 or IPv6 address
addr and the UDP port
port. Optionally the output rate
may be limited to *rate* per second
by appending ,*rate*. If an output
rate is specified, the scheduling frequency
freq may be set by appending
,*freq*. The default scheduling
frequency for rate limits is
100.

NMSG payloads are not immediately output to sockets
but are instead concatenated into a buffer of a certain size
(see the **`--mtu`** option) before being
sent. To circumvent this behavior see the
`--unbuffered` option.

`-z`, **`--zlibout`**
Perform transparent zlib compression of written NMSG
containers. This applies to both file (**`-w`**)

and socket (**-s**) outputs.

--mirror
Mirror NMSG payloads across data outputs. By default NMSG payloads regardless of input source are striped across all available outputs. When **--mirror** is set, NMSG payloads are duplicated to every output. This option has no effect if there is only a single output.

--setsource *sonum*
Set the "source" field of output NMSG payloads to *sonum*.

NMSG payloads have an optional "source" field which is meant to be used as a unique opaque identifier identifying the immediate source of a redistributed payload. The *sonum* value should be specified as an unsigned 32 bit integer in hexadecimal format with a leading "0x".

In the **nmsg** presentation form output header, the source field is the fourth bracketed value.

--getsource *sonum*
Filter the "source" field of input NMSG payloads against *sonum*.

--setoperator *opname*
Set the "operator" field of output NMSG payloads to *opname*.

NMSG payloads have an optional "operator" field which is meant to identify the operator generating a payload. The operator field is represented as a 32 bit integer on the wire but is aliased to a symbolic string for presentation purposes by the file *nmsg.opalias* in the system configuration directory. The alias file contains one number/name pair separated by whitespace per line.

In the **nmsg** presentation form output header, the operator field is the fifth bracketed value.

--getoperator *opname*
Filter the "operator" field of input NMSG payloads against *opname*.

--setgroup *grname*
Set the "group" field of output NMSG payloads to *grname*.

NMSG payloads have an optional "group" field which is meant to identify the campaign or group that a payload belongs to. The group field is represented as a 32 bit integer on the wire but is aliased to a symbolic string for presentation purposes by the file *`nmsg.gralias`* in the system configuration directory. The alias file contains one number/name pair separated by whitespace per line.

In the **nmsg** presentation form output header, the group field is the sixth bracketed value.

`--getgroup` *grname*
Filter the "group" name of input NMSG payloads against *grname*.

EXAMPLES

To read NMSG payloads from a socket and write presentation form data to stdout:

```
nmsgtool -l 192.0.2.1/8430
```

To read NMSG payloads from a file and write presentation form data to stdout:

```
nmsgtool -r /tmp/file.nmsg
```

To read NMSG payloads from a socket and write to a binary NMSG file:

```
nmsgtool -l 192.0.2.1/8430 -w /tmp/file.nmsg
```

To read reassembled IP datagrams from a network interface in promiscuous mode, convert these datagrams to NMSG using the ISC/ncap nmsgpb module, and write to a file:

```
nmsgtool -i eth0+ -V ISC -T ncap -w /tmp/ncapfile.nmsg
```

To read NMSG payloads from multiple socket inputs and write to a series of compressed files, rotated every hour:

```
nmsgtool -l 192.0.2.255/8430..8437 -w /tmp/file -t 3600 -k '' -z
```