NMSG

Robert Edmonds
Internet Systems Consortium, Inc.

December 3, 2009

Introduction

NMSG is a file and wire format for storing and transmitting blobs of information.

Introduction

NMSG is a file and wire format for storing and transmitting blobs of information.

- Blobs of information on the order of 10 - 10,000 octets long.

Introduction

NMSG is a file and wire format for storing and transmitting blobs of information.

- Blobs of information on the order of 10 - 10,000 octets long.
- Network transport optimized for UDP over jumbo frame Ethernet.

Introduction

NMSG is a file and wire format for storing and transmitting blobs of information.

- Blobs of information on the order of 10 - 10,000 octets long.
- Network transport optimized for UDP over jumbo frame Ethernet.
- Framing encoded using Google Protocol Buffers. Blobs typically encoded using GPB as well.

Framing

- Constant length header part.

Framing

- Constant length header part.
- Variable length part. One of:

Framing

- Constant length header part.
- Variable length part. One of:
  - One or more payloads.

Framing

- Constant length header part.
- Variable length part. One of:
  - One or more payloads.
  - A fragment of a payload.

Constant length header part

- Magic value: 'N', 'M', 'S', 'G'.

Constant length header part

- Magic value: 'N', 'M', 'S', 'G'.
- Flags octet.

Constant length header part

- Magic value: 'N', 'M', 'S', 'G'.
- Flags octet.
  - `NMSG_FLAG_FRAGMENT`

Constant length header part

- Magic value: 'N', 'M', 'S', 'G'.
- Flags octet.
  - `NMSG_FLAG_FRAGMENT`
  - `NMSG_FLAG_ZLIB`

Constant length header part

- Magic value: 'N', 'M', 'S', 'G'.
- Flags octet.
  - `NMSG_FLAG_FRAGMENT`
  - `NMSG_FLAG_ZLIB`
- Version octet.

Constant length header part

- Magic value: 'N', 'M', 'S', 'G'.
- Flags octet.
  - `NMSG_FLAG_FRAGMENT`
  - `NMSG_FLAG_ZLIB`
- Version octet.
  - Current version is 2.

Constant length header part

- Magic value: 'N', 'M', 'S', 'G'.
- Flags octet.
  - `NMSG_FLAG_FRAGMENT`
  - `NMSG_FLAG_ZLIB`
- Version octet.
  - Current version is 2.
- 32 bit length of the variable length part of the message.

Flags

- NMSG_FLAG_FRAGMENT

Flags

- `NMSG_FLAG_FRAGMENT`
  - The variable length part of the message is fragmented into multiple frames.

Flags

- `NMSG_FLAG_FRAGMENT`
  - The variable length part of the message is fragmented into multiple frames.
  - Used when the size of an NMSG payload exceeds the NMSG buffer size. (NMSG buffer size is typically 8 KB for UDP sockets and 1 MB for files.)

Flags

- `NMSG_FLAG_FRAGMENT`
  - The variable length part of the message is fragmented into multiple frames.
  - Used when the size of an NMSG payload exceeds the NMSG buffer size. (NMSG buffer size is typically 8 KB for UDP sockets and 1 MB for files.)
  - Can't use IP fragmentation; max IP datagram size is 64 KB.

Flags

- `NMSG_FLAG_FRAGMENT`
  - The variable length part of the message is fragmented into multiple frames.
  - Used when the size of an NMSG payload exceeds the NMSG buffer size. (NMSG buffer size is typically 8 KB for UDP sockets and 1 MB for files.)
  - Can't use IP fragmentation; max IP datagram size is 64 KB.
- `NMSG_FLAG_ZLIB`

Flags

- `NMSG_FLAG_FRAGMENT`
  - The variable length part of the message is fragmented into multiple frames.
  - Used when the size of an NMSG payload exceeds the NMSG buffer size. (NMSG buffer size is typically 8 KB for UDP sockets and 1 MB for files.)
  - Can't use IP fragmentation; max IP datagram size is 64 KB.
- `NMSG_FLAG_ZLIB`
  - The variable length part of the message has been compressed with zlib.

Flags

- `NMSG_FLAG_FRAGMENT`
  - The variable length part of the message is fragmented into multiple frames.
  - Used when the size of an NMSG payload exceeds the NMSG buffer size. (NMSG buffer size is typically 8 KB for UDP sockets and 1 MB for files.)
  - Can't use IP fragmentation; max IP datagram size is 64 KB.
- `NMSG_FLAG_ZLIB`
  - The variable length part of the message has been compressed with zlib.
- `NMSG_FLAG_FRAGMENT | NMSG_FLAG_ZLIB`

Flags

- `NMSG_FLAG_FRAGMENT`
  - The variable length part of the message is fragmented into multiple frames.
  - Used when the size of an NMSG payload exceeds the NMSG buffer size. (NMSG buffer size is typically 8 KB for UDP sockets and 1 MB for files.)
  - Can't use IP fragmentation; max IP datagram size is 64 KB.
- `NMSG_FLAG_ZLIB`
  - The variable length part of the message has been compressed with zlib.
- `NMSG_FLAG_FRAGMENT | NMSG_FLAG_ZLIB`
  - The variable length part of the message has been compressed, **then** fragmented.

Payload header

- Vendor ID

Payload header

- Vendor ID
- Message type

Payload header

- Vendor ID
- Message type
- Timestamp

Payload header

- Vendor ID
- Message type
- Timestamp
- Optional classification

Payload header

- Vendor ID
- Message type
- Timestamp
- Optional classification
  - Source

Payload header

- Vendor ID
- Message type
- Timestamp
- Optional classification
  - Source
  - Operator

Payload header

- Vendor ID
- Message type
- Timestamp
- Optional classification
  - Source
  - Operator
  - Group

Payload

- The ‘blob’ of information.

Payload

- The ‘blob’ of information.
- Each (*vendor ID*, *message type*) tuple identifies a unique type of message.

Payload

- The ‘blob’ of information.
- Each (*vendor ID*, *message type*) tuple identifies a unique type of message.
- Payload blobs can be encoded with GPB, but are not required to be.

libnmsg client API

- C library for applications that want to send, receive, read, write and process NMSG messages.

libnmsg client API

- C library for applications that want to send, receive, read, write and process NMSG messages.
- Includes single-threaded and multi-threaded input/output interfaces.

libnmsg client API

- C library for applications that want to send, receive, read, write and process NMSG messages.
- Includes single-threaded and multi-threaded input/output interfaces.
  - Multi-threaded I/O code spreads deserialization load across multiple CPUs when using multiple inputs.

libnmsg client API

- C library for applications that want to send, receive, read, write and process NMSG messages.
- Includes single-threaded and multi-threaded input/output interfaces.
  - Multi-threaded I/O code spreads deserialization load across multiple CPUs when using multiple inputs.
- Python and Perl bindings under development.

`libnmsg` message module interface

- Plugins can extend at run-time the messages types that `libnmsg` understands.

`libnmsg` message module interface

- Plugins can extend at run-time the messages types that `libnmsg` understands.
- Typically just `protobuf-c` generated object code plus glue.

`libnmsg` message module interface

- Plugins can extend at run-time the messages types that `libnmsg` understands.
- Typically just `protobuf-c` generated object code plus glue.
  - Can be more complex: the `ISC/dns` message module uses hooks to pretty print the data fields used in DNS.