NAME

warts — format for scamper's warts storage.

DESCRIPTION

This document describes the warts binary file format used by scamper(1). A warts file consists of a set of records, each of which begin with an 8 byte object header.

WARTS OBJECT HEADER

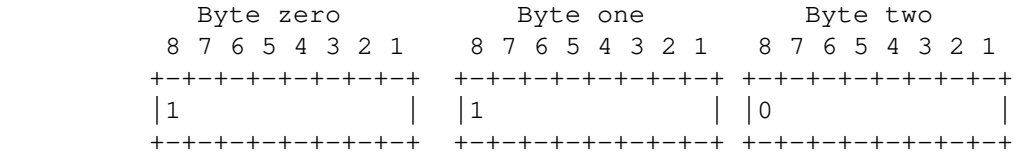
The header consists of a 2-byte magic number (0x1205), a 2-byte type value, and a 4-byte length value defining the size of the object that follows. All 16-bit and 32-bit numbers are written in network byte order. The currently defined types, and the types of scamper(1) object they map to, are as follows:

- 0x0001: List (scamper_list_t)
- 0x0002: Cycle start (scamper_cycle_t)
- 0x0003: Cycle definition (scamper_cycle_t)
- 0x0004: Cycle stop (scamper_cycle_t)
- 0x0005: Address (scamper_addr_t) -- deprecated
- 0x0006: Traceroute (scamper_trace_t)
- 0x0007: Ping (scamper_ping_t)
- 0x0008: MDA traceroute (scamper_tracelb_t)
- 0x0009: Alias resolution (scamper_dealias_t)
- 0x000a: Neighbour Discovery (scamper_neighbourdisc_t)
- 0x000b: TCP Behaviour Inference Tool (scamper_tbit_t)
- 0x000c: Sting (scamper_sting_t)
- 0x000d: Sniff (scamper_sniff_t)

A new type number can be requested by emailing the author of scamper. The structure of each warts record beyond the header is arbitrary, though some conventions have been established to promote extensibility.

FLAGS AND PARAMETERS

The warts routines in scamper provide the ability to conditionally store arbitrary data in a forwards compatible method. A set of flags and parameters begins with a sequence of bytes that denote which items are included. If any flags are set, then after the flags is a 2-byte field that records the length of the parameters that follow. Finally, the data follows. The following figure illustrates how flags are recorded:

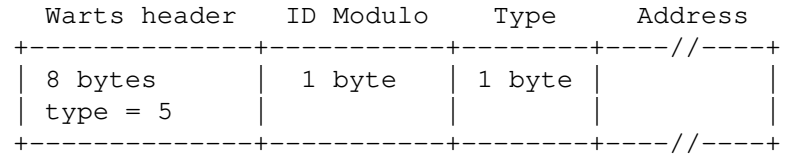


The most significant bit of each byte is the 'link' bit; it determines if the next byte in the sequence contains flags. The low-order 7 bits of each byte signal if the corresponding field is written out in the parameters that follow. In the figure, the link bit is set to one in the first two bytes, and zero in the final byte, signifying that three flag-bytes are included.

The rest of each byte is used to record flags, whose position in the sequence signifies if a particular parameter is included. For example, if bit 6 of byte zero is set, then parameter 6 is included, and if bit 5 of byte one is set, then parameter 12 is included, and if bit 2 of byte two is set, then parameter 16 is included.

ADDRESSES

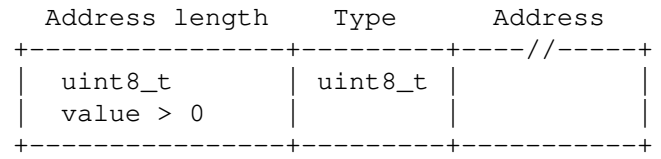
A warts file may have addresses embedded in two ways. The first is deprecated: the address is written as a data object that can be globally referenced before the data object that uses it is written. A reader therefore must keep a copy of all addresses it reads in order to be able to decode data objects that subsequently reference it, which can consume a significant amount of memory. The format of the address is



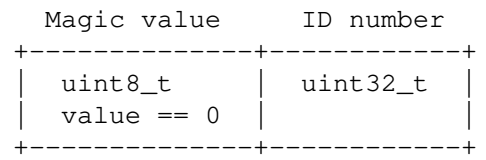
A reader determines the ID number of each address by the order in which it appears, and can sanity check the ID number it determines by comparing the lower 8 bits of the computed ID with the ID that is embedded in the record. Address ID numbers start at one; zero is reserved for when no address is embedded. The type corresponds to the type of address that follows. The currently defined types are as follows:

- 0x01 IPv4 address
- 0x02 IPv6 address
- 0x03 48-bit Ethernet MAC address
- 0x04 64-bit Firewire link address

The second method to embed an address is to embed the address in each data object that requires that address. The format of that address can take one of two forms, depending on whether or not the address is being defined or referenced. A defined address declares a new address that has scope for the data object being embedded; a reader adds the address to the end of a table so that it can be later referenced without having to re-define the address. In this method, ID numbers start from zero. The format of a defined address is:



The format of a referenced address is:



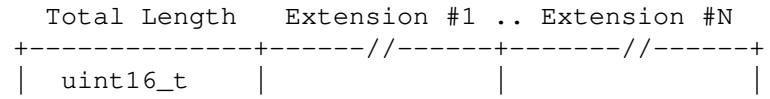
EMBEDDING OTHER TYPES

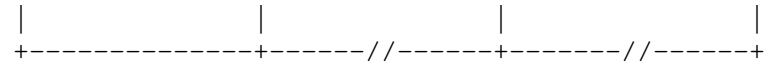
Bytes, unsigned 16-bit integers, and unsigned 32 bit integers are embedded directly, using network byte order where appropriate. ASCII strings are also embedded directly, including the trailing null byte to terminate the string. Time values (timeval) are embedded as two unsigned 32 bit integers; the first number counts the number of seconds that have elapsed since the Unix epoch, the second number counts the number of microseconds that have elapsed in the current second. Round-trip-time (RTT) values are embedded as a single unsigned 32 bit integer that counts the number of microseconds that elapsed.

Several measurements record ICMP extension data, so there is a standardised method to record a set of ICMP extensions. Individual ICMP extension records are written in the following format: The format of a list structure is:

- uint16_t: length of data that follows
- uint8_t: ICMP extension class number
- uint8_t: ICMP extension type number
- Bytes: ICMP extension data, if any.

A set of ICMP extension records is written in the following format:





LIST STRUCTURE

The format of a list structure is:

- 8 bytes: Warts header, type 0x0001
- uint32_t: List ID assigned by warts from a counter
- uint32_t: List ID assigned by a person.
- String: List Name assigned by a person
- Variable: Flags
- uint16_t: Parameter length (optional, included if any flags are set)
- String: Description, included if flag 1 is set
- String: Monitor name, included if flag 2 is set

The List ID assigned by warts is subsequently used by objects that reference the list to identify which list they refer to.

CYCLE STRUCTURE

Three types of cycle records may be written: a start record denoting the starting point for a new cycle, a definition record declaring a cycle record whose corresponding start record is in a different file, and a cycle stop record, denoting the end point for a cycle. The format of the cycle start and definition structures is:

- 8 bytes: Warts header, type 0x0002 or 0x0003
- uint32_t: Cycle ID, assigned by warts from a counter
- uint32_t: List ID, referencing the list this cycle is over
- uint32_t: Cycle ID, assigned by a human
- uint32_t: Start time of the cycle, seconds since Unix epoch
- Variable: Flags
- uint16_t: Parameter length, included if any flags are set
- uint32_t: Stop time of the cycle in seconds since Unix epoch, included if flag 1 is set
- String: Hostname at cycle start point, included if flag 2 is set

The format of the cycle stop structure is:

- 8 bytes: Warts header, type 0x0004
- uint32_t: Cycle ID, assigned by warts from a counter, referencing the cycle structure that is being updated.
- uint32_t: Stop time of the cycle, seconds since Unix epoch
- Variable: Flags. currently set to zero.

TRACEROUTE STRUCTURE

Traceroute structures consist of traceroute parameters, hop records, and an optional series of additional data types for special types of traceroute invokation. The general form of a traceroute recorded in warts is as follows:

- 8 bytes: Warts header, type 0x0006
- Variable: Flags describing traceroute parameters and high-level outcomes
- uint16_t: Parameter length, included if any flags are set
- Variable: Traceroute parameters, depending on flags
- uint16_t: Hop record count
- Variable: Hop records, if hop record count > 0
- Variable: Optional traceroute data; pmtud, doubletree
- uint16_t: End of traceroute record; value is zero.

The flags and data types that describe traceroute are as follows:

- uint32_t: List ID assigned by warts, included if flag 1 is set
- uint32_t: Cycle ID assigned by warts, included if flag 2 is set
- uint32_t: Src IP address ID assigned by warts, included if flag 3 is set
- uint32_t: Dst IP address ID assigned by warts, included if flag 4 is set
- timeval: Time traceroute commenced, included if flag 5 is set
- uint8_t: Stop reason, included if flag 6 is set
- uint8_t: Stop data, included if flag 7 is set
- uint8_t: Trace flags, included if flag 8 is set
- uint8_t: Attempts, included if flag 9 is set
- uint8_t: Hoplimit, included if flag 10 is set
- uint8_t: Trace type, included if flag 11 is set
- uint16_t: Probe size, included if flag 12 is set
- uint16_t: Source port, included if flag 13 is set
- uint16_t: Destination port, included if flag 14 is set
- uint8_t: TTL of first probe, included if flag 15 is set
- uint8_t: IP ToS set in probe packets, included if flag 16 is set
- uint8_t: Timeout length for each probe in seconds, included if flag 17 is set
- uint8_t: How many loops are allowed before probing halts, included if flag 18 is set
- uint16_t: Number of hops probed, included if flag 19 is set
- uint8_t: Gap limit before probing halts, included if flag 20 is set
- uint8_t: What to do when the gap limit is reached, included if flag 21 is set
- uint8_t: What to do when a loop is found, included if flag 22 is set
- uint16_t: Number of probes sent, included if flag 23 is set
- uint8_t: Minimum time to wait between probes in centiseconds, included if flag 24 is set
- uint8_t: Confidence level to attain that all hops have replied at a given distance in the path, included if flag 25 is set
- address: Source address used in probes, included if flag 26 is set
- address: Destination address used in probes, included if flag 27 is set
- uint32_t: User ID assigned to the traceroute, included if flag 28 is set
- uint16_t: IP offset value used in probes, included if flag 29 is set
- address: Router address used to send probes, included if flag 30 is set

The traceroute flags field has the following fields:

- If bit 1 is set, traceroute sent all allotted attempts.
- If bit 2 is set, traceroute was instructed to conduct path MTU discovery.
- If bit 3 is set, traceroute should use the datalink to obtain timestamps.
- If bit 4 is set, traceroute should not halt probing if a TTL expired message is received from the destination.
- If bit 5 is set, traceroute should use Doubletree to reduce redundant probing.
- If bit 6 is set, the ICMP checksum used in echo probes can be found is stored where the UDP destination port value is.

Hop records are written in series. Each hop record takes the following form:

- Variable: Flags describing which hop parameters are recorded
- uint16_t: Parameter length, included if any flags are set
- uint32_t: Hop address, ID corresponding to global warts address; included if flag 1 is set
- uint8_t: IP TTL of probe packet, included if flag 2 is set
- uint8_t: IP TTL of reply packet, included if flag 3 is set
- uint8_t: Hop flags, included if flag 4 is set
- uint8_t: Hop probe ID - how many probes have been sent for the given TTL. Included if flag 5 is set.

- RTT: Round trip time - the length of time in microseconds it took this reply to arrive after the probe was transmitted. Included if flag 6 is set.
- uint16_t: ICMP type, code. The first byte is the ICMP type of the response, the second byte is the ICMP code. Included if flag 7 is set.
- uint16_t: Probe size - the size of the probe sent. Included if flag 8 is set.
- uint16_t: Reply size - the size of the response received. Included if flag 9 is set.
- uint16_t: IPID - the IP identifier value set in the response packet. Included if flag 10 is set, else it is zero.
- uint8_t: Type of Service - the value of the ToS byte in the IP header, including ECN bits. Included if flag 11 is set.
- uint16_t: Next-hop MTU - the value of the next-hop MTU field if the response is an ICMP packet too big message. Included if flag 12 is set.
- uint16_t: Quoted IP length - the value of the IP length field found in the ICMP quotation. Included if flag 13 is set, else it is the same as the probe size.
- uint8_t: Quoted TTL - the value of the IP TTL field found in the ICMP quotation. Included if flag 14 is set, else it is one.
- uint8_t: TCP flags - the value of the TCP flags received in response to TCP probes. Included if flag 15 is set.
- uint8_t: Quoted TOS - the value of the IP ToS byte found in the ICMP quotation. Included if flag 16 is set.
- icmpext: ICMP extension data, included if flag 17 is set.
- address: Hop address, included if flag 18 is set.
- timeval: Hop tx, included if flag 19 is set.

Optional traceroute data, such as PMTUD and doubletree control and result structures are included after hop records. Optional traceroute data begins with a 16-bit header; the first four bits define the type of record, and the remaining 12 bits specify the length of the record. Currently, three types of optional data are defined: PMTUD data (1), Last-ditch probing results (2), and doubletree (3).

The format of the last-ditch data is:
- uint16_t: traceroute optional data header, type = 1.
- Variable: Flags describing which last-ditch parameters are recorded. Currently, no flags are defined.
- uint16_t: Parameter length, included if any flags are set.
- uint16_t: Number of responses received to last-ditch probing, recorded as hop records.
- Variable: Hop records.

The format of PMTUD data is:
- uint16_t: traceroute optional data header, type = 2.
- Variable: PMTUD flags and parameters
- uint16_t: Number of hop records that follow
- Variable: Hop Records, if any
- Variable: Notes, if any

The format of the PMTUD flags and attributes is:
- Variable: Flags describing which hop parameters are recorded
- uint16_t: Parameter length, included if any flags are set
- uint16_t: MTU of the interface, included if flag 1 is set.
- uint16_t: Path MTU, included if flag 2 is set.
- uint16_t: MTU to the first hop, included if flag 3 is set and if it differs to the MTU of the interface.
- uint8_t: verson of the PMTUD attribute, included if flag 4 is set, otherwise version 1 can be assumed.
- uint8_t: note count - number of PMTUD note structures that follow the hops.

The format of the PMTUD notes is:

- Variable: Flags describing which hop parameters are recorded
- uint16_t: Parameter length, included if any flags are set
- uint8_t: type of note, included if flag 1 is set.
- uint16_t: next-hop MTU inferred, included if flag 2 is set.
- uint16_t: Index of corresponding hop record in the PMTUD hops, included if flag 3 is set.

The tree types of PMTUD notes are: ordinary PTB (1), PTB with invalid next-hop MTU (2), and an inferred MTU in the absence of a PTB (3).

The format of doubletree data is:
- uint16_t: traceroute optional data header, type = 3.
- Variable: doubletree flags and parameters

The format of the doubletree flags and attributes is:
- uint32_t: Local Stop Set stop IP address ID assigned by warts, included if flag 1 is set
- uint32_t: Global Stop Set stop IP address ID assigned by warts, included if flag 2 is set
- uint8_t: First hop to probe from, included if flag 3 is set.
- address: Local Stop Set stop address, included if flag 4 is set.
- address: Global Stop Set stop address, included if flag 5 is set.
- String: Local Stop Set name, included if flag 6 is set.
- uint8_t: doubltree flags, included if flag 7 is set.

PING STRUCTURE

Ping structures consist of ping parameters and responses. The general form of a ping recorded in warts is as follows:
- 8 bytes: Warts header, type 0x0007
- Variable: Flags describing ping parameters and high-level outcomes
- uint16_t: Parameter length, included if any flags are set
- Variable: ping parameters, depending on flags
- uint16_t: Ping reply count
- Variable: Ping replies, if ping reply count > 0

The flags and data types that describe ping are as follows:
- uint32_t: List ID assigned by warts, included if flag 1 is set
- uint32_t: Cycle ID assigned by warts, included if flag 2 is set
- uint32_t: Src IP address ID assigned by warts, included if flag 3 is set
- uint32_t: Dst IP address ID assigned by warts, included if flag 4 is set
- timeval: Time ping commenced, included if flag 5 is set
- uint8_t: Stop reason, included if flag 6 is set
- uint8_t: Stop data, included if flag 7 is set
- uint16_t: Data length, included if flag 8 is set
- Variable: data bytes, included if flag 9 is set
- uint16_t: Probe count, included if flag 10 is set
- uint16_t: Probe size, included if flag 11 is set
- uint8_t: Probe wait (seconds), included if flag 12 is set
- uint8_t: Probe TTL, included if flag 13 is set
- uint16_t: Reply count, included if flag 14 is set
- uint16_t: Pings sent, included if flag 15 is set
- uint8_t: Ping method, included if flag 16 is set
- uint16_t: Probe source port, included if flag 17 is set
- uint16_t: Probe destination port, included if flag 18 is set
- uint32_t: User ID, included if flag 19 is set

- address: Source address used, included if flag 20 is set
- address: Destination address used, included if flag 21 is set
- uint8_t: Ping flags, included if flag 22 is set
- uint8_t: Probe TOS, included if flag 23 is set
- variable: Probe Pre-specified timestamp option, included if flag 24 is set
- uint16_t: Probe ICMP checksum, included if flag 25 is set
- uint16_t: Reply psuedo Path MTU, included if flag 26 is set
- uint8_t: Probe timeout, included if flag 27 is set
- uint32_t: Probe wait (microseconds), included if flag 28 is set
- uint32_t: Probe TCP acknowledgment value, included if flag 29 is set
- uint16_t: Ping flags, included if flag 30 is set.
- uint32_t: Probe TCP sequence number value, included if flag 31 is set
- address: Router address used to send probes, included if flag 32 is set

Ping response records are written in series. Each record takes the following form:

SEE ALSO

`scamper`(1), `libscamperfile`(3), `sc_wartsdump`(1),

M. Luckie, *Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet*, Proc. ACM/SIGCOMM Internet Measurement Conference 2010.

AUTHORS

`warts` was written by Matthew Luckie <mjl@luckie.org.nz>.