libnet

1.2-rc3

Generated by Doxygen 1.8.14

Contents

1 Libnet Packet Assembly Library **1**

1.1 Overview . . . 1

2 File Index **3**

2.1 File List . . . 3

3 File Documentation **5**

3.1 include/libnet.h File Reference . . . 5

3.1.1 Detailed Description . . . 5

3.2 include/libnet/libnet-functions.h File Reference . . . 6

3.2.1 Detailed Description . . . 10

3.2.2 Function Documentation . . . 10

3.2.2.1 libnet_addr2name4() . . . 10

3.2.2.2 libnet_addr2name6_r() . . . 11

3.2.2.3 libnet_adv_cull_header() . . . 11

3.2.2.4 libnet_adv_cull_packet() . . . 12

3.2.2.5 libnet_adv_free_packet() . . . 12

3.2.2.6 libnet_adv_write_link() . . . 12

3.2.2.7 libnet_adv_write_raw_ipv4() . . . 13

3.2.2.8 libnet_autobuild_arp() . . . 14

3.2.2.9 libnet_autobuild_ethernet() . . . 14

3.2.2.10 libnet_autobuild_fddi() . . . 15

3.2.2.11 libnet_autobuild_ipv4() . . . 15

3.2.2.12 libnet_autobuild_ipv6() . . . 16

3.2.2.13 libnet_autobuild_link() . . . 17
3.2.2.14 libnet_autobuild_token_ring() . . . 17
3.2.2.15 libnet_build_802_1q() . . . 19
3.2.3 Packet Builder Functions . . . 19
3.2.3.1 The Payload Interface . . . 20
3.2.3.2 Protocol Tags and Packet Builder Return Values . . . 20
3.2.3.3 libnet_build_802_1x() . . . 21
3.2.3.4 libnet_build_802_2() . . . 21
3.2.3.5 libnet_build_802_2snap() . . . 22
3.2.3.6 libnet_build_802_3() . . . 23
3.2.3.7 libnet_build_arp() . . . 24
3.2.3.8 libnet_build_bgp4_header() . . . 25
3.2.3.9 libnet_build_bgp4_notification() . . . 25
3.2.3.10 libnet_build_bgp4_open() . . . 26
3.2.3.11 libnet_build_bgp4_update() . . . 27
3.2.3.12 libnet_build_bootpv4() . . . 28
3.2.3.13 libnet_build_cdp() . . . 29
3.2.3.14 libnet_build_data() . . . 30
3.2.3.15 libnet_build_dhcpv4() . . . 30
3.2.3.16 libnet_build_dnsv4() . . . 32
3.2.3.17 libnet_build_egre() . . . 33
3.2.3.18 libnet_build_ethernet() . . . 34
3.2.3.19 libnet_build_fddi() . . . 35
3.2.3.20 libnet_build_gre() . . . 36
3.2.3.21 libnet_build_gre_last_sre() . . . 36
3.2.3.22 libnet_build_gre_sre() . . . 37
3.2.3.23 libnet_build_hsrp() . . . 38
3.2.3.24 libnet_build_icmpv4_echo() . . . 39
3.2.3.25 libnet_build_icmpv4_mask() . . . 39
3.2.3.26 libnet_build_icmpv4_redirect() . . . 40

3.2.3.27 libnet_build_icmpv4_timeexceed() . . . 41
3.2.3.28 libnet_build_icmpv4_timestamp() . . . 41
3.2.3.29 libnet_build_icmpv4_unreach() . . . 42
3.2.3.30 libnet_build_icmpv6_echo() . . . 43
3.2.3.31 libnet_build_icmpv6_ndp_nadv() . . . 44
3.2.3.32 libnet_build_icmpv6_ndp_nsol() . . . 45
3.2.3.33 libnet_build_icmpv6_ndp_opt() . . . 45
3.2.3.34 libnet_build_icmpv6_unreach() . . . 46
3.2.3.35 libnet_build_igmp() . . . 47
3.2.3.36 libnet_build_ipsec_ah() . . . 47
3.2.3.37 libnet_build_ipsec_esp_ftr() . . . 48
3.2.3.38 libnet_build_ipsec_esp_hdr() . . . 49
3.2.3.39 libnet_build_ipv4() . . . 50
3.2.3.40 libnet_build_ipv4_options() . . . 51
3.2.3.41 libnet_build_ipv6() . . . 51
3.2.3.42 libnet_build_ipv6_destopts() . . . 52
3.2.3.43 libnet_build_ipv6_frag() . . . 53
3.2.3.44 libnet_build_ipv6_hbhopts() . . . 53
3.2.3.45 libnet_build_ipv6_routing() . . . 54
3.2.3.46 libnet_build_isl() . . . 55
3.2.3.47 libnet_build_link() . . . 56
3.2.3.48 libnet_build_mpls() . . . 57
3.2.3.49 libnet_build_ntp() . . . 57
3.2.3.50 libnet_build_ospfv2() . . . 59
3.2.3.51 libnet_build_ospfv2_dbd() . . . 59
3.2.3.52 libnet_build_ospfv2_hello() . . . 60
3.2.3.53 libnet_build_ospfv2_lsa() . . . 61
3.2.3.54 libnet_build_ospfv2_lsa_as() . . . 62
3.2.3.55 libnet_build_ospfv2_lsa_net() . . . 62
3.2.3.56 libnet_build_ospfv2_lsa_rtr() . . . 63

3.2.3.57 libnet_build_ospfv2_lsa_sum() . . . 64
3.2.3.58 libnet_build_ospfv2_lsr() . . . 64
3.2.3.59 libnet_build_ospfv2_lsu() . . . 65
3.2.3.60 libnet_build_rip() . . . 66
3.2.3.61 libnet_build_rpc_call() . . . 66
3.2.3.62 libnet_build_sebek() . . . 67
3.2.3.63 libnet_build_stp_conf() . . . 68
3.2.3.64 libnet_build_stp_tcn() . . . 69
3.2.3.65 libnet_build_tcp() . . . 70
3.2.3.66 libnet_build_tcp_options() . . . 71
3.2.3.67 libnet_build_token_ring() . . . 72
3.2.3.68 libnet_build_udp() . . . 73
3.2.3.69 libnet_build_vrrp() . . . 73
3.2.3.70 libnet_clear_packet() . . . 74
3.2.3.71 libnet_cq_add() . . . 75
3.2.3.72 libnet_cq_destroy() . . . 75
3.2.3.73 libnet_cq_end_loop() . . . 75
3.2.3.74 libnet_cq_find_by_label() . . . 75
3.2.3.75 libnet_cq_getlabel() . . . 76
3.2.3.76 libnet_cq_head() . . . 76
3.2.3.77 libnet_cq_last() . . . 77
3.2.3.78 libnet_cq_next() . . . 77
3.2.3.79 libnet_cq_remove() . . . 77
3.2.3.80 libnet_cq_remove_by_label() . . . 78
3.2.3.81 libnet_cq_size() . . . 78
3.2.3.82 libnet_destroy() . . . 78
3.2.3.83 libnet_diag_dump_context() . . . 79
3.2.3.84 libnet_diag_dump_hex() . . . 79
3.2.3.85 libnet_diag_dump_pblock() . . . 79
3.2.3.86 libnet_diag_dump_pblock_type() . . . 80

3.2.3.87 libnet_get_hwaddr() . . . 80
3.2.3.88 libnet_get_ipaddr4() . . . 81
3.2.3.89 libnet_get_ipaddr6() . . . 81
3.2.3.90 libnet_get_prand() . . . 81
3.2.3.91 libnet_getdevice() . . . 82
3.2.3.92 libnet_geterror() . . . 82
3.2.3.93 libnet_getfd() . . . 83
3.2.3.94 libnet_getgre_length() . . . 83
3.2.3.95 libnet_getpacket_size() . . . 83
3.2.3.96 libnet_getpbuf() . . . 84
3.2.3.97 libnet_getpbuf_size() . . . 84
3.2.3.98 libnet_hex_aton() . . . 84
3.2.3.99 libnet_in6_is_error() . . . 85
3.2.3.100 libnet_init() . . . 85
3.2.3.101 libnet_name2addr4() . . . 86
3.2.3.102 libnet_name2addr6() . . . 86
3.2.3.103 libnet_plist_chain_dump() . . . 87
3.2.3.104 libnet_plist_chain_dump_string() . . . 87
3.2.3.105 libnet_plist_chain_free() . . . 88
3.2.3.106 libnet_plist_chain_new() . . . 88
3.2.3.107 libnet_plist_chain_next_pair() . . . 89
3.2.3.108 libnet_seed_prand() . . . 89
3.2.3.109 libnet_stats() . . . 89
3.2.3.110 libnet_toggle_checksum() . . . 90
3.2.3.111 libnet_version() . . . 90
3.2.3.112 libnet_write() . . . 91
3.3 include/libnet/libnet-macros.h File Reference . . . 91
3.3.1 Detailed Description . . . 92
3.3.2 Macro Definition Documentation . . . 92
3.3.2.1 for_each_context_in_cq . . . 92
3.3.2.2 IN6ADDR_ERROR_INIT . . . 92
3.3.2.3 LIBNET_DONT_RESOLVE . . . 92
3.3.2.4 LIBNET_ERRBUF_SIZE . . . 93
3.3.2.5 LIBNET_MAX_PACKET . . . 93
3.3.2.6 LIBNET_MAXOPTION_SIZE . . . 93
3.3.2.7 LIBNET_OFF . . . 93
3.3.2.8 LIBNET_ON . . . 93
3.3.2.9 LIBNET_PR2 . . . 93
3.3.2.10 LIBNET_RESOLVE . . . 93

Index **95**

Chapter 1

Libnet Packet Assembly Library

1.1 Overview

Libnet is a high-level API (toolkit) allowing the application programmer to construct and inject network packets. It provides a portable and simplified interface for low-level network packet shaping, handling and injection. Libnet hides much of the tedium of packet creation from the application programmer such as multiplexing, buffer management, arcane packet header information, byte-ordering, OS-dependent issues, and much more. Libnet features portable packet creation interfaces at the IP layer and link layer, as well as a host of supplementary and complementary functionality. Using libnet, quick and simple packet assembly applications can be whipped up with little effort. With a bit more time, more complex programs can be written (Traceroute and ping were easily rewritten using libnet and `libpcap`).

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

include/libnet.h
Toplevel libnet header file 5
include/libnet/libnet-functions.h
Libnet exported function prototypes 6
include/libnet/libnet-macros.h
Libnet macros and symbolic constants 91

Chapter 3

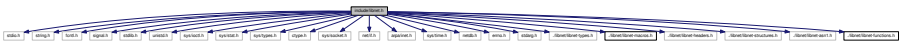
File Documentation

3.1 include/libnet.h File Reference

toplevel libnet header file

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <ctype.h>
#include <sys/socket.h>
#include <net/if.h>
#include <arpa/inet.h>
#include <sys/time.h>
#include <netdb.h>
#include <errno.h>
#include <stdarg.h>
#include "./libnet/libnet-types.h"
#include "./libnet/libnet-macros.h"
#include "./libnet/libnet-headers.h"
#include "./libnet/libnet-structures.h"
#include "./libnet/libnet-asn1.h"
#include "./libnet/libnet-functions.h"
```

Include dependency graph for libnet.h:



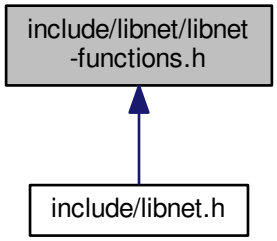
3.1.1 Detailed Description

toplevel libnet header file

3.2 include/libnet/libnet-functions.h File Reference

libnet exported function prototypes

This graph shows which files directly or indirectly include this file:



Functions

- LIBNET_API libnet_t * libnet_init (int injection_type, const char *device, char *err_buf)
- LIBNET_API void libnet_destroy (libnet_t *l)
- LIBNET_API void libnet_clear_packet (libnet_t *l)
- LIBNET_API void libnet_stats (libnet_t *l, struct libnet_stats *ls)
- LIBNET_API int libnet_getfd (libnet_t *l)
- LIBNET_API const char * libnet_getdevice (libnet_t *l)
- LIBNET_API uint8_t * libnet_getpbuf (libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API uint32_t libnet_getpbuf_size (libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API char * libnet_geterror (libnet_t *l)
- LIBNET_API uint32_t libnet_getpacket_size (libnet_t *l)
- LIBNET_API int libnet_seed_prand (libnet_t *l)
- LIBNET_API uint32_t libnet_get_prand (int mod)
- LIBNET_API int libnet_toggle_checksum (libnet_t *l, libnet_ptag_t ptag, int mode)
- LIBNET_API char * libnet_addr2name4 (uint32_t in, uint8_t use_name)
- LIBNET_API uint32_t libnet_name2addr4 (libnet_t *l, const char *host_name, uint8_t use_name)
- LIBNET_API int libnet_in6_is_error (struct libnet_in6_addr addr)
- LIBNET_API struct libnet_in6_addr libnet_name2addr6 (libnet_t *l, const char *host_name, uint8_t use_↩ name)
- LIBNET_API void libnet_addr2name6_r (struct libnet_in6_addr addr, uint8_t use_name, char *host_name, int host_name_len)
- LIBNET_API int libnet_plist_chain_new (libnet_t *l, libnet_plist_t **plist, char *token_list)
- LIBNET_API int libnet_plist_chain_next_pair (libnet_plist_t *plist, uint16_t *bport, uint16_t *eport)
- LIBNET_API int libnet_plist_chain_dump (libnet_plist_t *plist)
- LIBNET_API char * libnet_plist_chain_dump_string (libnet_plist_t *plist)
- LIBNET_API int libnet_plist_chain_free (libnet_plist_t *plist)
- LIBNET_API libnet_ptag_t libnet_build_802_1q (const uint8_t *dst, const uint8_t *src, uint16_t tpi, uint8_t priority, uint8_t cfi, uint16_t vlan_id, uint16_t len_proto, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_802_1x (uint8_t eap_ver, uint8_t eap_type, uint16_t length, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)

- LIBNET_API libnet_ptag_t libnet_build_802_2 (uint8_t dsap, uint8_t ssap, uint8_t control, const uint8_↩ t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_802_2snap (uint8_t dsap, uint8_t ssap, uint8_t control, uint8_t *oui, uint16_t type, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_802_3 (const uint8_t *dst, const uint8_t *src, uint16_t len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ethernet (const uint8_t *dst, const uint8_t *src, uint16_t type, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_autobuild_ethernet (const uint8_t *dst, uint16_t type, libnet_t *l)
- LIBNET_API libnet_ptag_t libnet_build_fddi (uint8_t fc, const uint8_t *dst, const uint8_t *src, uint8_t dsap, uint8_t ssap, uint8_t cf, const uint8_t *oui, uint16_t type, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_autobuild_fddi (uint8_t fc, const uint8_t *dst, uint8_t dsap, uint8_t ssap, uint8_t cf, const uint8_t *oui, uint16_t type, libnet_t *l)
- LIBNET_API libnet_ptag_t libnet_build_arp (uint16_t hrd, uint16_t pro, uint8_t hln, uint8_t pln, uint16_t op, const uint8_t *sha, const uint8_t *spa, const uint8_t *tha, const uint8_t *tpa, const uint8_t *payload, uint32↩ _t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_autobuild_arp (uint16_t op, const uint8_t *sha, const uint8_t *spa, const uint8_t *tha, uint8_t *tpa, libnet_t *l)
- LIBNET_API libnet_ptag_t libnet_build_tcp (uint16_t sp, uint16_t dp, uint32_t seq, uint32_t ack, uint8_↩ t control, uint16_t win, uint16_t sum, uint16_t urg, uint16_t len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_tcp_options (const uint8_t *options, uint32_t options_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_udp (uint16_t sp, uint16_t dp, uint16_t len, uint16_t sum, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_cdp (uint8_t version, uint8_t ttl, uint16_t sum, uint16_t type, uint16_t value_s, const uint8_t *value, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv4_echo (uint8_t type, uint8_t code, uint16_t sum, uint16_t id, uint16_t seq, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv4_mask (uint8_t type, uint8_t code, uint16_t sum, uint16_t id, uint16_t seq, uint32_t mask, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv4_unreach (uint8_t type, uint8_t code, uint16_t sum, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv4_redirect (uint8_t type, uint8_t code, uint16_t sum, uint32_t gateway, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv4_timeexceed (uint8_t type, uint8_t code, uint16_t sum, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv4_timestamp (uint8_t type, uint8_t code, uint16_t sum, uint16↩ _t id, uint16_t seq, uint32_t otime, uint32_t rtime, uint32_t ttime, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv6_echo (uint8_t type, uint8_t code, uint16_t sum, uint16_t id, uint16_t seq, uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv6_unreach (uint8_t type, uint8_t code, uint16_t sum, uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv6_ndp_nsol (uint8_t type, uint8_t code, uint16_t sum, struct libnet_in6_addr target, uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv6_ndp_nadv (uint8_t type, uint8_t code, uint16_t sum, uint32↩ _t flags, struct libnet_in6_addr target, uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_icmpv6_ndp_opt (uint8_t type, uint8_t *option, uint32_t option_↩ s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_igmp (uint8_t type, uint8_t reserved, uint16_t sum, uint32_t ip, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipv4 (uint16_t ip_len, uint8_t tos, uint16_t id, uint16_t frag, uint8_t ttl, uint8_t prot, uint16_t sum, uint32_t src, uint32_t dst, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)

- LIBNET_API libnet_ptag_t libnet_build_ipv4_options (const uint8_t *options, uint32_t options_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_autobuild_ipv4 (uint16_t len, uint8_t prot, uint32_t dst, libnet_t *l)
- LIBNET_API libnet_ptag_t libnet_build_ipv6 (uint8_t tc, uint32_t fl, uint16_t len, uint8_t nh, uint8_t hl, struct libnet_in6_addr src, struct libnet_in6_addr dst, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipv6_frag (uint8_t nh, uint8_t reserved, uint16_t frag, uint32_t id, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipv6_routing (uint8_t nh, uint8_t len, uint8_t rtype, uint8_t segments, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipv6_destopts (uint8_t nh, uint8_t len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipv6_hbhopts (uint8_t nh, uint8_t len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_autobuild_ipv6 (uint16_t len, uint8_t nh, struct libnet_in6_addr dst, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_isl (uint8_t *dhost, uint8_t type, uint8_t user, uint8_t *shost, uint16_t len, const uint8_t *snap, uint16_t vid, uint16_t portindex, uint16_t reserved, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipsec_esp_hdr (uint32_t spi, uint32_t seq, uint32_t iv, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipsec_esp_ftr (uint8_t len, uint8_t nh, int8_t *auth, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ipsec_ah (uint8_t nh, uint8_t len, uint16_t res, uint32_t spi, uint32_t seq, uint32_t auth, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_dnsv4 (uint16_t h_len, uint16_t id, uint16_t flags, uint16_t num_q, uint16_t num_anws_rr, uint16_t num_auth_rr, uint16_t num_addi_rr, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_rip (uint8_t cmd, uint8_t version, uint16_t rd, uint16_t af, uint16_t rt, uint32_t addr, uint32_t mask, uint32_t next_hop, uint32_t metric, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_rpc_call (uint32_t rm, uint32_t xid, uint32_t prog_num, uint32_t prog_vers, uint32_t procedure, uint32_t cflavor, uint32_t clength, uint8_t *cdata, uint32_t vflavor, uint32_t vlength, const uint8_t *vdata, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_stp_conf (uint16_t id, uint8_t version, uint8_t bpdu_type, uint8_t flags, const uint8_t *root_id, uint32_t root_pc, const uint8_t *bridge_id, uint16_t port_id, uint16_t message_age, uint16_t max_age, uint16_t hello_time, uint16_t f_delay, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_stp_tcn (uint16_t id, uint8_t version, uint8_t bpdu_type, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_token_ring (uint8_t ac, uint8_t fc, const uint8_t *dst, const uint8_t *src, uint8_t dsap, uint8_t ssap, uint8_t cf, const uint8_t *oui, uint16_t type, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_autobuild_token_ring (uint8_t ac, uint8_t fc, const uint8_t *dst, uint8_t dsap, uint8_t ssap, uint8_t cf, const uint8_t *oui, uint16_t type, libnet_t *l)
- LIBNET_API libnet_ptag_t libnet_build_vrrp (uint8_t version, uint8_t type, uint8_t vrouter_id, uint8_t priority, uint8_t ip_count, uint8_t auth_type, uint8_t advert_int, uint16_t sum, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_mpls (uint32_t label, uint8_t experimental, uint8_t bos, uint8_t ttl, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ntp (uint8_t leap_indicator, uint8_t version, uint8_t mode, uint8_t stratum, uint8_t poll, uint8_t precision, uint16_t delay_int, uint16_t delay_frac, uint16_t dispersion_int, uint16_t dispersion_frac, uint32_t reference_id, uint32_t ref_ts_int, uint32_t ref_ts_frac, uint32_t orig_ts_int, uint32_t orig_ts_frac, uint32_t rec_ts_int, uint32_t rec_ts_frac, uint32_t xmt_ts_int, uint32_t xmt_ts_frac, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2 (uint16_t len, uint8_t type, uint32_t rtr_id, uint32_t area_id, uint16_t sum, uint16_t autype, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)

- LIBNET_API libnet_ptag_t libnet_build_ospfv2_hello (uint32_t netmask, uint16_t interval, uint8_t opts, uint8_t priority, uint32_t dead_int, uint32_t des_rtr, uint32_t bkup_rtr, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_dbd (uint16_t dgram_len, uint8_t opts, uint8_t type, uint32_t seqnum, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsr (uint32_t type, uint32_t lsid, uint32_t advrtr, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsu (uint32_t num, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa (uint16_t age, uint8_t opts, uint8_t type, uint32_t lsid, uint32_t advrtr, uint32_t seqnum, uint16_t sum, uint16_t len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_rtr (uint16_t flags, uint16_t num, uint32_t id, uint32_t data, uint8_t type, uint8_t tos, uint16_t metric, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_net (uint32_t nmask, uint32_t rtrid, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_sum (uint32_t nmask, uint32_t metric, uint32_t tos, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_as (uint32_t nmask, uint32_t metric, uint32_t fwdaddr, uint32_t tag, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_data (const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_dhcpv4 (uint8_t opcode, uint8_t htype, uint8_t hlen, uint8_t hopcount, uint32_t xid, uint16_t secs, uint16_t flags, uint32_t cip, uint32_t yip, uint32_t sip, uint32_t gip, const uint8_t *chaddr, const char *sname, const char *file, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_bootpv4 (uint8_t opcode, uint8_t htype, uint8_t hlen, uint8_t hopcount, uint32_t xid, uint16_t secs, uint16_t flags, uint32_t cip, uint32_t yip, uint32_t sip, uint32_t gip, const uint8_t *chaddr, const char *sname, const char *file, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API uint32_t libnet_getgre_length (uint16_t fv)
- LIBNET_API libnet_ptag_t libnet_build_gre (uint16_t fv, uint16_t type, uint16_t sum, uint16_t offset, uint32_t key, uint32_t seq, uint16_t len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_egre (uint16_t fv, uint16_t type, uint16_t sum, uint16_t offset, uint32_t key, uint32_t seq, uint16_t len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_gre_sre (uint16_t af, uint8_t offset, uint8_t length, uint8_t *routing, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_gre_last_sre (libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_bgp4_header (uint8_t marker[LIBNET_BGP4_MARKER_SIZE], uint16_t len, uint8_t type, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_bgp4_open (uint8_t version, uint16_t src_as, uint16_t hold_time, uint32_t bgp_id, uint8_t opt_len, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_bgp4_update (uint16_t unfeasible_rt_len, const uint8_t *withdrawn_rt, uint16_t total_path_attr_len, const uint8_t *path_attributes, uint16_t info_len, uint8_t *reachability_info, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_bgp4_notification (uint8_t err_code, uint8_t err_subcode, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_sebek (uint32_t magic, uint16_t version, uint16_t type, uint32_t counter, uint32_t time_sec, uint32_t time_usec, uint32_t pid, uint32_t uid, uint32_t fd, uint8_t cmd[SEBEK_CMD_LENGTH], uint32_t length, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_hsrp (uint8_t version, uint8_t opcode, uint8_t state, uint8_t hello_time, uint8_t hold_time, uint8_t priority, uint8_t group, uint8_t reserved, uint8_t authdata[HSRP_AUTHDATA_LENGTH], uint32_t virtual_ip, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)
- LIBNET_API libnet_ptag_t libnet_build_link (const uint8_t *dst, const uint8_t *src, const uint8_t *oui, uint16_t type, const uint8_t *payload, uint32_t payload_s, libnet_t *l, libnet_ptag_t ptag)

- LIBNET_API libnet_ptag_t libnet_autobuild_link (const uint8_t *dst, const uint8_t *oui, uint16_t type, libnet_t *l)
- LIBNET_API int libnet_write (libnet_t *l)
- LIBNET_API uint32_t libnet_get_ipaddr4 (libnet_t *l)
- LIBNET_API struct libnet_in6_addr libnet_get_ipaddr6 (libnet_t *l)
- LIBNET_API struct libnet_ether_addr * libnet_get_hwaddr (libnet_t *l)
- LIBNET_API uint8_t * libnet_hex_aton (const char *s, int *len)
- LIBNET_API const char * libnet_version (void)
- LIBNET_API int libnet_adv_cull_packet (libnet_t *l, uint8_t **packet, uint32_t *packet_s)
- LIBNET_API int libnet_adv_cull_header (libnet_t *l, libnet_ptag_t ptag, uint8_t **header, uint32_t *header_s)
- LIBNET_API int libnet_adv_write_link (libnet_t *l, const uint8_t *packet, uint32_t packet_s)
- LIBNET_API int libnet_adv_write_raw_ipv4 (libnet_t *l, const uint8_t *packet, uint32_t packet_s)
- LIBNET_API void libnet_adv_free_packet (libnet_t *l, uint8_t *packet)
- int libnet_cq_add (libnet_t *l, char *label)
- LIBNET_API libnet_t * libnet_cq_remove (libnet_t *l)
- LIBNET_API libnet_t * libnet_cq_remove_by_label (char *label)
- LIBNET_API const char * libnet_cq_getlabel (libnet_t *l)
- LIBNET_API libnet_t * libnet_cq_find_by_label (char *label)
- LIBNET_API void libnet_cq_destroy (void)
- LIBNET_API libnet_t * libnet_cq_head (void)
- LIBNET_API int libnet_cq_last (void)
- LIBNET_API libnet_t * libnet_cq_next (void)
- LIBNET_API uint32_t libnet_cq_size (void)
- LIBNET_API uint32_t libnet_cq_end_loop (void)
- LIBNET_API void libnet_diag_dump_context (libnet_t *l)
- LIBNET_API void libnet_diag_dump_pblock (libnet_t *l)
- LIBNET_API char * libnet_diag_dump_pblock_type (uint8_t type)
- void libnet_diag_dump_hex (const uint8_t *packet, uint32_t len, int swap, FILE *stream)

3.2.1 Detailed Description

libnet exported function prototypes

3.2.2 Function Documentation

3.2.2.1 libnet_addr2name4()

```
LIBNET_API char* libnet_addr2name4 (
            uint32_t in,
            uint8_t use_name )
```

Takes a network byte ordered IPv4 address and returns a pointer to either a canonical DNS name (if it has one) or a string of dotted decimals. This may incur a DNS lookup if the hostname and mode is set to LIBNET_RESOLVE. If mode is set to LIBNET_DONT_RESOLVE, no DNS lookup will be performed and the function will return a pointer to a dotted decimal string. The function cannot fail – if no canonical name exists, it will fall back on returning a dotted decimal string. This function is non-reentrant.

Parameters

in	network byte ordered IPv4 address
use_name	LIBNET_RESOLVE or LIBNET_DONT_RESOLVE

Returns

a pointer to presentation format string

3.2.2.2 libnet_addr2name6_r()

```
LIBNET_API void libnet_addr2name6_r (
             struct libnet_in6_addr addr,
             uint8_t use_name,
             char * host_name,
             int host_name_len )
```

Should document this baby right here.

3.2.2.3 libnet_adv_cull_header()

```
LIBNET_API int libnet_adv_cull_header (
             libnet_t * l,
             libnet_ptag_t ptag,
             uint8_t ** header,
             uint32_t * header_s )
```

[Advanced Interface] Pulls the header from the specified ptag from the given libnet context. This function is part of the advanced interface and is only available when libnet is initialized in advanced mode. If the function fails libnet_geterror() can tell you why.

Parameters

l	pointer to a libnet context
ptag	the ptag referencing the header to pull
header	will contain the header
header_s	will contain the header size

Return values

1	on success
-1	on failure

3.2.2.4 libnet_adv_cull_packet()

```
LIBNET_API int libnet_adv_cull_packet (
             libnet_t * l,
             uint8_t ** packet,
             uint32_t * packet_s )
```

[Advanced Interface] Yanks a prebuilt, wire-ready packet from the given libnet context. If libnet was configured to do so (which it is by default) the packet will have all checksums written in. This function is part of the advanced interface and is only available when libnet is initialized in advanced mode. It is important to note that the function performs an implicit malloc() and a corresponding call to libnet_adv_free_packet() should be made to free the memory packet occupies. If the function fails libnet_geterror() can tell you why.

Parameters

l	pointer to a libnet context
packet	will contain the wire-ready packet
packet_s	will contain the packet size

Return values

1	on success
-1	on failure

3.2.2.5 libnet_adv_free_packet()

```
LIBNET_API void libnet_adv_free_packet (
             libnet_t * l,
             uint8_t * packet )
```

[Advanced Interface] Frees the memory allocated when libnet_adv_cull_packet() is called.

Parameters

l	pointer to a libnet context
packet	a pointer to the packet to free

3.2.2.6 libnet_adv_write_link()

```
LIBNET_API int libnet_adv_write_link (
             libnet_t * l,
             const uint8_t * packet,
             uint32_t packet_s )
```

[Advanced Interface] Writes a packet the network at the link layer. This function is useful to write a packet that has been constructed by hand by the application programmer or, more commonly, to write a packet that has been returned by a call to libnet_adv_cull_packet(). This function is part of the advanced interface and is only available when libnet is initialized in advanced mode. If the function fails libnet_geterror() can tell you why.

Parameters

l	pointer to a libnet context
packet	a pointer to the packet to inject
packet_s	the size of the packet

Returns

the number of bytes written

Return values

-1	on failure

3.2.2.7 libnet_adv_write_raw_ipv4()

```
LIBNET_API int libnet_adv_write_raw_ipv4 (
             libnet_t * l,
             const uint8_t * packet,
             uint32_t packet_s )
```

[Advanced Interface] Writes a packet the network at the raw socket layer. This function is useful to write a packet that has been constructed by hand by the application programmer or, more commonly, to write a packet that has been returned by a call to libnet_adv_cull_packet(). This function is part of the advanced interface and is only available when libnet is initialized in advanced mode. If the function fails libnet_geterror() can tell you why.

Parameters

l	pointer to a libnet context
packet	a pointer to the packet to inject
packet_s	the size of the packet

Returns

the number of bytes written

Return values

-1	on failure

3.2.2.8 libnet_autobuild_arp()

```
LIBNET_API libnet_ptag_t libnet_autobuild_arp (
             uint16_t op,
             const uint8_t * sha,
             const uint8_t * spa,
             const uint8_t * tha,
             uint8_t * tpa,
             libnet_t * l )
```

Autouilds an Address Resolution Protocol (ARP) header. Depending on the op value, the function builds one of several different types of RFC 826 or RFC 903 RARP packets.

Parameters

op	ARP operation type
sha	sender's hardware address
spa	sender's protocol address
tha	target hardware address
tpa	targer protocol address
l	pointer to a libnet context

Returns

protocol tag value on success

Return values

-1	on error

3.2.2.9 libnet_autobuild_ethernet()

```
LIBNET_API libnet_ptag_t libnet_autobuild_ethernet (
             const uint8_t * dst,
             uint16_t type,
             libnet_t * l )
```

Autobuilds an Ethernet header. The RFC 894 Ethernet II header is almost identical to the IEEE 802.3 header, with the exception that the field immediately following the source address holds the layer 3 protocol (as opposed to frame's length). You should only use this function when libnet is initialized with the LIBNET_LINK interface.

Parameters

dst	destination ethernet address
type	upper layer protocol type
l	pointer to a libnet context

Returns

protocol tag value on success

Return values

-1	on error

3.2.2.10 libnet_autobuild_fddi()

```
LIBNET_API libnet_ptag_t libnet_autobuild_fddi (
             uint8_t fc,
             const uint8_t * dst,
             uint8_t dsap,
             uint8_t ssap,
             uint8_t cf,
             const uint8_t * oui,
             uint16_t type,
             libnet_t * l )
```

Autobuilds a Fiber Distributed Data Interface (FDDI) header.

Parameters

fc	class format and priority
dst	destination fddi address
dsap	destination service access point
ssap	source service access point
cf	cf
oui	IEEE organizational code
type	upper layer protocol
l	pointer to a libnet context

Returns

protocol tag value on success

Return values

-1	on error

3.2.2.11 libnet_autobuild_ipv4()

```
LIBNET_API libnet_ptag_t libnet_autobuild_ipv4 (
             uint16_t len,
```

```
            uint8_t prot,
            uint32_t dst,
            libnet_t * l )
```

Autobuilds a version 4 Internet Protocol (IP) header. The function is useful to build an IP header quickly when you do not need a granular level of control. The function takes the same len, prot, and dst arguments as libnet_build_ipv4(). The function does not accept a ptag argument, but it does return a ptag. In other words, you can use it to build a new IP header but not to modify an existing one.

Parameters

len	total length of the IP packet including all subsequent data
prot	upper layer protocol
dst	destination IPv4 address (little endian)
l	pointer to a libnet context

Returns

protocol tag value on success

Return values

-1	on error

3.2.2.12 libnet_autobuild_ipv6()

```
LIBNET_API libnet_ptag_t libnet_autobuild_ipv6 (
            uint16_t len,
            uint8_t nh,
            struct libnet_in6_addr dst,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Autobuilds a version 6 RFC 2460 Internet Protocol (IP) header. The function is useful to build an IP header quickly when you do not need a granular level of control. The function takes the same len, nh, and dst arguments as libnet_build_ipv4(). The function does not accept a ptag argument, but it does return a ptag. In other words, you can use it to build a new IP header but not to modify an existing one. This function requires libnet_get_ipaddr6(), which is not yet implemented for Win32 platforms.

Parameters

len	length
nh	next header
dst	destination IPv6 address
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.2.13 libnet_autobuild_link()

```
LIBNET_API libnet_ptag_t libnet_autobuild_link (
             const uint8_t * dst,
             const uint8_t * oui,
             uint16_t type,
             libnet_t * l )
```

Automatically builds a link layer header for an initialized l. The function determines the proper link layer header format from how l was initialized. The function current supports Ethernet and Token Ring link layers.

Parameters

dst	the destination MAC address
oui	Organizationally Unique Identifier (unused for Ethernet)
type	the upper layer protocol type
l	pointer to a libnet context

Returns

protocol tag value on success

Return values

-1	on error

3.2.2.14 libnet_autobuild_token_ring()

```
LIBNET_API libnet_ptag_t libnet_autobuild_token_ring (
             uint8_t ac,
             uint8_t fc,
             const uint8_t * dst,
             uint8_t dsap,
             uint8_t ssap,
             uint8_t cf,
             const uint8_t * oui,
             uint16_t type,
             libnet_t * l )
```

Auto-builds a token ring header.

Parameters

ac	access control
fc	frame control
dst	destination address
dsap	destination service access point
ssap	source service access point
cf	control field
oui	Organizationally Unique Identifier
type	upper layer protocol type
l	pointer to a libnet context

Returns

protocol tag value on success

Return values

-1	on error

3.2.2.15 libnet_build_802_1q()

```
LIBNET_API libnet_ptag_t libnet_build_802_1q (
             const uint8_t * dst,
             const uint8_t * src,
             uint16_t tpi,
             uint8_t priority,
             uint8_t cfi,
             uint16_t vlan_id,
             uint16_t len_proto,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

3.2.3 Packet Builder Functions

The core of libnet is the platform-independent packet-building functionality. These functions enable an application programmer to build protocol headers (and data) in a simple and consistent manner without having to worry (too much) about low-level network odds and ends. Each libnet_build() function builds a piece of a packet (generally a protocol header). While it is perfectly possible to build an entire, ready-to-transmit packet with a single call to a libnet_build() function, generally more than one builder-class function call is required to construct a full packet. A complete wire-ready packet generally consists of more than one piece. Every function that builds a protocol header takes a series of arguments roughly corresponding to the header values as they appear on the wire. This process is intuitive but often makes for functions with huge prototypes and large stack frames. One important thing to note is that you must call these functions in order, corresponding to how they should appear on the wire (from the highest protocol layer on down). This building process is intuitive; it approximates what happens in an operating system kernel. In other words, to build a Network Time Protocol (NTP) packet by using the link-layer interface, the application programmer would call the libnet_build() functions in the following order:

1. libnet_build_ntp()
2. libnet_build_udp()
3. libnet_build_ipv4()
4. libnet_build_ethernet() This ordering is essential for libnet 1.1.x to properly link together the packet internally (previous libnet versions did not have the requirement).

3.2.3.1 The Payload Interface

The payload interface specifies an optional way to include data directly after the protocol header in question. You can use this function for a variety of purposes, including the following:

- Including additional or arbitrary protocol header information that is not available from a libnet interface
- Including a packet payload (data segment)
- Building another protocol header that is not available from a libnet interface To employ the interface, the application programmer should construct the i payload data and pass a const uint8_t * to this data and its size to the desired libnet_build() function. Libnet handles the rest.

It is important to note that some functions (notably the IPv6 builders) do use the payload interface to specify variable length but ostensibly non-optional data. See the individual libnet_build_ipv6*() functions for more information.

3.2.3.2 Protocol Tags and Packet Builder Return Values

Libnet uses the protocol tag (ptag) to identify individual pieces of a packet after being created. A new ptag results every time a libnet_build() function with an empty (0) ptag argument completes successfully. This new ptag now refers to the packet piece just created. The application programmer's responsibility is to save this value if he or she plans to modify this particular portion later on in the program. If the application programmer needs to modify some portion of that particular packet piece again, he or she calls the same libnet_build() function specifying the saved ptag argument. Libnet then searches for that packet piece and modifies it rather than creating a new one. Upon failure for any reason, libnet_build() functions return -1; libnet_geterror() tells you why. Builds an IEEE 802.1q VLAN tagging header. Depending on the value of len_proto, the function wraps the 802.1q header inside either an IEEE 802.3 header or an RFC 894 Ethernet II (DIX) header (both resulting in an 18-byte frame). If len is 1500 or less, most receiving protocol stacks parse the frame as an IEEE 802.3 encapsulated frame. If len is one of the Ethernet type values, most protocol stacks parse the frame as an RFC 894 Ethernet II encapsulated frame. Note the length value is calculated without the 802.1q header of 18 bytes.

Parameters

dst	pointer to a six byte source ethernet address
src	pointer to a six byte destination ethernet address
tpi	tag protocol identifier
priority	priority
cfi	canonical format indicator
vlan_id	vlan identifier
len_proto	length (802.3) protocol (Ethernet II)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.3 libnet_build_802_1x()

```
LIBNET_API libnet_ptag_t libnet_build_802_1x (
             uint8_t eap_ver,
             uint8_t eap_type,
             uint16_t length,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IEEE 802.1x extended authentication protocol header.

Parameters

eap_ver	the EAP version
eap_type	the EAP type
length	frame length
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.4 libnet_build_802_2()

```
LIBNET_API libnet_ptag_t libnet_build_802_2 (
             uint8_t dsap,
```

```
            uint8_t ssap,
            uint8_t control,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an IEEE 802.2 LLC header.

Parameters

dsap	destination service access point
ssap	source service access point
control	control field
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.5 libnet_build_802_2snap()

```
LIBNET_API libnet_ptag_t libnet_build_802_2snap (
            uint8_t dsap,
            uint8_t ssap,
            uint8_t control,
            uint8_t * oui,
            uint16_t type,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an IEEE 802.2 LLC SNAP header.

Parameters

dsap	destination service access point
ssap	source service access point
control	control field
oui	Organizationally Unique Identifier
type	upper layer protocol

Parameters

payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.6 libnet_build_802_3()

```
LIBNET_API libnet_ptag_t libnet_build_802_3 (
             const uint8_t * dst,
             const uint8_t * src,
             uint16_t len,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IEEE 802.3 header. The 802.3 header is almost identical to the RFC 894 Ethernet II header, the exception being that the field immediately following the source address holds the frame's length (as opposed to the layer 3 protocol). You should only use this function when libnet is initialized with the LIBNET_LINK interface.

Parameters

dst	destination ethernet address
src	source ethernet address
len	frame length sans header
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.7 libnet_build_arp()

```
LIBNET_API libnet_ptag_t libnet_build_arp (
             uint16_t hrd,
             uint16_t pro,
             uint8_t hln,
             uint8_t pln,
             uint16_t op,
             const uint8_t * sha,
             const uint8_t * spa,
             const uint8_t * tha,
             const uint8_t * tpa,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an Address Resolution Protocol (ARP) header. Depending on the op value, the function builds one of several different types of RFC 826 or RFC 903 RARP packets.

Parameters

hrd	hardware address format
pro	protocol address format
hln	hardware address length
pln	protocol address length
op	ARP operation type
sha	sender's hardware address
spa	sender's protocol address
tha	target hardware address
tpa	targer protocol address
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.8 libnet_build_bgp4_header()

```
LIBNET_API libnet_ptag_t libnet_build_bgp4_header (
             uint8_t marker[LIBNET_BGP4_MARKER_SIZE],
             uint16_t len,
             uint8_t type,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an RFC 1771 Border Gateway Protocol 4 (BGP-4) header. The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of Autonomous Systems (ASs) that reachability information traverses. This information is sufficient to construct a graph of AS connectivity from which routing loops may be pruned and some policy decisions at the AS level may be enforced. This function builds the base BGP header which is used as a preamble before any other BGP header. For example, a BGP KEEPALIVE message may be built with only this function, while an error notification requires a subsequent call to libnet_build_bgp4_notification.

Parameters

marker	a value the receiver can predict (if the message type is not BGP OPEN, or no authentication is used, these 16 bytes are normally set as all ones)
len	total length of the BGP message, including the header
type	type code of the message (OPEN, UPDATE, NOTIFICATION or KEEPALIVE)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.9 libnet_build_bgp4_notification()

```
LIBNET_API libnet_ptag_t libnet_build_bgp4_notification (
             uint8_t err_code,
             uint8_t err_subcode,
             const uint8_t * payload,
             uint32_t payload_s,
```

```
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an RFC 1771 Border Gateway Protocol 4 (BGP-4) notification header. A NOTIFICATION message is sent when an error condition is detected. Specific error information may be passed through the payload interface.

Parameters

err_code	type of notification
err_subcode	more specific information about the reported error.
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.10 libnet_build_bgp4_open()

```
LIBNET_API libnet_ptag_t libnet_build_bgp4_open (
            uint8_t version,
            uint16_t src_as,
            uint16_t hold_time,
            uint32_t bgp_id,
            uint8_t opt_len,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an RFC 1771 Border Gateway Protocol 4 (BGP-4) OPEN header. This is the first message sent by each side of a BGP connection. The optional parameters options should be constructed using the payload interface (see RFC 1771 for the options structures).

Parameters

version	protocol version (should be set to 4)
src_as	Autonomous System of the sender
hold_time	used to compute the maximum allowed time between the receipt of KEEPALIVE, and/or UPDATE messages by the sender
bgp_id	BGP identifier of the sender
opt_len	total length of the optional parameters field in bytes
payload	optional payload or NULL

Parameters

payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.11 libnet_build_bgp4_update()

```
LIBNET_API libnet_ptag_t libnet_build_bgp4_update (
             uint16_t unfeasible_rt_len,
             const uint8_t * withdrawn_rt,
             uint16_t total_path_attr_len,
             const uint8_t * path_attributes,
             uint16_t info_len,
             uint8_t * reachability_info,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an RFC 1771 Border Gateway Protocol 4 (BGP-4) update header. Update messages are used to transfer routing information between BGP peers.

Parameters

unfeasible_rt_len	indicates the length of the (next) "withdrawn routes" field in bytes
withdrawn_rt	list of IP addresses prefixes for the routes that are being withdrawn; each IP address prefix is built as a 2-tuple <length (1 byte), prefix (variable)>
total_path_attr_len	indicates the length of the (next) "path attributes" field in bytes
path_attributes	each attribute is a 3-tuple <type (2 bytes), length, value>
info_len	indicates the length of the (next) "network layer reachability information" field in bytes (needed for internal memory size calculation)
reachability_info	2-tuples <length (1 byte), prefix (variable)>.
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.12 libnet_build_bootpv4()

```
LIBNET_API libnet_ptag_t libnet_build_bootpv4 (
             uint8_t opcode,
             uint8_t htype,
             uint8_t hlen,
             uint8_t hopcount,
             uint32_t xid,
             uint16_t secs,
             uint16_t flags,
             uint32_t cip,
             uint32_t yip,
             uint32_t sip,
             uint32_t gip,
             const uint8_t * chaddr,
             const char * sname,
             const char * file,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Parameters

opcode	
htype	
hlen	
hopcount	
xid	
secs	
flags	
cip	
yip	
sip	
gip	
chaddr	client hardware address, length is hlen
sname	server host name, a null terminated string
file	boot file name, a null terminated string
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.13 libnet_build_cdp()

```
LIBNET_API libnet_ptag_t libnet_build_cdp (
             uint8_t version,
             uint8_t ttl,
             uint16_t sum,
             uint16_t type,
             uint16_t value_s,
             const uint8_t * value,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a Cisco Discovery Protocol (CDP) header. Cisco Systems designed CDP to aid in the network management of adjacent Cisco devices. The CDP protocol specifies data by using a type/length/value (TLV) setup. The first TLV can specified by using the functions type, length, and value arguments. To specify additional TLVs, the programmer could either use the payload interface or libnet_build_data() to construct them.

Parameters

version	CDP version
ttl	time to live (time information should be cached by recipient)
sum	checksum (0 for libnet to autofill)
type	type of data contained in value
value_s	length of value argument
value	the CDP information string
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.14 libnet_build_data()

```
LIBNET_API libnet_ptag_t libnet_build_data (
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a generic libnet protocol header. This is useful for including an optional payload to a packet that might need to change repeatedly inside of a loop. This won't work for TCP or IP payload, they have special types (this is probably a bug).

Parameters

payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.15 libnet_build_dhcpv4()

```
LIBNET_API libnet_ptag_t libnet_build_dhcpv4 (
             uint8_t opcode,
             uint8_t htype,
             uint8_t hlen,
             uint8_t hopcount,
             uint32_t xid,
             uint16_t secs,
             uint16_t flags,
             uint32_t cip,
             uint32_t yip,
             uint32_t sip,
             uint32_t gip,
             const uint8_t * chaddr,
             const char * sname,
             const char * file,
             const uint8_t * payload,
```

```
uint32_t payload_s,
libnet_t * l,
libnet_ptag_t ptag )
```

Parameters

opcode	
htype	
hlen	
hopcount	
xid	
secs	
flags	
cip	
yip	
sip	
gip	
chaddr	client hardware address, length is hlen
sname	server host name, a null terminated string
file	boot file name, a null terminated string
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.16 libnet_build_dnsv4()

```
LIBNET_API libnet_ptag_t libnet_build_dnsv4 (
             uint16_t h_len,
             uint16_t id,
             uint16_t flags,
             uint16_t num_q,
             uint16_t num_anws_rr,
             uint16_t num_auth_rr,
             uint16_t num_addi_rr,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an RFC 1035 version 4 DNS header. Additional DNS payload information should be specified using the payload interface.

Parameters

h_len	
id	DNS packet id
flags	control flags
num_q	number of questions
num_anws_rr	number of answer resource records
num_auth_rr	number of authority resource records
num_addi_rr	number of additional resource records
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.17 libnet_build_egre()

```
LIBNET_API libnet_ptag_t libnet_build_egre (
             uint16_t fv,
             uint16_t type,
             uint16_t sum,
             uint16_t offset,
             uint32_t key,
             uint32_t seq,
             uint16_t len,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Generic Routing Encapsulation (GRE - RFC 1701) is used to encapsulate any protocol. Hence, the IP part of the packet is usually referred as "delivery header". It is then followed by the GRE header and finally the encapsulated packet (IP or whatever). As GRE is very modular, the first GRE header describes the structure of the header, using bits and flag to specify which fields will be present in the header.

Parameters

fv	the 16 0 to 7: which fields are included in the header (checksum, seq. number, key, ...), bits 8 to 12: flag, bits 13 to 15: version.
type	which protocol is encapsulated (PPP, IP, ...)
sum	checksum (0 for libnet to autofill).

Parameters

offset	byte offset from the start of the routing field to the first byte of the SRE
key	inserted by the encapsulator to authenticate the source
seq	sequence number used by the receiver to sort the packets
len	size of the GRE packet
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.18 libnet_build_ethernet()

```
LIBNET_API libnet_ptag_t libnet_build_ethernet (
             const uint8_t * dst,
             const uint8_t * src,
             uint16_t type,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an Ethernet header. The RFC 894 Ethernet II header is almost identical to the IEEE 802.3 header, with the exception that the field immediately following the source address holds the layer 3 protocol (as opposed to frame's length). You should only use this function when libnet is initialized with the LIBNET_LINK interface.

Parameters

dst	destination ethernet address
src	source ethernet address
type	upper layer protocol type
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.19 libnet_build_fddi()

```
LIBNET_API libnet_ptag_t libnet_build_fddi (
             uint8_t fc,
             const uint8_t * dst,
             const uint8_t * src,
             uint8_t dsap,
             uint8_t ssap,
             uint8_t cf,
             const uint8_t * oui,
             uint16_t type,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a Fiber Distributed Data Interface (FDDI) header.

Parameters

fc	class format and priority
dst	destination fddi address
src	source fddi address
dsap	destination service access point
ssap	source service access point
cf	cf
oui	3 byte IEEE organizational code
type	upper layer protocol
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.20 libnet_build_gre()

```
LIBNET_API libnet_ptag_t libnet_build_gre (
            uint16_t fv,
            uint16_t type,
            uint16_t sum,
            uint16_t offset,
            uint32_t key,
            uint32_t seq,
            uint16_t len,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Generic Routing Encapsulation (GRE - RFC 1701) is used to encapsulate any protocol. Hence, the IP part of the packet is usually referred as "delivery header". It is then followed by the GRE header and finally the encapsulated packet (IP or whatever). As GRE is very modular, the first GRE header describes the structure of the header, using bits and flag to specify which fields will be present in the header.

Parameters

fv	the 16 0 to 7: which fields are included in the header (checksum, seq. number, key, ...), bits 8 to 12: flag, bits 13 to 15: version.
type	which protocol is encapsulated (PPP, IP, ...)
sum	checksum (0 for libnet to autofill).
offset	byte offset from the start of the routing field to the first byte of the SRE
key	inserted by the encapsulator to authenticate the source
seq	sequence number used by the receiver to sort the packets
len	size of the GRE packet
payload	
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.21 libnet_build_gre_last_sre()

```
LIBNET_API libnet_ptag_t libnet_build_gre_last_sre (
```

```
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.22 libnet_build_gre_sre()

```
LIBNET_API libnet_ptag_t libnet_build_gre_sre (
            uint16_t af,
            uint8_t offset,
            uint8_t length,
            uint8_t * routing,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

af	
offset	
length	
routing	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.23 libnet_build_hsrp()

```
LIBNET_API libnet_ptag_t libnet_build_hsrp (
             uint8_t version,
             uint8_t opcode,
             uint8_t state,
             uint8_t hello_time,
             uint8_t hold_time,
             uint8_t priority,
             uint8_t group,
             uint8_t reserved,
             uint8_t authdata[HSRP_AUTHDATA_LENGTH],
             uint32_t virtual_ip,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a HSRP header. HSRP is a Cisco propietary protocol defined in RFC 2281

Parameters

version	version of the HSRP messages
opcode	type of message
state	current state of the router
hello_time	period in seconds between hello messages
hold_time	seconds that the current hello message is valid
priority	priority for the election proccess
group	standby group
reserved	reserved field
authdata	password
virtual_ip	virtual ip address
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.24 libnet_build_icmpv4_echo()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv4_echo (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             uint16_t id,
             uint16_t seq,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 4 RFC 792 Internet Control Message Protocol (ICMP) echo request/reply header

Parameters

type	type of ICMP packet (should be ICMP_ECHOREPLY or ICMP_ECHO)
code	code of ICMP packet (should be 0)
sum	checksum (0 for libnet to autofill)
id	identification number
seq	packet sequence number
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.25 libnet_build_icmpv4_mask()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv4_mask (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             uint16_t id,
             uint16_t seq,
             uint32_t mask,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 4 RFC 792 Internet Control Message Protocol (ICMP) IP netmask request/reply header.

Parameters

type	type of ICMP packet (should be ICMP_MASKREQ or ICMP_MASKREPLY)
code	code of ICMP packet (should be 0)
sum	checksum (0 for libnet to autofill)
id	identification number
seq	packet sequence number
mask	subnet mask
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.26 libnet_build_icmpv4_redirect()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv4_redirect (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             uint32_t gateway,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 4 RFC 792 Internet Message Control Protocol (ICMP) redirect header. The IP header that caused the error message should be built by a previous call to libnet_build_ipv4().

Parameters

type	type of ICMP packet (should be ICMP_REDIRECT)
code	code of ICMP packet (should be one of the four redirect codes)
sum	checksum (0 for libnet to autofill)
gateway	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.27 libnet_build_icmpv4_timeexceed()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv4_timeexceed (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 4 RFC 792 Internet Control Message Protocol (ICMP) time exceeded header. The IP header that caused the error message should be built by a previous call to libnet_build_ipv4().

Parameters

type	type of ICMP packet (should be ICMP_TIMXCEED)
code	code of ICMP packet (ICMP_TIMXCEED_INTRANS / ICMP_TIMXCEED_REASS)
sum	checksum (0 for libnet to autofill)
payload	optional payload or NULL
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.28 libnet_build_icmpv4_timestamp()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv4_timestamp (
             uint8_t type,
```

```
            uint8_t code,
            uint16_t sum,
            uint16_t id,
            uint16_t seq,
            uint32_t otime,
            uint32_t rtime,
            uint32_t ttime,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an IP version 4 RFC 792 Internet Control Message Protocol (ICMP) timestamp request/reply header.

Parameters

type	type of ICMP packet (should be ICMP_TSTAMP or ICMP_TSTAMPREPLY)
code	code of ICMP packet (should be 0)
sum	checksum (0 for libnet to autofill)
id	identification number
seq	sequence number
otime	originate timestamp
rtime	receive timestamp
ttime	transmit timestamp
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.29 libnet_build_icmpv4_unreach()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv4_unreach (
            uint8_t type,
            uint8_t code,
            uint16_t sum,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an IP version 4 RFC 792 Internet Control Message Protocol (ICMP) unreachable header. The IP header that caused the error message should be built by a previous call to libnet_build_ipv4().

Parameters

type	type of ICMP packet (should be ICMP_UNREACH)
code	code of ICMP packet (should be one of the 16 unreachable codes)
sum	checksum (0 for libnet to autofill)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.30 libnet_build_icmpv6_echo()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv6_echo (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             uint16_t id,
             uint16_t seq,
             uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 6 RFC 4443 Internet Control Message Protocol (ICMP) echo or echo reply header.

Parameters

type	type of ICMP packet (should be ICMP6_ECHO_REQUEST or ICMP6_ECHO_REPLY)
code	code of ICMP packet (should be zero)
sum	checksum (0 for libnet to autofill)
id	echo id number
seq	echo sequence number
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.31 libnet_build_icmpv6_ndp_nadv()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv6_ndp_nadv (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             uint32_t flags,
             struct libnet_in6_addr target,
             uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 6 RFC 2461 Internet Control Message Protocol (ICMP) NDP neighbour advertisement header. Could be used with libnet_build_icmpv6_ndp_opt() and ND_OPT_TARGET_LINKADDR.

Parameters

type	type of ICMP packet (should be ND_NEIGHBOR_ADVERT)
code	code of ICMP packet (should be zero)
sum	checksum (0 for libnet to autofill)
flags	should be a bitwise or of any applicable ND_NA_FLAG_* flags
target	target ipv6 address
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.32 libnet_build_icmpv6_ndp_nsol()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv6_ndp_nsol (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             struct libnet_in6_addr target,
             uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 6 RFC 2461 Internet Control Message Protocol (ICMP) NDP neighbour solicitation header. Could be used with libnet_build_icmpv6_ndp_opt() and ICMPV6_NDP_OPT_SLLA.

Parameters

type	type of ICMP packet (should be ND_NEIGHBOR_SOLICIT)
code	code of ICMP packet (should be zero)
sum	checksum (0 for libnet to autofill)
target	target ipv6 address
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.33 libnet_build_icmpv6_ndp_opt()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv6_ndp_opt (
             uint8_t type,
             uint8_t * option,
             uint32_t option_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds ICMPv6 NDP options.

Parameters

type	one of ND_OPT_* types

Parameters

option	option data
option_s	size of option data (will be padded out to an 8-byte boundary)
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.34 libnet_build_icmpv6_unreach()

```
LIBNET_API libnet_ptag_t libnet_build_icmpv6_unreach (
             uint8_t type,
             uint8_t code,
             uint16_t sum,
             uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an IP version 6 RFC 4443 Internet Control Message Protocol (ICMP) unreachable header. The IP header that caused the error message should be built by a previous call to libnet_build_ipv6().

Parameters

type	type of ICMP packet (should be ICMP6_DST_UNREACH)
code	code of ICMP packet (should be one of the 5 ICMP6_DST_UNREACH_* codes)
sum	checksum (0 for libnet to autofill)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.35 libnet_build_igmp()

```
LIBNET_API libnet_ptag_t libnet_build_igmp (
             uint8_t type,
             uint8_t reserved,
             uint16_t sum,
             uint32_t ip,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an RFC 1112 Internet Group Memebership Protocol (IGMP) header.

Parameters

type	packet type
reserved	(should be 0 for IGMPv1)
sum	checksum (0 for libnet to autofill)
ip	IPv4 address (in standard/network byte order)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

Note

'reserved' was previously called 'code', which it is not, in any IGMP version.

3.2.3.36 libnet_build_ipsec_ah()

```
LIBNET_API libnet_ptag_t libnet_build_ipsec_ah (
             uint8_t nh,
```

```
            uint8_t len,
            uint16_t res,
            uint32_t spi,
            uint32_t seq,
            uint32_t auth,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an Internet Protocol Security Authentication header.

Parameters

nh	next header
len	payload length
res	reserved
spi	security parameter index
seq	sequence number
auth	authentication data
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.37 libnet_build_ipsec_esp_ftr()

```
LIBNET_API libnet_ptag_t libnet_build_ipsec_esp_ftr (
            uint8_t len,
            uint8_t nh,
            int8_t * auth,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an Internet Protocol Security Encapsulating Security Payload footer.

Parameters

len	padding length

Parameters

nh	next header
auth	authentication data
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.38 libnet_build_ipsec_esp_hdr()

```
LIBNET_API libnet_ptag_t libnet_build_ipsec_esp_hdr (
             uint32_t spi,
             uint32_t seq,
             uint32_t iv,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an Internet Protocol Security Encapsulating Security Payload header.

Parameters

spi	security parameter index
seq	ESP sequence number
iv	initialization vector
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.39 libnet_build_ipv4()

```
LIBNET_API libnet_ptag_t libnet_build_ipv4 (
             uint16_t ip_len,
             uint8_t tos,
             uint16_t id,
             uint16_t frag,
             uint8_t ttl,
             uint8_t prot,
             uint16_t sum,
             uint32_t src,
             uint32_t dst,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a version 4 RFC 791 Internet Protocol (IP) header.

Parameters

ip_len	total length of the IP packet including all subsequent data (subsequent data includes any IP options and IP options padding)
tos	type of service bits
id	IP identification number
frag	fragmentation bits and offset
ttl	time to live in the network
prot	upper layer protocol
sum	checksum (0 for libnet to autofill)
src	source IPv4 address (little endian)
dst	destination IPv4 address (little endian)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.40 libnet_build_ipv4_options()

```
LIBNET_API libnet_ptag_t libnet_build_ipv4_options (
            const uint8_t * options,
            uint32_t options_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an version 4 Internet Protocol (IP) options header. The function expects options to be a valid IP options string of size options_s, no larger than 40 bytes (the maximum size of an options string).

When building a chain, the options must be built, then the IPv4 header.

When updating a chain, if the block following the options is an IPv4 header, it's total length and header length will be updated if the options block size changes.

Parameters

options	byte string of IP options (it will be padded up to be an integral multiple of 32-bit words).
options_s	length of options string
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.41 libnet_build_ipv6()

```
LIBNET_API libnet_ptag_t libnet_build_ipv6 (
            uint8_t tc,
            uint32_t fl,
            uint16_t len,
            uint8_t nh,
            uint8_t hl,
            struct libnet_in6_addr src,
            struct libnet_in6_addr dst,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds a version 6 RFC 2460 Internet Protocol (IP) header.

Parameters

tc	traffic class
fl	flow label
len	total length of the IP packet
nh	next header
hl	hop limit
src	source IPv6 address
dst	destination IPv6 address
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.42 libnet_build_ipv6_destopts()

```
LIBNET_API libnet_ptag_t libnet_build_ipv6_destopts (
             uint8_t nh,
             uint8_t len,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a version 6 RFC 2460 Internet Protocol (IP) destination options header. This function is special in that it uses the payload interface to include the options data. The application programmer will build an IPv6 options byte string and pass it to the function using the payload interface.

Parameters

nh	next header
len	length of the header in 8-byte octets not including the first 8 octets
payload	options payload
payload_s	payload length
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.43 libnet_build_ipv6_frag()

```
LIBNET_API libnet_ptag_t libnet_build_ipv6_frag (
             uint8_t nh,
             uint8_t reserved,
             uint16_t frag,
             uint32_t id,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a version 6 RFC 2460 Internet Protocol (IP) fragmentation header.

Parameters

nh	next header
reserved	unused value... OR IS IT!
frag	fragmentation bits (ala ipv4)
id	packet identification
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.44 libnet_build_ipv6_hbhopts()

```
LIBNET_API libnet_ptag_t libnet_build_ipv6_hbhopts (
             uint8_t nh,
```

```
            uint8_t len,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds a version 6 RFC 2460 Internet Protocol (IP) hop by hop options header. This function is special in that it uses the payload interface to include the options data. The application programmer will build an IPv6 hop by hop options byte string and pass it to the function using the payload interface.

Parameters

nh	next header
len	length of the header in 8-byte octets not including the first 8 octets
payload	options payload
payload_s	payload length
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.45 libnet_build_ipv6_routing()

```
LIBNET_API libnet_ptag_t libnet_build_ipv6_routing (
            uint8_t nh,
            uint8_t len,
            uint8_t rtype,
            uint8_t segments,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds a version 6 RFC 2460 Internet Protocol (IP) routing header. This function is special in that it uses the payload interface to include the "type-specific data"; that is the routing information. Most often this will be a number of 128-bit IPv6 addresses. The application programmer will build a byte string of IPv6 address and pass them to the function using the payload interface.

Parameters

nh	next header
len	length of the header in 8-byte octets not including the first 8 octets
rtype	routing header type

Parameters

segments	number of routing segments that follow
payload	optional payload of routing information
payload_s	payload length
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.46 libnet_build_isl()

```
LIBNET_API libnet_ptag_t libnet_build_isl (
             uint8_t * dhost,
             uint8_t type,
             uint8_t user,
             uint8_t * shost,
             uint16_t len,
             const uint8_t * snap,
             uint16_t vid,
             uint16_t portindex,
             uint16_t reserved,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a Cisco Inter-Switch Link (ISL) header.

Parameters

dhost	destination address (should be 01:00:0c:00:00)
type	type of frame
user	user defined data
shost	source mac address
len	total length of the encapuslated packet less 18 bytes
snap	SNAP information (0xaaaa03 + vendor code)
vid	15 bit VLAN ID, 1 bit BPDU or CDP indicator
portindex	port index
reserved	used for FDDI and token ring
payload	optional payload or NULL

Parameters

payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.47 libnet_build_link()

```
LIBNET_API libnet_ptag_t libnet_build_link (
             const uint8_t * dst,
             const uint8_t * src,
             const uint8_t * oui,
             uint16_t type,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a link layer header for an initialized l. The function determines the proper link layer header format from how l was initialized. The function current supports Ethernet and Token Ring link layers.

Parameters

dst	the destination MAC address
src	the source MAC address
oui	Organizationally Unique Identifier (unused for Ethernet)
type	the upper layer protocol type
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.48 libnet_build_mpls()

```
LIBNET_API libnet_ptag_t libnet_build_mpls (
             uint32_t label,
             uint8_t experimental,
             uint8_t bos,
             uint8_t ttl,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an RFC 3032 Multi-Protocol Label Switching (MPLS) header.

Parameters

label	20-bit label value
experimental	3-bit reserved field
bos	1-bit bottom of stack identifier
ttl	time to live
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.49 libnet_build_ntp()

```
LIBNET_API libnet_ptag_t libnet_build_ntp (
             uint8_t leap_indicator,
             uint8_t version,
             uint8_t mode,
             uint8_t stratum,
             uint8_t poll,
```

```
uint8_t precision,
uint16_t delay_int,
uint16_t delay_frac,
uint16_t dispersion_int,
uint16_t dispersion_frac,
uint32_t reference_id,
uint32_t ref_ts_int,
uint32_t ref_ts_frac,
uint32_t orig_ts_int,
uint32_t orig_ts_frac,
uint32_t rec_ts_int,
uint32_t rec_ts_frac,
uint32_t xmt_ts_int,
uint32_t xmt_ts_frac,
const uint8_t * payload,
uint32_t payload_s,
libnet_t * l,
libnet_ptag_t ptag )
```

Builds an RFC 958 Network Time Protocol (NTP) header.

Parameters

leap_indicator	the leap indicator
version	NTP protocol version
mode	NTP mode
stratum	stratum
poll	polling interval
precision	precision
delay_int	delay interval
delay_frac	delay fraction
dispersion_int	dispersion interval
dispersion_frac	dispersion fraction
reference_id	reference id
ref_ts_int	reference timestamp integer
ref_ts_frac	reference timestamp fraction
orig_ts_int	original timestamp integer
orig_ts_frac	original timestamp fraction
rec_ts_int	receiver timestamp integer
rec_ts_frac	reciever timestamp fraction
xmt_ts_int	transmit timestamp integer
xmt_ts_frac	transmit timestamp integer
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.50 libnet_build_ospfv2()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2 (
             uint16_t len,
             uint8_t type,
             uint32_t rtr_id,
             uint32_t area_id,
             uint16_t sum,
             uint16_t autype,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Parameters

len	
type	
rtr_id	
area_id	
sum	
autype	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.51 libnet_build_ospfv2_dbd()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_dbd (
             uint16_t dgram_len,
             uint8_t opts,
```

```
            uint8_t type,
            uint32_t seqnum,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

dgram_len	
opts	
type	
seqnum	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.52 libnet_build_ospfv2_hello()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_hello (
            uint32_t netmask,
            uint16_t interval,
            uint8_t opts,
            uint8_t priority,
            uint32_t dead_int,
            uint32_t des_rtr,
            uint32_t bkup_rtr,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

netmask	
interval	
opts	
priority	
dead_int	
des_rtr	

Parameters

bkup_rtr	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.53 libnet_build_ospfv2_lsa()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa (
             uint16_t age,
             uint8_t opts,
             uint8_t type,
             uint32_t lsid,
             uint32_t advrtr,
             uint32_t seqnum,
             uint16_t sum,
             uint16_t len,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Parameters

age	
opts	
type	
lsid	
advrtr	
seqnum	
sum	
len	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.54 libnet_build_ospfv2_lsa_as()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_as (
             uint32_t nmask,
             uint32_t metric,
             uint32_t fwdaddr,
             uint32_t tag,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Parameters

nmask	
metric	
fwdaddr	
tag	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.55 libnet_build_ospfv2_lsa_net()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_net (
             uint32_t nmask,
             uint32_t rtrid,
```

```
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

nmask	
rtrid	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.56 libnet_build_ospfv2_lsa_rtr()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_rtr (
            uint16_t flags,
            uint16_t num,
            uint32_t id,
            uint32_t data,
            uint8_t type,
            uint8_t tos,
            uint16_t metric,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

flags	
num	
id	
data	
type	
tos	
metric	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.57 libnet_build_ospfv2_lsa_sum()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsa_sum (
             uint32_t nmask,
             uint32_t metric,
             uint32_t tos,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Parameters

nmask	
metric	
tos	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.58 libnet_build_ospfv2_lsr()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsr (
             uint32_t type,
             uint32_t lsid,
             uint32_t advrtr,
             const uint8_t * payload,
```

```
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

type	
lsid	
advrtr	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.59 libnet_build_ospfv2_lsu()

```
LIBNET_API libnet_ptag_t libnet_build_ospfv2_lsu (
            uint32_t num,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Parameters

num	
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.60 libnet_build_rip()

```
LIBNET_API libnet_ptag_t libnet_build_rip (
             uint8_t cmd,
             uint8_t version,
             uint16_t rd,
             uint16_t af,
             uint16_t rt,
             uint32_t addr,
             uint32_t mask,
             uint32_t next_hop,
             uint32_t metric,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a Routing Information Protocol header (RFCs 1058 and 2453).

Parameters

cmd	command
version	protocol version
rd	version one: 0, version two: routing domain
af	address family
rt	version one: 0, version two: route tag
addr	IPv4 address
mask	version one: 0, version two: subnet mask
next_hop	version one: 0, version two: next hop address
metric	routing metric
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.61 libnet_build_rpc_call()

```
LIBNET_API libnet_ptag_t libnet_build_rpc_call (
```

```
            uint32_t rm,
            uint32_t xid,
            uint32_t prog_num,
            uint32_t prog_vers,
            uint32_t procedure,
            uint32_t cflavor,
            uint32_t clength,
            uint8_t * cdata,
            uint32_t vflavor,
            uint32_t vlength,
            const uint8_t * vdata,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an Remote Procedure Call (Version 2) Call message header as specified in RFC 1831. This builder provides the option for specifying the record marking which is required when used with streaming protocols (TCP).

Parameters

rm	record marking indicating the position in a stream, 0 otherwise
xid	transaction identifier used to link calls and replies
prog_num	remote program specification typically between 0 - 1fffffff
prog_vers	remote program version specification
procedure	procedure to be performed by remote program
cflavor	authentication credential type
clength	credential length (should be 0)
cdata	opaque credential data (currently unused)
vflavor	authentication verifier type
vlength	verifier length (should be 0)
vdata	opaque verifier data (currently unused)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.62 libnet_build_sebek()

```
LIBNET_API libnet_ptag_t libnet_build_sebek (
            uint32_t magic,
```

```
            uint16_t version,
            uint16_t type,
            uint32_t counter,
            uint32_t time_sec,
            uint32_t time_usec,
            uint32_t pid,
            uint32_t uid,
            uint32_t fd,
            uint8_t cmd[SEBEK_CMD_LENGTH],
            uint32_t length,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds a Sebek header. The Sebek protocol was designed by the Honeynet Project as a transport mechanism for post-intrusion forensic data. More information may be found here: http://www.honeynet.org/papers/sebek.pdf.

Parameters

magic	identify packets that should be hidden
version	protocol version, currently 1
type	type of record (read data is type 0, write data is type 1)
counter	PDU counter used to identify when packet are lost
time_sec	seconds since EPOCH according to the honeypot
time_usec	residual microseconds
pid	PID
uid	UID
fd	FD
cmd	12 first characters of the command
length	length in bytes of the PDU's body
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.63 libnet_build_stp_conf()

```
LIBNET_API libnet_ptag_t libnet_build_stp_conf (
            uint16_t id,
```

```
            uint8_t version,
            uint8_t bpdu_type,
            uint8_t flags,
            const uint8_t * root_id,
            uint32_t root_pc,
            const uint8_t * bridge_id,
            uint16_t port_id,
            uint16_t message_age,
            uint16_t max_age,
            uint16_t hello_time,
            uint16_t f_delay,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an IEEE 802.1d Spanning Tree Protocol (STP) configuration header. STP frames are usually encapsulated inside of an 802.2 + 802.3 frame combination.

Parameters

id	protocol id
version	protocol version
bpdu_type	bridge protocol data unit type
flags	flags
root_id	root id
root_pc	root path cost
bridge_id	bridge id
port_id	port id
message_age	message age
max_age	max age
hello_time	hello time
f_delay	forward delay
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.64 libnet_build_stp_tcn()

```
LIBNET_API libnet_ptag_t libnet_build_stp_tcn (
            uint16_t id,
```

```
            uint8_t version,
            uint8_t bpdu_type,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an IEEE 802.1d Spanning Tree Protocol (STP) topology change notification header. STP frames are usually encapsulated inside of an 802.2 + 802.3 frame combination.

Parameters

id	protocol id
version	protocol version
bpdu_type	bridge protocol data unit type
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.65 libnet_build_tcp()

```
LIBNET_API libnet_ptag_t libnet_build_tcp (
            uint16_t sp,
            uint16_t dp,
            uint32_t seq,
            uint32_t ack,
            uint8_t control,
            uint16_t win,
            uint16_t sum,
            uint16_t urg,
            uint16_t len,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an RFC 793 Transmission Control Protocol (TCP) header.

Parameters

sp	source port
dp	destination port
seq	sequence number
ack	acknowledgement number
control	control flags
win	window size
sum	checksum (0 for libnet to autofill)
urg	urgent pointer
len	total length of the TCP packet (for checksum calculation)
payload	
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.66 libnet_build_tcp_options()

```
LIBNET_API libnet_ptag_t libnet_build_tcp_options (
             const uint8_t * options,
             uint32_t options_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an RFC 793 Transmission Control Protocol (TCP) options header. The function expects options to be a valid TCP options string of size options_s, which is no larger than 40 bytes (the maximum size of an options string). The function checks to ensure that the packet consists of a TCP header preceded by an IPv4 header, and that the addition of the options string would not result in a packet larger than 65,535 bytes (IPMAXPACKET). The function counts up the number of 32-bit words in the options string and adjusts the TCP header length value as necessary.

Parameters

options	byte string of TCP options
options_s	length of options string
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.67 libnet_build_token_ring()

```
LIBNET_API libnet_ptag_t libnet_build_token_ring (
             uint8_t ac,
             uint8_t fc,
             const uint8_t * dst,
             const uint8_t * src,
             uint8_t dsap,
             uint8_t ssap,
             uint8_t cf,
             const uint8_t * oui,
             uint16_t type,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds a token ring header.

Parameters

ac	access control
fc	frame control
dst	destination address
src	source address
dsap	destination service access point
ssap	source service access point
cf	control field
oui	Organizationally Unique Identifier
type	upper layer protocol type
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.68 libnet_build_udp()

```
LIBNET_API libnet_ptag_t libnet_build_udp (
             uint16_t sp,
             uint16_t dp,
             uint16_t len,
             uint16_t sum,
             const uint8_t * payload,
             uint32_t payload_s,
             libnet_t * l,
             libnet_ptag_t ptag )
```

Builds an RFC 768 User Datagram Protocol (UDP) header.

Parameters

sp	source port
dp	destination port
len	total length of the UDP packet
sum	checksum (0 for libnet to autofill)
payload	optional payload or NULL
payload_s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.69 libnet_build_vrrp()

```
LIBNET_API libnet_ptag_t libnet_build_vrrp (
             uint8_t version,
             uint8_t type,
             uint8_t vrouter_id,
```

```
            uint8_t priority,
            uint8_t ip_count,
            uint8_t auth_type,
            uint8_t advert_int,
            uint16_t sum,
            const uint8_t * payload,
            uint32_t payload_s,
            libnet_t * l,
            libnet_ptag_t ptag )
```

Builds an RFC 2338 Virtual Router Redundacy Protool (VRRP) header. Use the payload interface to specify address and autthentication information. To build a "legal" packet, the destination IPv4 address should be the multicast ∗ address 224.0.0.18, the IP TTL should be set to 255, and the IP protocol should be set to 112.

Parameters

version	VRRP version (should be 2)
type	VRRP packet type (should be 1 – ADVERTISEMENT)
vrouter↩ _id	virtual router identification
priority	priority (higher numbers indicate higher priority)
ip_count	number of IPv4 addresses contained in this advertisement
auth_type	type of authentication (0, 1, 2 – see RFC)
advert_int	interval between advertisements
sum	checksum (0 for libnet to autofill)
payload	optional payload or NULL
payload↩ _s	payload length or 0
l	pointer to a libnet context
ptag	protocol tag to modify an existing header, 0 to build a new one

Returns

protocol tag value on success

Return values

-1	on error

3.2.3.70 libnet_clear_packet()

```
LIBNET_API void libnet_clear_packet (
            libnet_t * l )
```

Clears the current packet referenced and frees all pblocks. Should be called when the programmer want to send a completely new packet of a different type using the same context.

Parameters

l	pointer to a libnet context

3.2.3.71 libnet_cq_add()

```
int libnet_cq_add (
             libnet_t * l,
             char * label )
```

[Context Queue] Adds a new context to the libnet context queue. If no queue exists, this function will create the queue and add the specified libnet context as the first entry on the list. The functions checks to ensure niether l nor label are NULL, and that label doesn't refer to an existing context already in the queue. Additionally, l should refer to a libnet context previously initialized with a call to libnet_init(). If the context queue in write locked, this function will fail.

Parameters

l	pointer to a libnet context
label	a canonical name given to recognize the new context, no longer than LIBNET_LABEL_SIZE

Return values

1	on success
-1	on failure

3.2.3.72 libnet_cq_destroy()

```
LIBNET_API void libnet_cq_destroy (
             void  )
```

[Context Queue] Destroys the entire context queue, calling libnet_destroy() on each member context.

3.2.3.73 libnet_cq_end_loop()

```
LIBNET_API uint32_t libnet_cq_end_loop (
             void  )
```

[Context Queue]

3.2.3.74 libnet_cq_find_by_label()

```
LIBNET_API libnet_t* libnet_cq_find_by_label (
             char * label )
```

[Context Queue] Locates a libnet context from the queue, indexed by a canonical label.

Parameters

label	canonical label of the libnet context to retrieve

Returns

the expected libnet context

Return values

NULL	on failure

3.2.3.75 libnet_cq_getlabel()

```
LIBNET_API const char* libnet_cq_getlabel (
             libnet_t * l )
```

[Context Queue] Returns the canonical label associated with the context.

Parameters

l	pointer to a libnet context

Returns

pointer to the libnet context's label

3.2.3.76 libnet_cq_head()

```
LIBNET_API libnet_t* libnet_cq_head (
             void  )
```

[Context Queue] Intiailizes the interator interface and set a write lock on the entire queue. This function is intended to be called just prior to interating through the entire list of contexts (with the probable intent of inject a series of packets in rapid succession). This function is often used as per the following:

for (l = libnet_cq_head(); libnet_cq_last(); l = libnet_cq_next()) { ... }

Much of the time, the application programmer will use the iterator as it is written above; as such, libnet provides a macro to do exactly that, for_each_context_in_cq(l). Warning: do not call the iterator more than once in a single loop.

Returns

the head of the context queue

3.2.3.77 libnet_cq_last()

```
LIBNET_API int libnet_cq_last (
             void  )
```

[Context Queue] Check whether the iterator is at the last context in the queue.

Return values

1	if at the end of the context queue
0	otherwise

3.2.3.78 libnet_cq_next()

```
LIBNET_API libnet_t* libnet_cq_next (
             void  )
```

[Context Queue] Get next context from the context queue.

Returns

the next context from the context queue

3.2.3.79 libnet_cq_remove()

```
LIBNET_API libnet_t* libnet_cq_remove (
             libnet_t * l )
```

[Context Queue] Removes a specified context from the libnet context queue by specifying the libnet context pointer. Note the function will remove the specified context from the context queue and cleanup internal memory from the queue, it is up to the application programmer to free the returned libnet context with a call to libnet_destroy(). Also, as it is not necessary to keep the libnet context pointer when initially adding it to the context queue, most application programmers will prefer to refer to entries on the context queue by canonical name and would use libnet_cq_↩ remove_by_label(). If the context queue is write locked, this function will fail.

Parameters

l	pointer to a libnet context

Returns

the pointer to the removed libnet context

Return values

NULL	on failure

3.2.3.80 libnet_cq_remove_by_label()

```
LIBNET_API libnet_t* libnet_cq_remove_by_label (
             char * label )
```

[Context Queue] Removes a specified context from the libnet context queue by specifying the canonical name. Note the function will remove the specified context from the context queue and cleanup internal memory from the queue, it is up to the application programmer to free the returned libnet context with a call to libnet_destroy(). If the context queue is write locked, this function will fail.

Parameters

label	canonical name of the context to remove

Returns

the pointer to the removed libnet context

Return values

NULL	on failure

3.2.3.81 libnet_cq_size()

```
LIBNET_API uint32_t libnet_cq_size (
             void  )
```

[Context Queue] Function returns the number of libnet contexts that are in the queue.

Returns

the number of libnet contexts currently in the queue

3.2.3.82 libnet_destroy()

```
LIBNET_API void libnet_destroy (
             libnet_t * l )
```

Shuts down the libnet session referenced by l. It closes the network interface and frees all internal memory structures associated with l.

Parameters

l	pointer to a libnet context

3.2.3.83 libnet_diag_dump_context()

```
LIBNET_API void libnet_diag_dump_context (
             libnet_t * l )
```

[Diagnostic] Prints the contents of the given context.

Parameters

l	pointer to a libnet context

3.2.3.84 libnet_diag_dump_hex()

```
void libnet_diag_dump_hex (
             const uint8_t * packet,
             uint32_t len,
             int swap,
             FILE * stream )
```

[Diagnostic] Function prints the contents of the supplied buffer to the supplied stream pointer. Will swap endianness based disposition of mode variable. Useful to be used in conjunction with the advanced interface and a culled packet.

Parameters

packet	the packet to print
len	length of the packet in bytes
swap	1 to swap byte order, 0 to not. Counter-intuitively, it is necessary to swap in order to see the byte order as it is on the wire (this may be a bug).
stream	a stream pointer to print to

3.2.3.85 libnet_diag_dump_pblock()

```
LIBNET_API void libnet_diag_dump_pblock (
             libnet_t * l )
```

[Diagnostic] Prints the contents of every pblock.

Parameters

l	pointer to a libnet context

3.2.3.86 libnet_diag_dump_pblock_type()

```
LIBNET_API char* libnet_diag_dump_pblock_type (
             uint8_t type )
```

[Diagnostic] Returns the canonical name of the pblock type.

Parameters

type	pblock type

Returns

a string representing the pblock type type

Return values

unknown	for an unknown value

3.2.3.87 libnet_get_hwaddr()

```
LIBNET_API struct libnet_ether_addr* libnet_get_hwaddr (
             libnet_t * l )
```

Returns the MAC address for the device libnet was initialized with. If libnet was initialized without a device the function will attempt to find one. If the function fails and returns NULL a call to libnet_geterror() will tell you why.

Parameters

l	pointer to a libnet context

Returns

a pointer to the MAC address or NULL

3.2.3.88 libnet_get_ipaddr4()

```
LIBNET_API uint32_t libnet_get_ipaddr4 (
             libnet_t * l )
```

Returns the IP address for the device libnet was initialized with. If libnet was initialized without a device (in raw socket mode) the function will attempt to find one. If the function fails and returns -1 a call to libnet_geterrror() will tell you why.

Parameters

l	pointer to a libnet context

Returns

a big endian IP address suitable for use in a libnet_build function

Return values

-1	

3.2.3.89 libnet_get_ipaddr6()

```
LIBNET_API struct libnet_in6_addr libnet_get_ipaddr6 (
             libnet_t * l )
```

Returns the IPv6 address for the device libnet was initialized with. If libnet was initialized without a device (in raw socket mode) the function will attempt to find one. If the function fails and returns in6addr_error, a call to libnet_↩ geterrror() will tell you why. This function is not yet implemented for Win32 platforms.

Parameters

l	pointer to a libnet context

Return values

in6addr_error	on failure

3.2.3.90 libnet_get_prand()

```
LIBNET_API uint32_t libnet_get_prand (
             int mod )
```

Generates an unsigned psuedo-random value within the range specified by mod. LIBNET_PR2 0 - 1 LIBNET_PR8 0 - 255 LIBNET_PR16 0 - 32767 LIBNET_PRu16 0 - 65535 LIBNET_PR32 0 - 2147483647 LIBNET_PRu32 0 - 4294967295

Parameters

mod	one the of LIBNET_PR∗ constants

Return values

1	on success
-1	on failure

3.2.3.91 libnet_getdevice()

```
LIBNET_API const char* libnet_getdevice (
             libnet_t * l )
```

Returns the canonical name of the device used for packet injection.

Parameters

l	pointer to a libnet context

Returns

the canonical name of the device used for packet injection. Note it can be NULL without being an error.

3.2.3.92 libnet_geterror()

```
LIBNET_API char* libnet_geterror (
             libnet_t * l )
```

Returns the last error set inside of the referenced libnet context. This function should be called anytime a function fails or an error condition is detected inside of libnet.

Parameters

l	pointer to a libnet context

Returns

an error string or NULL if no error has occured

3.2.3.93 libnet_getfd()

```
LIBNET_API int libnet_getfd (
             libnet_t * l )
```

Returns the FILENO of the file descriptor used for packet injection.

Parameters

l	pointer to a libnet context

Returns

the file number of the file descriptor used for packet injection

3.2.3.94 libnet_getgre_length()

```
LIBNET_API uint32_t libnet_getgre_length (
             uint16_t fv )
```

Parameters

fv	see libnet_build_gre().

Returns

size

See also

libnet_build_gre().

3.2.3.95 libnet_getpacket_size()

```
LIBNET_API uint32_t libnet_getpacket_size (
             libnet_t * l )
```

Returns the sum of the size of all of the pblocks inside of l (this should be the resuling packet size).

Parameters

l	pointer to a libnet context

Returns

the size of the packet in l

3.2.3.96 libnet_getpbuf()

```
LIBNET_API uint8_t* libnet_getpbuf (
             libnet_t * l,
             libnet_ptag_t ptag )
```

Returns the pblock buffer contents for the specified ptag; a subsequent call to libnet_getpbuf_size() should be made to determine the size of the buffer.

Parameters

l	pointer to a libnet context
ptag	the ptag reference number

Returns

a pointer to the pblock buffer or NULL on error

3.2.3.97 libnet_getpbuf_size()

```
LIBNET_API uint32_t libnet_getpbuf_size (
             libnet_t * l,
             libnet_ptag_t ptag )
```

Returns the pblock buffer size for the specified ptag; a previous call to libnet_getpbuf() should be made to pull the actual buffer contents.

Parameters

l	pointer to a libnet context
ptag	the ptag reference number

Returns

the size of the pblock buffer

3.2.3.98 libnet_hex_aton()

```
LIBNET_API uint8_t* libnet_hex_aton (
             const char * s,
             int * len )
```

Takes a colon separated hexidecimal address (from the command line) and returns a bytestring suitable for use in a libnet_build function. Note this function performs an implicit malloc and the return value should be freed after its use.

Parameters

s	the string to be parsed
len	the resulting size of the returned byte string

Returns

a byte string or NULL on failure

3.2.3.99 libnet_in6_is_error()

```
LIBNET_API int libnet_in6_is_error (
             struct libnet_in6_addr addr )
```

Check a libnet_in6_addr structure for identity with in6addr_error.

Parameters

addr	address to check

Return values

1	if addr is in6addr_error
0	if it is not

3.2.3.100 libnet_init()

```
LIBNET_API libnet_t* libnet_init (
             int injection_type,
             const char * device,
             char * err_buf )
```

Creates the libnet environment. It initializes the library and returns a libnet context. If the injection_type is LIBNE↩T_LINK or LIBNET_LINK_ADV, the function initializes the injection primitives for the link-layer interface enabling the application programmer to build packets starting at the data-link layer (which also provides more granular control over the IP layer). If libnet uses the link-layer and the device argument is non-NULL, the function attempts to use the specified network device for packet injection. This is either a canonical string that references the device (such as "eth0" for a 100MB Ethernet card on Linux or "fxp0" for a 100MB Ethernet card on OpenBSD) or the dots and decimals representation of the device's IP address (192.168.0.1). If device is NULL, libnet attempts to find a suitable device to use. If the injection_type is LIBNET_RAW4 or LIBNET_RAW4_ADV, the function initializes the injection primitives for the IPv4 raw socket interface. The final argument, err_buf, should be a buffer of size LIBNET_E↩RRBUF_SIZE and holds an error message if the function fails. This function requires root privileges to execute successfully. Upon success, the function returns a valid libnet context for use in later function calls; upon failure, the function returns NULL.

Parameters

injection_type	packet injection type (LIBNET_LINK, LIBNET_LINK_ADV, LIBNET_RAW4, LIBNET_RAW4_ADV, LIBNET_RAW6, LIBNET_RAW6_ADV)
device	the interface to use (NULL and libnet will choose one)
err_buf	will contain an error message on failure

Returns

libnet context ready for use or NULL on error.

3.2.3.101 libnet_name2addr4()

```
LIBNET_API uint32_t libnet_name2addr4 (
             libnet_t * l,
             const char * host_name,
             uint8_t use_name )
```

Takes a dotted decimal string or a canonical DNS name and returns a network byte ordered IPv4 address. This may incur a DNS lookup if mode is set to LIBNET_RESOLVE and host_name refers to a canonical DNS name. If mode is set to LIBNET_DONT_RESOLVE no DNS lookup will occur. The function can fail if DNS lookup fails or if mode is set to LIBNET_DONT_RESOLVE and host_name refers to a canonical DNS name.

Parameters

l	pointer to a libnet context
host_name	pointer to a string containing a presentation format host name
use_name	LIBNET_RESOLVE or LIBNET_DONT_RESOLVE

Returns

address in network byte order

Return values

-1	(2^32 - 1) on error

3.2.3.102 libnet_name2addr6()

```
LIBNET_API struct libnet_in6_addr libnet_name2addr6 (
             libnet_t * l,
             const char * host_name,
             uint8_t use_name )
```

Takes a dotted decimal string or a canonical DNS name and returns a network byte ordered IPv6 address. This may incur a DNS lookup if mode is set to LIBNET_RESOLVE and host_name refers to a canonical DNS name. If mode is set to LIBNET_DONT_RESOLVE no DNS lookup will occur. The function can fail if DNS lookup fails or if mode is set to LIBNET_DONT_RESOLVE and host_name refers to a canonical DNS name.

Parameters

l	pointer to a libnet context
host_name	pointer to a string containing a presentation format host name
use_name	LIBNET_RESOLVE or LIBNET_DONT_RESOLVE

Returns

network byte ordered IPv6 address structure

3.2.3.103 libnet_plist_chain_dump()

```
LIBNET_API int libnet_plist_chain_dump (
             libnet_plist_t * plist )
```

Runs through the port list and prints the contents of the port list chain list to stdout.

Parameters

plist	previously created portlist

Return values

1	on success
-1	on failure

3.2.3.104 libnet_plist_chain_dump_string()

```
LIBNET_API char* libnet_plist_chain_dump_string (
             libnet_plist_t * plist )
```

Runs through the port list and prints the contents of the port list chain list to string. This function uses strdup and is not re-entrant. It also has a memory leak and should not really be used.

Parameters

plist	previously created portlist

Returns

a printable string containing the port list contents on success or NULL on error

3.2.3.105 libnet_plist_chain_free()

```
LIBNET_API int libnet_plist_chain_free (
             libnet_plist_t * plist )
```

Frees all memory associated with port list chain.

Parameters

plist	previously created portlist

Return values

1	on success
-1	on failure

3.2.3.106 libnet_plist_chain_new()

```
LIBNET_API int libnet_plist_chain_new (
             libnet_t * l,
             libnet_plist_t ** plist,
             char * token_list )
```

Creates a new port list. Port list chains are useful for TCP and UDP-based applications that need to send packets to a range of ports (contiguous or otherwise). The port list chain, which token_list points to, should contain a series of int8_tacters from the following list: "0123456789,-" of the general format "x - y, z", where "xyz" are port numbers between 0 and 65,535. plist points to the front of the port list chain list for use in further libnet_plist_chain() functions. Upon success, the function returns

1. Upon failure, the function returns -1 and libnet_geterror() can tell you why.

Parameters

l	pointer to a libnet context
plist	if successful, will refer to the portlist, if not, NULL
token_list	string containing the port list primitive

Return values

1	on success
-1	on failure

3.2.3.107 libnet_plist_chain_next_pair()

```
LIBNET_API int libnet_plist_chain_next_pair (
             libnet_plist_t * plist,
             uint16_t * bport,
             uint16_t * eport )
```

Returns the next port list chain pair from the port list chain plist. bport and eport contain the starting port number and ending port number, respectively. Upon success, the function returns 1 and fills in the port variables; however, if the list is empty, the function returns 0 and sets both port variables to 0. Upon failure, the function returns -1.

Parameters

plist	previously created portlist
bport	will contain the beginning port number or 0
eport	will contain the ending port number or 0

Return values

1	on success
0	if empty
-1	on failure

3.2.3.108 libnet_seed_prand()

```
LIBNET_API int libnet_seed_prand (
             libnet_t * l )
```

Seeds the psuedo-random number generator.

Parameters

l	pointer to a libnet context

Return values

1	on success
-1	on failure

3.2.3.109 libnet_stats()

```
LIBNET_API void libnet_stats (
```

```
            libnet_t * l,
            struct libnet_stats * ls )
```

Fills in a libnet_stats structure with packet injection statistics (packets written, bytes written, packet sending errors).

Parameters

l	pointer to a libnet context
ls	pointer to a libnet statistics structure

3.2.3.110 libnet_toggle_checksum()

```
LIBNET_API int libnet_toggle_checksum (
            libnet_t * l,
            libnet_ptag_t ptag,
            int mode )
```

If a given protocol header is built with the checksum field set to "0", by default libnet will calculate the header checksum prior to injection. If the header is set to any other value, by default libnet will not calculate the header checksum. To over-ride this behavior, use libnet_toggle_checksum(). Switches auto-checksumming on or off for the specified ptag. If mode is set to LIBNET_ON, libnet will mark the specificed ptag to calculate a checksum for the ptag prior to injection. This assumes that the ptag refers to a protocol that has a checksum field. If mode is set to LIBNET_OFF, libnet will clear the checksum flag and no checksum will be computed prior to injection. This assumes that the programmer will assign a value (zero or otherwise) to the checksum field. Often times this is useful if a precomputed checksum or some other predefined value is going to be used. Note that when libnet is initialized with LIBNET_RAW4, the IPv4 header checksum will always be computed by the kernel prior to injection, regardless of what the programmer sets.

Parameters

l	pointer to a libnet context
ptag	the ptag reference number
mode	LIBNET_ON or LIBNET_OFF

Return values

1	on success
-1	on failure

3.2.3.111 libnet_version()

```
LIBNET_API const char* libnet_version (
            void  )
```

Returns the version of libnet.

Returns

the libnet version

3.2.3.112 libnet_write()

```
LIBNET_API int libnet_write (
             libnet_t * l )
```

Writes a prebuilt packet to the network. The function assumes that l was previously initialized (via a call to libnet_init()) and that a previously constructed packet has been built inside this context (via one or more calls to the libnet_build* family of functions) and is ready to go. Depending on how libnet was initialized, the function will write the packet to the wire either via the raw or link layer interface. The function will also bump up the internal libnet stat counters which are retrievable via libnet_stats().

Parameters

l	pointer to a libnet context

Returns

the number of bytes written

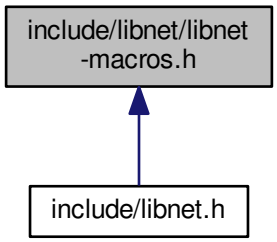
Return values

-1	on error

3.3 include/libnet/libnet-macros.h File Reference

libnet macros and symbolic constants

This graph shows which files directly or indirectly include this file:



Macros

- #define LIBNET_DONT_RESOLVE 0
- #define LIBNET_RESOLVE 1
- #define LIBNET_ON 0
- #define LIBNET_OFF 1
- #define IN6ADDR_ERROR_INIT
- #define LIBNET_PR2 0
- #define LIBNET_MAX_PACKET 0xffff
- #define LIBNET_ERRBUF_SIZE 0x100
- #define LIBNET_MAXOPTION_SIZE 0x28
- #define for_each_context_in_cq(l) for (l = libnet_cq_head(); libnet_cq_last(); l = libnet_cq_next())

3.3.1 Detailed Description

libnet macros and symbolic constants

3.3.2 Macro Definition Documentation

3.3.2.1 for_each_context_in_cq

```
#define for_each_context_in_cq(
              l ) for (l = libnet_cq_head(); libnet_cq_last(); l = libnet_cq_next())
```

Provides an interface to iterate through the context queue of libnet contexts. Before calling this macro, be sure to set the queue using libnet_cq_head().

3.3.2.2 IN6ADDR_ERROR_INIT

```
#define IN6ADDR_ERROR_INIT
```

Value:

```
{ { { 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, \
                                0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, \
                                0xff, 0xff } } }
```

IPv6 error code

3.3.2.3 LIBNET_DONT_RESOLVE

```
#define LIBNET_DONT_RESOLVE 0
```

Used for libnet's name resolution functions, specifies that no DNS lookups should be performed and the IP address should be kept in numeric form.

3.3.2.4 LIBNET_ERRBUF_SIZE

```
#define LIBNET_ERRBUF_SIZE 0x100
```

The libnet error buffer is 256 bytes long.

3.3.2.5 LIBNET_MAX_PACKET

```
#define LIBNET_MAX_PACKET 0xffff
```

The biggest an IP packet can be – 65,535 bytes.

3.3.2.6 LIBNET_MAXOPTION_SIZE

```
#define LIBNET_MAXOPTION_SIZE 0x28
```

IP and TCP options can be up to 40 bytes long.

3.3.2.7 LIBNET_OFF

```
#define LIBNET_OFF 1
```

Used several places, to specify "on" or "one"

3.3.2.8 LIBNET_ON

```
#define LIBNET_ON 0
```

Used several places, to specify "on" or "one"

3.3.2.9 LIBNET_PR2

```
#define LIBNET_PR2 0
```

Used for libnet_get_prand() to specify function disposition

3.3.2.10 LIBNET_RESOLVE

```
#define LIBNET_RESOLVE 1
```

Used for libnet's name resolution functions, specifies that a DNS lookup can be performed if needed to resolve the IP address to a canonical form.

Index

for_each_context_in_cq
    libnet-macros.h, 92

IN6ADDR_ERROR_INIT
    libnet-macros.h, 92
include/libnet.h, 5
include/libnet/libnet-functions.h, 6
include/libnet/libnet-macros.h, 91

LIBNET_DONT_RESOLVE
    libnet-macros.h, 92
LIBNET_ERRBUF_SIZE
    libnet-macros.h, 92
LIBNET_MAX_PACKET
    libnet-macros.h, 93
LIBNET_MAXOPTION_SIZE
    libnet-macros.h, 93
LIBNET_OFF
    libnet-macros.h, 93
LIBNET_ON
    libnet-macros.h, 93
LIBNET_PR2
    libnet-macros.h, 93
LIBNET_RESOLVE
    libnet-macros.h, 93
libnet-functions.h
    libnet_addr2name4, 10
    libnet_addr2name6_r, 11
    libnet_adv_cull_header, 11
    libnet_adv_cull_packet, 11
    libnet_adv_free_packet, 12
    libnet_adv_write_link, 12
    libnet_adv_write_raw_ipv4, 13
    libnet_autobuild_arp, 14
    libnet_autobuild_ethernet, 14
    libnet_autobuild_fddi, 15
    libnet_autobuild_ipv4, 15
    libnet_autobuild_ipv6, 16
    libnet_autobuild_link, 17
    libnet_autobuild_token_ring, 17
    libnet_build_802_1q, 19
    libnet_build_802_1x, 21
    libnet_build_802_2, 21
    libnet_build_802_2snap, 22
    libnet_build_802_3, 23
    libnet_build_arp, 24
    libnet_build_bgp4_header, 25
    libnet_build_bgp4_notification, 25
    libnet_build_bgp4_open, 26
    libnet_build_bgp4_update, 27
libnet_build_bootpv4, 28
libnet_build_cdp, 29
libnet_build_data, 30
libnet_build_dhcpv4, 30
libnet_build_dnsv4, 32
libnet_build_egre, 33
libnet_build_ethernet, 34
libnet_build_fddi, 35
libnet_build_gre, 36
libnet_build_gre_last_sre, 36
libnet_build_gre_sre, 37
libnet_build_hsrp, 38
libnet_build_icmpv4_echo, 38
libnet_build_icmpv4_mask, 39
libnet_build_icmpv4_redirect, 40
libnet_build_icmpv4_timeexceed, 41
libnet_build_icmpv4_timestamp, 41
libnet_build_icmpv4_unreach, 42
libnet_build_icmpv6_echo, 43
libnet_build_icmpv6_ndp_nadv, 44
libnet_build_icmpv6_ndp_nsol, 44
libnet_build_icmpv6_ndp_opt, 45
libnet_build_icmpv6_unreach, 46
libnet_build_igmp, 47
libnet_build_ipsec_ah, 47
libnet_build_ipsec_esp_ftr, 48
libnet_build_ipsec_esp_hdr, 49
libnet_build_ipv4, 50
libnet_build_ipv4_options, 51
libnet_build_ipv6, 51
libnet_build_ipv6_destopts, 52
libnet_build_ipv6_frag, 53
libnet_build_ipv6_hbhopts, 53
libnet_build_ipv6_routing, 54
libnet_build_isl, 55
libnet_build_link, 56
libnet_build_mpls, 57
libnet_build_ntp, 57
libnet_build_ospfv2, 59
libnet_build_ospfv2_dbd, 59
libnet_build_ospfv2_hello, 60
libnet_build_ospfv2_lsa, 61
libnet_build_ospfv2_lsa_as, 62
libnet_build_ospfv2_lsa_net, 62
libnet_build_ospfv2_lsa_rtr, 63
libnet_build_ospfv2_lsa_sum, 64
libnet_build_ospfv2_lsr, 64
libnet_build_ospfv2_lsu, 65
libnet_build_rip, 66

libnet_build_rpc_call, 66
libnet_build_sebek, 67
libnet_build_stp_conf, 68
libnet_build_stp_tcn, 69
libnet_build_tcp, 70
libnet_build_tcp_options, 71
libnet_build_token_ring, 72
libnet_build_udp, 73
libnet_build_vrrp, 73
libnet_clear_packet, 74
libnet_cq_add, 75
libnet_cq_destroy, 75
libnet_cq_end_loop, 75
libnet_cq_find_by_label, 75
libnet_cq_getlabel, 76
libnet_cq_head, 76
libnet_cq_last, 76
libnet_cq_next, 77
libnet_cq_remove, 77
libnet_cq_remove_by_label, 78
libnet_cq_size, 78
libnet_destroy, 78
libnet_diag_dump_context, 79
libnet_diag_dump_hex, 79
libnet_diag_dump_pblock, 79
libnet_diag_dump_pblock_type, 80
libnet_get_hwaddr, 80
libnet_get_ipaddr4, 80
libnet_get_ipaddr6, 81
libnet_get_prand, 81
libnet_getdevice, 82
libnet_geterror, 82
libnet_getfd, 82
libnet_getgre_length, 83
libnet_getpacket_size, 83
libnet_getpbuf, 84
libnet_getpbuf_size, 84
libnet_hex_aton, 84
libnet_in6_is_error, 85
libnet_init, 85
libnet_name2addr4, 86
libnet_name2addr6, 86
libnet_plist_chain_dump, 87
libnet_plist_chain_dump_string, 87
libnet_plist_chain_free, 88
libnet_plist_chain_new, 88
libnet_plist_chain_next_pair, 89
libnet_seed_prand, 89
libnet_stats, 89
libnet_toggle_checksum, 90
libnet_version, 90
libnet_write, 91
libnet-macros.h
for_each_context_in_cq, 92
IN6ADDR_ERROR_INIT, 92
LIBNET_DONT_RESOLVE, 92
LIBNET_ERRBUF_SIZE, 92
LIBNET_MAX_PACKET, 93
LIBNET_MAXOPTION_SIZE, 93
LIBNET_OFF, 93
LIBNET_ON, 93
LIBNET_PR2, 93
LIBNET_RESOLVE, 93
libnet_addr2name4
libnet-functions.h, 10
libnet_addr2name6_r
libnet-functions.h, 11
libnet_adv_cull_header
libnet-functions.h, 11
libnet_adv_cull_packet
libnet-functions.h, 11
libnet_adv_free_packet
libnet-functions.h, 12
libnet_adv_write_link
libnet-functions.h, 12
libnet_adv_write_raw_ipv4
libnet-functions.h, 13
libnet_autobuild_arp
libnet-functions.h, 14
libnet_autobuild_ethernet
libnet-functions.h, 14
libnet_autobuild_fddi
libnet-functions.h, 15
libnet_autobuild_ipv4
libnet-functions.h, 15
libnet_autobuild_ipv6
libnet-functions.h, 16
libnet_autobuild_link
libnet-functions.h, 17
libnet_autobuild_token_ring
libnet-functions.h, 17
libnet_build_802_1q
libnet-functions.h, 19
libnet_build_802_1x
libnet-functions.h, 21
libnet_build_802_2
libnet-functions.h, 21
libnet_build_802_2snap
libnet-functions.h, 22
libnet_build_802_3
libnet-functions.h, 23
libnet_build_arp
libnet-functions.h, 24
libnet_build_bgp4_header
libnet-functions.h, 25
libnet_build_bgp4_notification
libnet-functions.h, 25
libnet_build_bgp4_open
libnet-functions.h, 26
libnet_build_bgp4_update
libnet-functions.h, 27
libnet_build_bootpv4
libnet-functions.h, 28
libnet_build_cdp
libnet-functions.h, 29
libnet_build_data

libnet-functions.h, 30
libnet_build_dhcpv4
libnet-functions.h, 30
libnet_build_dnsv4
libnet-functions.h, 32
libnet_build_egre
libnet-functions.h, 33
libnet_build_ethernet
libnet-functions.h, 34
libnet_build_fddi
libnet-functions.h, 35
libnet_build_gre
libnet-functions.h, 36
libnet_build_gre_last_sre
libnet-functions.h, 36
libnet_build_gre_sre
libnet-functions.h, 37
libnet_build_hsrp
libnet-functions.h, 38
libnet_build_icmpv4_echo
libnet-functions.h, 38
libnet_build_icmpv4_mask
libnet-functions.h, 39
libnet_build_icmpv4_redirect
libnet-functions.h, 40
libnet_build_icmpv4_timeexceed
libnet-functions.h, 41
libnet_build_icmpv4_timestamp
libnet-functions.h, 41
libnet_build_icmpv4_unreach
libnet-functions.h, 42
libnet_build_icmpv6_echo
libnet-functions.h, 43
libnet_build_icmpv6_ndp_nadv
libnet-functions.h, 44
libnet_build_icmpv6_ndp_nsol
libnet-functions.h, 44
libnet_build_icmpv6_ndp_opt
libnet-functions.h, 45
libnet_build_icmpv6_unreach
libnet-functions.h, 46
libnet_build_igmp
libnet-functions.h, 47
libnet_build_ipsec_ah
libnet-functions.h, 47
libnet_build_ipsec_esp_ftr
libnet-functions.h, 48
libnet_build_ipsec_esp_hdr
libnet-functions.h, 49
libnet_build_ipv4
libnet-functions.h, 50
libnet_build_ipv4_options
libnet-functions.h, 51
libnet_build_ipv6
libnet-functions.h, 51
libnet_build_ipv6_destopts
libnet-functions.h, 52
libnet_build_ipv6_frag
libnet-functions.h, 53
libnet_build_ipv6_hbhopts
libnet-functions.h, 53
libnet_build_ipv6_routing
libnet-functions.h, 54
libnet_build_isl
libnet-functions.h, 55
libnet_build_link
libnet-functions.h, 56
libnet_build_mpls
libnet-functions.h, 57
libnet_build_ntp
libnet-functions.h, 57
libnet_build_ospfv2
libnet-functions.h, 59
libnet_build_ospfv2_dbd
libnet-functions.h, 59
libnet_build_ospfv2_hello
libnet-functions.h, 60
libnet_build_ospfv2_lsa
libnet-functions.h, 61
libnet_build_ospfv2_lsa_as
libnet-functions.h, 62
libnet_build_ospfv2_lsa_net
libnet-functions.h, 62
libnet_build_ospfv2_lsa_rtr
libnet-functions.h, 63
libnet_build_ospfv2_lsa_sum
libnet-functions.h, 64
libnet_build_ospfv2_lsr
libnet-functions.h, 64
libnet_build_ospfv2_lsu
libnet-functions.h, 65
libnet_build_rip
libnet-functions.h, 66
libnet_build_rpc_call
libnet-functions.h, 66
libnet_build_sebek
libnet-functions.h, 67
libnet_build_stp_conf
libnet-functions.h, 68
libnet_build_stp_tcn
libnet-functions.h, 69
libnet_build_tcp
libnet-functions.h, 70
libnet_build_tcp_options
libnet-functions.h, 71
libnet_build_token_ring
libnet-functions.h, 72
libnet_build_udp
libnet-functions.h, 73
libnet_build_vrrp
libnet-functions.h, 73
libnet_clear_packet
libnet-functions.h, 74
libnet_cq_add
libnet-functions.h, 75
libnet_cq_destroy

libnet-functions.h, 75
libnet_cq_end_loop
libnet-functions.h, 75
libnet_cq_find_by_label
libnet-functions.h, 75
libnet_cq_getlabel
libnet-functions.h, 76
libnet_cq_head
libnet-functions.h, 76
libnet_cq_last
libnet-functions.h, 76
libnet_cq_next
libnet-functions.h, 77
libnet_cq_remove
libnet-functions.h, 77
libnet_cq_remove_by_label
libnet-functions.h, 78
libnet_cq_size
libnet-functions.h, 78
libnet_destroy
libnet-functions.h, 78
libnet_diag_dump_context
libnet-functions.h, 79
libnet_diag_dump_hex
libnet-functions.h, 79
libnet_diag_dump_pblock
libnet-functions.h, 79
libnet_diag_dump_pblock_type
libnet-functions.h, 80
libnet_get_hwaddr
libnet-functions.h, 80
libnet_get_ipaddr4
libnet-functions.h, 80
libnet_get_ipaddr6
libnet-functions.h, 81
libnet_get_prand
libnet-functions.h, 81
libnet_getdevice
libnet-functions.h, 82
libnet_geterror
libnet-functions.h, 82
libnet_getfd
libnet-functions.h, 82
libnet_getgre_length
libnet-functions.h, 83
libnet_getpacket_size
libnet-functions.h, 83
libnet_getpbuf
libnet-functions.h, 84
libnet_getpbuf_size
libnet-functions.h, 84
libnet_hex_aton
libnet-functions.h, 84
libnet_in6_is_error
libnet-functions.h, 85
libnet_init
libnet-functions.h, 85
libnet_name2addr4
libnet-functions.h, 86
libnet_name2addr6
libnet-functions.h, 86
libnet_plist_chain_dump
libnet-functions.h, 87
libnet_plist_chain_dump_string
libnet-functions.h, 87
libnet_plist_chain_free
libnet-functions.h, 88
libnet_plist_chain_new
libnet-functions.h, 88
libnet_plist_chain_next_pair
libnet-functions.h, 89
libnet_seed_prand
libnet-functions.h, 89
libnet_stats
libnet-functions.h, 89
libnet_toggle_checksum
libnet-functions.h, 90
libnet_version
libnet-functions.h, 90
libnet_write
libnet-functions.h, 91