## NAME

**inet -** Internet protocol family

## SYNOPSIS

**#include <sys/types.h>**
**#include <netinet/in.h>**

## DESCRIPTION

The Internet protocol family is a collection of protocols layered atop the *Internet Protocol* (IP) transport layer, and utilizing the Internet address format. The Internet family provides protocol support for the SOCK_STREAM, SOCK_DGRAM, and SOCK_RAW socket types; the SOCK_RAW interface provides access to the IP protocol.

## ADDRESSING

Internet addresses are four byte quantities, stored in network standard format (on little endian machines, such as the alpha, amd64 and i386 these are word and byte reversed). The include file <*netinet/in.h*> defines this address as a discriminated union.

Sockets bound to the Internet protocol family utilize the following addressing structure,

```
struct sockaddr_in {
        uint8_t         sin_len;
        sa_family_t     sin_family;
        in_port_t sin_port;
        struct in_addr  sin_addr;
        char            sin_zero[8];
};
```

Sockets may be created with the local address INADDR_ANY to affect "wildcard" matching on incoming messages. The address in a connect(2) or sendto(2) call may be given as INADDR_ANY to mean "this host". The distinguished address INADDR_BROADCAST is allowed as a shorthand for the broadcast address on the primary network if the first network configured supports broadcast.

## PROTOCOLS

The Internet protocol family is comprised of the IP network protocol, Internet Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). TCP is used to support the SOCK_STREAM abstraction while UDP is used to support the SOCK_DGRAM abstraction. A raw interface to IP is available by creating an Internet socket of type SOCK_RAW. The ICMP message protocol is accessible from a raw socket.

The **inet** address on an interface consist of the address itself, the netmask, either broadcast address in case of a broadcast interface or peers address in case of point-to-point interface. The following ioctl(2) commands are provided for a datagram socket in the Internet domain:

| | |
|---|---|
| SIOCAIFADDR | Add address to an interface. The command requires *struct in_aliasreq* as argument. |
| SIOCDIFADDR | Delete address from an interface. The command requires *struct ifreq* as argument. |
| SIOCGIFADDR<br>SIOCGIFBRDADDR<br>SIOCGIFDSTADDR | |
| SIOCGIFNETMASK | Return address information from interface. The returned value is in *struct ifreq*. This way of address information retrieval is obsoleted, a preferred way is to use getifaddrs(3) API. |

**MIB (sysctl) Variables**

In addition to the variables supported by the transport protocols in *net.inet* (for which the respective manual pages may be consulted), there are a number of general variables implemented in the *net.inet.ip* branch of the sysctl(3) MIB, which can be also read or modified with sysctl(8). The following general variables are defined:

*accept_sourceroute* Boolean: enable/disable accepting of source-routed IP packets (default false).

*allow_net0* Boolean: allow experimental use of addresses in 0.0.0.0/8 as endpoints, and allow forwarding of packets with these addresses.

*allow_net240* Boolean: allow experimental use of addresses in 240.0.0.0/4 as endpoints, and allow forwarding of packets with these addresses.

*curfrags* Integer: Current number of IPv4 fragments across all reassembly queues in all VNETs (read-only).

*forwarding* Boolean: enable/disable forwarding of IP packets. Defaults to off.

*fragpackets* Integer: Current number of IPv4 fragment reassembly queue entries for the VNET (read-only).

*fragttl* Integer: time to live for IPv4 packet fragments in the per-VNET reassemby queue.

*loopback_prefixlen*

Integer: prefix length of the address space reserved for loopback purposes. The default is 8, meaning that 127.0.0.0/8 is reserved for loopback, and cannot be sent, received, or forwarded on a non-loopback interface. Use of other values is experimental.

*maxfragbucketsize* Integer: maximum number of reassembly queues per bucket. Fragmented packets are hashed to buckets. Each bucket has a list of reassembly queues. The system must compare the incoming packets to the existing reassembly queues in the bucket to find a matching reassembly queue. To preserve system resources, the system limits the number of reassembly queues allowed in each bucket. This limit is recalculated when the number of mbuf clusters is changed or when the value of *maxfragpackets* changes. This is a per-VNET limit.

*maxfragpackets* Integer: maximum number of fragmented packets the host will accept and simultaneously hold in the reassembly queue for a particular VNET. 0 means that the host will not accept any fragmented packets for that VNET. -1 means that the host will not apply this limit for that VNET. This limit is recalculated when the number of mbuf clusters is changed. This is a per-VNET limit.

*maxfrags* Integer: maximum number of fragments the host will accept and simultaneously hold across all reassembly queues in all VNETs. If set to 0, reassembly is disabled. If set to -1, this limit is not applied. This limit is recalculated when the number of mbuf clusters is changed. This is a global limit.

*maxfragsperpacket* Integer: maximum number of fragments the host will accept and hold in the reassembly queue for a packet. 0 means that the host will not accept any fragmented packets for the VNET. This is a per-VNET limit.

*mcast* Variables under the *net.inet.ip.mcast* node are documented in ip(4).

*no_same_prefix* Boolean: Refuse to create same prefixes on different interfaces. This is a per-VNET value.

*portrange* Variables under the *net.inet.ip.portrange* node control port ranges used by transport protocols; see ip(4) for details.

*process_options* Integer: control IP options processing. By setting this variable to 0, all IP options in the incoming packets will be ignored, and the packets will be passed unmodified. By setting to 1, IP options in the incoming packets will be processed accordingly. By setting to 2, an ICMP "prohibited by filter" message will be sent back in

response to incoming packets with IP options. Default is 1. This sysctl(8) variable affects packets destined for a local host as well as packets forwarded to some other host.

*random_id* Boolean: control IP IDs generation behavior. Setting this sysctl(8) to 1 causes the ID field in *non-atomic* IP datagrams (or all IP datagrams, if *rfc6864* is disabled) to be randomized instead of incremented by 1 with each packet generated. This closes a minor information leak which allows remote observers to determine the rate of packet generation on the machine by watching the counter. At the same time, on high-speed links, it can decrease the ID reuse cycle greatly. Default is 0 (sequential IP IDs). IPv6 flow IDs and fragment IDs are always random.

*random_id_collisions*
Integer: count of IP ID collisions (read-only, per-VNET).

*random_id_period* Integer: size of the IP ID array, which is the number of previous packets for which the IDs are recorded. The number must be between 512 and 32768 inclusive. This is a per-VNET value.

*random_id_total* Integer: count of IP IDs created (read-only, per-VNET).

*reass_hashsize* Number of hash slots in the IPv4 reassembly queue (loader tunable).

*redirect* Boolean: enable/disable sending of ICMP redirects in response to IP packets for which a better, and for the sender directly reachable, route and next hop is known. Defaults to on.

*rfc1122_strong_es* Boolean: in non-forwarding mode (forwarding is disabled) partially implement the Strong End System model per RFC1122. If a packet with destination address that is local arrives on a different interface than the interface the address belongs to, the packet would be silently dropped. Enabling this option may break certain setups, e.g. having an alias address(es) on loopback that are expected to be reachable by outside traffic. Enabling some other network features, e.g. carp(4) or destination address rewriting pfil(4) filters may override and bypass this check. Disabled by default.

*rfc6864* Boolean: control IP IDs generation behaviour. True value enables RFC6864 support, which specifies that IP ID field of *atomic* datagrams can be set to any value. The FreeBSD implementation sets it to zero. Enabled by default.

*source_address_validation*
Boolean: perform source address validation for packets destined for the local host. Consider this as following Section 3.2 of RFC3704/BCP84, where we treat local host as our own infrastructure. Forwarded packets are unaffected by this and it should not be considered an anti-spoof feature for a router. Enabled by default.

*sourceroute* Boolean: enable/disable forwarding of source-routed IP packets (default false).

*ttl* Integer: default time-to-live ("TTL") to use for outgoing IP packets.

## SEE ALSO

ioctl(2), socket(2), getifaddrs(3), sysctl(3), icmp(4), intro(4), ip(4), ipfirewall(4), route(4), tcp(4), udp(4), sysctl(8), pfil(9)

"An Introductory 4.3 BSD Interprocess Communication Tutorial", *PS1*, 7.

"An Advanced 4.3 BSD Interprocess Communication Tutorial", *PS1*, 8.

## HISTORY

The **inet** protocol interface appeared in 4.2BSD. The "protocol cloning" code appeared in FreeBSD 2.1.

## CAVEATS

The Internet protocol support is subject to change as the Internet protocols develop. Users should not depend on details of the current implementation, but rather the services exported.