Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'netdevice.7' command

$ man netdevice.7

NETDEVICE(7) Linux Programmer's Manual NETDEVICE(7)

NAME

netdevice - low-level access to Linux network devices

SYNOPSIS

#include <sys/ioctl.h>

#include <net/if.h>

DESCRIPTION

This man page describes the sockets interface which is used to config?

ure network devices.

Linux supports some standard ioctls to configure network devices. They

can be used on any socket's file descriptor regardless of the family or

type. Most of them pass an ifreq structure:

```
struct ifreq {
    char ifr_name[IFNAMSIZ]; /* Interface name */
    union {
        struct sockaddr ifr_addr;
        struct sockaddr ifr_dstaddr;
        struct sockaddr ifr_broadaddr;
        struct sockaddr ifr_netmask;
        struct sockaddr ifr_hwaddr;
        short           ifr_flags;
        int             ifr_ifindex;
        int             ifr_metric;
```

```
        int         ifr_mtu;
        struct ifmap    ifr_map;
        char        ifr_slave[IFNAMSIZ];
        char        ifr_newname[IFNAMSIZ];
        char        *ifr_data;
    };
};
```

Normally, the user specifies which device to affect by setting ifr_name to the name of the interface. All other members of the structure may share memory.

Ioctls

If an ioctl is marked as privileged, then using it requires an effective user ID of 0 or the CAP_NET_ADMIN capability. If this is not the case, EPERM will be returned.

SIOCGIFNAME

Given the ifr_ifindex, return the name of the interface in ifr_name. This is the only ioctl which returns its result in ifr_name.

SIOCGIFINDEX

Retrieve the interface index of the interface into ifr_ifindex.

SIOCGIFFLAGS, SIOCSIFFLAGS

Get or set the active flag word of the device. ifr_flags contains a bit mask of the following values:

Device flags	
IFF_UP	Interface is running.
IFF_BROADCAST	Valid broadcast address set.
IFF_DEBUG	Internal debugging flag.
IFF_LOOPBACK	Interface is a loopback interface.
IFF_POINTOPOINT	Interface is a point-to-point link.
IFF_RUNNING	Resources allocated.
IFF_NOARP	No arp protocol, L2 destination address not set.
IFF_PROMISC	Interface is in promiscuous mode.

IFF_NOTRAILERS Avoid use of trailers.

IFF_ALLMULTI Receive all multicast packets.

IFF_MASTER Master of a load balancing bundle.

IFF_SLAVE Slave of a load balancing bundle.

IFF_MULTICAST Supports multicast

IFF_PORTSEL Is able to select media type via ifmap.

IFF_AUTOMEDIA Auto media selection active.

IFF_DYNAMIC The addresses are lost when the interface goes down.

IFF_LOWER_UP Driver signals L1 up (since Linux 2.6.17)

IFF_DORMANT Driver signals dormant (since Linux 2.6.17)

IFF_ECHO Echo sent packets (since Linux 2.6.25)

Setting the active flag word is a privileged operation, but any process may read it.

SIOCGIFPFLAGS, SIOCSIFPFLAGS

Get or set extended (private) flags for the device. ifr_flags contains a bit mask of the following values:

Private flags

IFF_802_1Q_VLAN Interface is 802.1Q VLAN device.

IFF_EBRIDGE Interface is Ethernet bridging device.

IFF_SLAVE_INACTIVE Interface is inactive bonding slave.

IFF_MASTER_8023AD Interface is 802.3ad bonding master.

IFF_MASTER_ALB Interface is balanced-alb bonding master.

IFF_BONDING Interface is a bonding master or slave.

IFF_SLAVE_NEEDARP Interface needs ARPs for validation.

IFF_ISATAP Interface is RFC4214 ISATAP interface.

Setting the extended (private) interface flags is a privileged opera? tion.

SIOCGIFADDR, SIOCSIFADDR

Get or set the address of the device using ifr_addr. Setting the interface address is a privileged operation. For compati? bility, only AF_INET addresses are accepted or returned.

SIOCGIFDSTADDR, SIOCSIFDSTADDR

Get or set the destination address of a point-to-point device using ifr_dstaddr. For compatibility, only AF_INET addresses are accepted or returned. Setting the destination address is a privileged operation.

SIOCGIFBRDADDR, SIOCSIFBRDADDR

Get or set the broadcast address for a device using ifr_brdaddr. For compatibility, only AF_INET addresses are accepted or returned. Setting the broadcast address is a privileged operation.

SIOCGIFNETMASK, SIOCSIFNETMASK

Get or set the network mask for a device using ifr_netmask. For compatibility, only AF_INET addresses are accepted or returned. Setting the network mask is a privileged operation.

SIOCGIFMETRIC, SIOCSIFMETRIC

Get or set the metric of the device using ifr_metric. This is currently not implemented; it sets ifr_metric to 0 if you attempt to read it and returns EOPNOTSUPP if you attempt to set it.

SIOCGIFMTU, SIOCSIFMTU

Get or set the MTU (Maximum Transfer Unit) of a device using ifr_mtu. Setting the MTU is a privileged operation. Setting the MTU to too small values may cause kernel crashes.

SIOCGIFHWADDR, SIOCSIFHWADDR

Get or set the hardware address of a device using ifr_hwaddr. The hardware address is specified in a struct sockaddr. sa_family contains the ARPHRD_* device type, sa_data the L2 hardware address starting from byte 0. Setting the hardware address is a privileged operation.

SIOCSIFHWBROADCAST

Set the hardware broadcast address of a device from ifr_hwaddr. This is a privileged operation.

SIOCGIFMAP, SIOCSIFMAP

Get or set the interface's hardware parameters using ifr_map.

Setting the parameters is a privileged operation.

```
struct ifmap {
    unsigned long   mem_start;
    unsigned long   mem_end;
    unsigned short  base_addr;
    unsigned char   irq;
    unsigned char   dma;
    unsigned char   port;
};
```

The interpretation of the ifmap structure depends on the device driver and the architecture.

SIOCADDMULTI, SIOCDELMULTI

Add an address to or delete an address from the device's link layer multicast filters using ifr_hwaddr. These are privileged operations. See also packet(7) for an alternative.

SIOCGIFTXQLEN, SIOCSIFTXQLEN

Get or set the transmit queue length of a device using ifr_qlen. Setting the transmit queue length is a privileged operation.

SIOCSIFNAME

Changes the name of the interface specified in ifr_name to ifr_newname. This is a privileged operation. It is allowed only when the interface is not up.

SIOCGIFCONF

Return a list of interface (network layer) addresses. This cur? rently means only addresses of the AF_INET (IPv4) family for compatibility. Unlike the others, this ioctl passes an ifconf structure:

```
struct ifconf {
    int                 ifc_len; /* size of buffer */
    union {
        char           *ifc_buf; /* buffer address */
        struct ifreq   *ifc_req; /* array of structures */
    };
```

};

If ifc_req is NULL, SIOCGIFCONF returns the necessary buffer size in bytes for receiving all available addresses in ifc_len.

Otherwise, ifc_req contains a pointer to an array of ifreq structures to be filled with all currently active L3 interface addresses. ifc_len contains the size of the array in bytes. Within each ifreq structure, ifr_name will receive the interface name, and ifr_addr the address. The actual number of bytes transferred is returned in ifc_len.

If the size specified by ifc_len is insufficient to store all the addresses, the kernel will skip the exceeding ones and re? turn success. There is no reliable way of detecting this condi? tion once it has occurred. It is therefore recommended to ei? ther determine the necessary buffer size beforehand by calling SIOCGIFCONF with ifc_req set to NULL, or to retry the call with a bigger buffer whenever ifc_len upon return differs by less than sizeof(struct ifreq) from its original value.

If an error occurs accessing the ifconf or ifreq structures, EFAULT will be returned.

Most protocols support their own ioctls to configure protocol-specific interface options. See the protocol man pages for a description. For configuring IP addresses, see ip(7).

In addition, some devices support private ioctls. These are not de? scribed here.

NOTES

SIOCGIFCONF and the other ioctls that accept or return only AF_INET socket addresses are IP-specific and perhaps should rather be docu? mented in ip(7).

The names of interfaces with no addresses or that don't have the IFF_RUNNING flag set can be found via /proc/net/dev.

Local IPv6 IP addresses can be found via /proc/net or via rtnetlink(7).

BUGS

glibc 2.1 is missing the ifr_newname macro in <net/if.h>. Add the fol?

lowing to your program as a workaround:

```
#ifndef ifr_newname
#define ifr_newname     ifr_ifru.ifru_slave
#endif
```

SEE ALSO

proc(5), capabilities(7), ip(7), rtnetlink(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.