ip-link(8) — Linux manual page

```
IP-LINK(8)                       Linux                       IP-LINK(8)
```

NAME top

```
       ip-link - network device configuration
```

SYNOPSIS top

```
       ip link  { COMMAND | help }

       ip link add [ link DEVICE ] [ name ] NAME
               [ txqueuelen PACKETS ]
               [ address LLADDR ] [ broadcast LLADDR ]
               [ mtu MTU ] [ index IDX ]
               [ numtxqueues QUEUE_COUNT ] [ numrxqueues QUEUE_COUNT ]
               [ gso_max_size BYTES ] [ gso_max_segs SEGMENTS ]
               type TYPE [ ARGS ]

       ip link delete { DEVICE | group GROUP } type TYPE [ ARGS ]

       ip link set { DEVICE | group GROUP }
               [ { up | down } ]
               [ type ETYPE TYPE_ARGS ]
               [ arp { on | off } ]
               [ dynamic { on | off } ]
               [ multicast { on | off } ]
               [ allmulticast { on | off } ]
               [ promisc { on | off } ]
               [ protodown { on | off } ]
               [ protodown_reason PREASON { on | off } ]
               [ trailers { on | off } ]
               [ txqueuelen PACKETS ]
               [ name NEWNAME ]
               [ address LLADDR ]
               [ broadcast LLADDR ]
               [ mtu MTU ]
               [ netns { PID | NETNSNAME } ]
               [ link-netnsid ID ]
               [ alias NAME ]
               [ vf NUM [ mac LLADDR ]
                        [ VFVLAN-LIST ]
                        [ rate TXRATE ]
                        [ max_tx_rate TXRATE ]
                        [ min_tx_rate TXRATE ]
                        [ spoofchk { on | off } ]
                        [ query_rss { on | off } ]
                        [ state { auto | enable | disable } ]
                        [ trust { on | off } ]
                        [ node_guid eui64 ]
                        [ port_guid eui64 ] ]
               [ { xdp | xdpgeneric | xdpdrv | xdpoffload } { off |
                       object FILE [ section NAME ] [ verbose ] |
                       pinned FILE } ]
               [ master DEVICE ]
               [ nomaster ]
               [ vrf NAME ]
               [ addrgenmode { eui64 | none | stable_secret | random } ]
               [ macaddr [ MACADDR ]
                         [ { flush | add | del } MACADDR ]
                         [ set MACADDR ] ]

                 ip link show [ DEVICE | group GROUP ] [ up ] [ master
                         DEVICE ] [ type ETYPE ] [ vrf NAME ]

                 ip link xstats type TYPE [ ARGS ]

                 ip link afstats [ dev DEVICE ]

                 ip link help [ TYPE ]

                 TYPE := [ bridge | bond | can | dummy | hsr | ifb |
                         ipoib | macvlan | macvtap | vcan | vxcan | veth
                         | vlan | vxlan | ip6tnl | ipip | sit | gre |
                         gretap | erspan | ip6gre | ip6gretap |
                         ip6erspan | vti | nlmon | ipvlan | ipvtap |
                         lowpan | geneve | bareudp | vrf | macsec |
                         netdevsim | rmnet | xfrm ]

                 ETYPE := [ TYPE | bridge_slave | bond_slave ]

                 VFVLAN-LIST := [ VFVLAN-LIST ] VFVLAN

                 VFVLAN := [ vlan VLANID [ qos VLAN-QOS ] [ proto VLAN-
                         PROTO ] ]

         ip link property add [ altname NAME .. ]

         ip link property del [ altname NAME .. ]
```

DESCRIPTION top

ip link add - add virtual link

link *DEVICE*

specifies the physical device to act operate on.

NAME specifies the name of the new virtual device.

TYPE specifies the type of the new device.

Link types:

bridge - Ethernet Bridge device

bond - Bonding device

dummy - Dummy network interface

hsr - High-availability Seamless Redundancy device

ifb - Intermediate Functional Block device

ipoib - IP over Infiniband device

macvlan - Virtual interface base on link layer address (MAC)

macvtap - Virtual interface based on link layer address (MAC) and TAP.

vcan - Virtual Controller Area Network interface

vxcan - Virtual Controller Area Network tunnel interface

veth - Virtual ethernet interface

vlan - 802.1q tagged virtual LAN interface

vxlan - Virtual eXtended LAN

ip6tnl - Virtual tunnel interface IPv4|IPv6 over IPv6

ipip - Virtual tunnel interface IPv4 over IPv4

sit - Virtual tunnel interface IPv6 over IPv4

gre - Virtual tunnel interface GRE over IPv4

gretap - Virtual L2 tunnel interface GRE over IPv4

erspan - Encapsulated Remote SPAN over GRE and IPv4

ip6gre - Virtual tunnel interface GRE over IPv6

ip6gretap - Virtual L2 tunnel interface GRE over IPv6

ip6erspan - Encapsulated Remote SPAN over GRE and IPv6

vti - Virtual tunnel interface

nlmon - Netlink monitoring device

ipvlan - Interface for L3 (IPv6/IPv4) based VLANs

ipvtap - Interface for L3 (IPv6/IPv4) based VLANs and TAP

lowpan - Interface for 6LoWPAN (IPv6) over IEEE 802.15.4 / Bluetooth

geneve - GEneric NEtwork Virtualization Encapsulation

bareudp - Bare UDP L3 encapsulation support

macsec - Interface for IEEE 802.1AE MAC Security (MACsec)

vrf - Interface for L3 VRF domains

netdevsim - Interface for netdev API tests

rmnet - Qualcomm rmnet device

xfrm - Virtual xfrm interface

numtxqueues *QUEUE_COUNT*
specifies the number of transmit queues for new device.

numrxqueues *QUEUE_COUNT*
specifies the number of receive queues for new device.

gso_max_size *BYTES*
specifies the recommended maximum size of a Generic Segment Offload packet the new device should accept.

gso_max_segs *SEGMENTS*
specifies the recommended maximum number of a Generic Segment Offload segments the new device should accept.

index *IDX*
specifies the desired index of the new virtual device. The link creation fails, if the index is busy.

VETH, VXCAN Type Support
For a link of types *VETH/VXCAN* the following additional arguments are supported:

ip link add *DEVICE* **type** { **veth** | **vxcan** } [**peer name** *NAME*]

peer name *NAME* - specifies the virtual pair device name of the *VETH/VXCAN* tunnel.

MACVLAN and MACVTAP Type Support
For a link of type *MACVLAN* or *MACVTAP* the following additional arguments are supported:

ip link add link *DEVICE* **name** *NAME* **type** { **macvlan** | **macvtap** } **mode** { **private** | **vepa** | **bridge** | **passthru** [**nopromisc**] | **source** }

type { **macvlan** | **macvtap** } - specifies the link type to use. **macvlan** creates just a virtual interface, while **macvtap** in addition creates a character device **/dev/tapX** to be used just like a **tuntap** device.

mode private - Do not allow communication between **macvlan** instances on the same physical interface, even if the external switch supports hairpin mode.

mode vepa - Virtual Ethernet Port Aggregator mode. Data from one **macvlan** instance to the other on the same physical interface is transmitted over the physical interface. Either the attached switch needs to support hairpin mode, or there must be a TCP/IP router forwarding the packets in order to allow communication. This is the default mode.

mode bridge - In bridge mode, all endpoints are directly connected to each other, communication is not redirected through the physical interface's peer.

mode passthru [**nopromisc**] - This mode gives more power to a single endpoint, usually in **macvtap** mode. It is not allowed for more than one endpoint on the same physical interface. All traffic will be forwarded to this endpoint, allowing virtio guests to change MAC address or set promiscuous mode in order to bridge the interface or create vlan interfaces on top of it. By default, this mode forces the underlying interface into promiscuous mode. Passing the **nopromisc** flag prevents this, so the promisc flag may be controlled using standard tools.

mode source - allows one to set a list of allowed mac address, which is used to match against source mac address from received frames on underlying interface. This allows creating mac based VLAN associations, instead of standard port or tag based. The feature is useful to deploy 802.1x mac based behavior, where drivers of underlying interfaces doesn't allows that.

High-availability Seamless Redundancy (HSR) Support

For a link of type *HSR* the following additional arguments are supported:

ip link add link *DEVICE* **name** *NAME* **type hsr slave1** *SLAVE1-IF* **slave2** *SLAVE2-IF* [**supervision** *ADDR-BYTE*] [**version** { **0** | **1** } [**proto** { **0** | **1** }]

type hsr - specifies the link type to use, here HSR.

slave1 *SLAVE1-IF* - Specifies the physical device used for the first of the two ring ports.

slave2 *SLAVE2-IF* - Specifies the physical device used for the second of the two ring ports.

supervision *ADDR-BYTE* - The last byte of the multicast address used for HSR supervision frames. Default option is "0", possible values 0-255.

version { **0** | **1** } - Selects the protocol version of the interface. Default option is "0", which corresponds to the 2010 version of the HSR standard. Option "1" activates the 2012 version.

proto { **0** | **1** } - Selects the protocol at the interface. Default option is "0", which corresponds to the HSR standard. Option "1" activates the Parallel Redundancy Protocol (PRP).

BRIDGE Type Support

For a link of type *BRIDGE* the following additional arguments are supported:

ip link add *DEVICE* **type bridge** [**ageing_time** *AGEING_TIME*] [**group_fwd_mask** *MASK*] [**group_address** *ADDRESS*] [

forward_delay *FORWARD_DELAY*] [**hello_time** *HELLO_TIME*] [**max_age** *MAX_AGE*] [**stp_state** *STP_STATE*] [**priority** *PRIORITY*] [**vlan_filtering** *VLAN_FILTERING*] [**vlan_protocol** *VLAN_PROTOCOL*] [**vlan_default_pvid** *VLAN_DEFAULT_PVID*] [**vlan_stats_enabled** *VLAN_STATS_ENABLED*] [**vlan_stats_per_port** *VLAN_STATS_PER_PORT*] [**mcast_snooping** *MULTICAST_SNOOPING*] [**mcast_router** *MULTICAST_ROUTER*] [**mcast_query_use_ifaddr** *MCAST_QUERY_USE_IFADDR*] [**mcast_querier** *MULTICAST_QUERIER*] [**mcast_hash_elasticity** *HASH_ELASTICITY*] [**mcast_hash_max** *HASH_MAX*] [**mcast_last_member_count** *LAST_MEMBER_COUNT*] [**mcast_startup_query_count** *STARTUP_QUERY_COUNT*] [**mcast_last_member_interval** *LAST_MEMBER_INTERVAL*] [**mcast_membership_interval** *MEMBERSHIP_INTERVAL*] [**mcast_querier_interval** *QUERIER_INTERVAL*] [**mcast_query_interval** *QUERY_INTERVAL*] [**mcast_query_response_interval** *QUERY_RESPONSE_INTERVAL*] [**mcast_startup_query_interval** *STARTUP_QUERY_INTERVAL*] [**mcast_stats_enabled** *MCAST_STATS_ENABLED*] [**mcast_igmp_version** *IGMP_VERSION*] [**mcast_mld_version** *MLD_VERSION*] [**nf_call_iptables** *NF_CALL_IPTABLES*] [**nf_call_ip6tables** *NF_CALL_IP6TABLES*] [**nf_call_arptables** *NF_CALL_ARPTABLES*]

ageing_time *AGEING_TIME* - configure the bridge's FDB entries ageing time, ie the number of seconds a MAC address will be kept in the FDB after a packet has been received from that address. after this time has passed, entries are cleaned up.

group_fwd_mask *MASK* - set the group forward mask. This is the bitmask that is applied to decide whether to forward incoming frames destined to link-local addresses, ie addresses of the form 01:80:C2:00:00:0X (defaults to 0, ie the bridge does not forward any link-local frames).

group_address *ADDRESS* - set the MAC address of the multicast group this bridge uses for STP. The address must be a link-local address in standard Ethernet MAC address format, ie an address of the form 01:80:C2:00:00:0X, with X in [0, 4..f].

forward_delay *FORWARD_DELAY* - set the forwarding delay in seconds, ie the time spent in LISTENING state (before moving to LEARNING) and in LEARNING state (before moving to FORWARDING). Only relevant if STP is enabled. Valid values are between 2 and 30.

hello_time *HELLO_TIME* - set the time in seconds between hello packets sent by the bridge, when it is a root bridge or a designated bridges. Only relevant if STP is enabled. Valid values are between 1 and 10.

max_age *MAX_AGE* - set the hello packet timeout, ie the time in seconds until another bridge in the spanning tree is assumed to be dead, after reception of its last hello message. Only relevant if STP is enabled. Valid values are between 6 and

40.

stp_state *STP_STATE* - turn spanning tree protocol on (*STP_STATE* > 0) or off (*STP_STATE* == 0). for this bridge.

priority *PRIORITY* - set this bridge's spanning tree priority, used during STP root bridge election. *PRIORITY* is a 16bit unsigned integer.

nf_call_iptables *NF_CALL_IPTABLES* - enable (*NF_CALL_IPTABLES* > 0) or disable (*NF_CALL_IPTABLES* == 0) iptables hooks on the bridge.

nf_call_ip6tables *NF_CALL_IP6TABLES* - enable (*NF_CALL_IP6TABLES* > 0) or disable (*NF_CALL_IP6TABLES* == 0) ip6tables hooks on the bridge.

nf_call_arptables *NF_CALL_ARPTABLES* - enable (*NF_CALL_ARPTABLES* > 0) or disable (*NF_CALL_ARPTABLES* == 0) arptables hooks on the bridge.

ip link delete - delete virtual link

dev *DEVICE*

specifies the virtual device to act operate on.

group *GROUP*

specifies the group of virtual links to delete. Group 0 is not allowed to be deleted since it is the default group.

type *TYPE*

specifies the type of the device.

ip link set - change device attributes

Warning: If multiple parameter changes are requested, **ip** aborts immediately after any of the changes have failed. This is the only case when **ip** can move the system to an unpredictable state. The solution is to avoid changing several parameters with one **ip link set** call. The modifier **change** is equivalent to **set**.

dev *DEVICE*

DEVICE specifies network device to operate on. When configuring SR-IOV Virtual Function (VF) devices, this keyword should specify the associated Physical Function (PF) device.

group *GROUP*

GROUP has a dual role: If both group and dev are present, then move the device to the specified group. If only a group is specified, then the command operates on all devices in that group.

up and **down**

change the state of the device to **UP** or **DOWN**.

arp on or **arp off**

change the **NOARP** flag on the device.

multicast on or **multicast off**

change the **MULTICAST** flag on the device.

allmulticast on or **allmulticast off**
change the **ALLMULTI** flag on the device. When enabled, instructs network driver to retrieve all multicast packets from the network to the kernel for further processing.

promisc on or **promisc off**
change the **PROMISC** flag on the device. When enabled, activates promiscuous operation of the network device.

trailers on or **trailers off**
change the **NOTRAILERS** flag on the device, **NOT** used by the Linux and exists for BSD compatibility.

protodown on or **protodown off**
change the **PROTODOWN** state on the device. Indicates that a protocol error has been detected on the port. Switch drivers can react to this error by doing a phys down on the switch port.

protodown_reason PREASON on or **off**
set **PROTODOWN** reasons on the device. protodown reason bit names can be enumerated under /etc/iproute2/protodown_reasons.d/. possible reasons bits 0-31

dynamic on or **dynamic off**
change the **DYNAMIC** flag on the device. Indicates that address can change when interface goes down (currently **NOT** used by the Linux).

name *NAME*
change the name of the device. This operation is not recommended if the device is running or has some addresses already configured.

txqueuelen *NUMBER*

txqlen *NUMBER*
change the transmit queue length of the device.

mtu *NUMBER*
change the *MTU* of the device.

address *LLADDRESS*
change the station address of the interface.

broadcast *LLADDRESS*

brd *LLADDRESS*

peer *LLADDRESS*
change the link layer broadcast address or the peer address when the interface is *POINTOPOINT*.

netns *NETNSNAME* | *PID*
move the device to the network namespace associated with name *NETNSNAME* or process *PID*.

Some devices are not allowed to change network namespace: loopback, bridge, wireless. These are network namespace local devices. In such case **ip** tool will return "Invalid argument" error. It is possible to find out if device is local to a single network namespace by checking **netns-local** flag in the output of the **ethtool**:

```
ethtool -k DEVICE
```

To change network namespace for wireless devices the **iw** tool can be used. But it allows to change network namespace only for physical devices and by process *PID*.

alias *NAME*
give the device a symbolic name for easy reference.

group *GROUP*
specify the group the device belongs to. The available groups are listed in file **/etc/iproute2/group**.

vf *NUM* specify a Virtual Function device to be configured. The associated PF device must be specified using the **dev** parameter.

mac *LLADDRESS* - change the station address for the specified VF. The **vf** parameter must be specified.

vlan *VLANID* - change the assigned VLAN for the specified VF. When specified, all traffic sent from the VF will be tagged with the specified VLAN ID. Incoming traffic will be filtered for the specified VLAN ID, and will have all VLAN tags stripped before being passed to the VF. Setting this parameter to 0 disables VLAN tagging and filtering. The **vf** parameter must be specified.

qos *VLAN-QOS* - assign VLAN QOS (priority) bits for the VLAN tag. When specified, all VLAN tags transmitted by the VF will include the specified priority bits in the VLAN tag. If not specified, the value is assumed to be 0. Both the **vf** and **vlan** parameters must be specified. Setting both **vlan** and **qos** as 0 disables VLAN tagging and filtering for the VF.

proto *VLAN-PROTO* - assign VLAN PROTOCOL for the VLAN tag, either 802.1Q or 802.1ad. Setting to 802.1ad, all traffic sent from the VF will be tagged with VLAN S-Tag. Incoming traffic will have VLAN S-Tags stripped before being passed to the VF. Setting to 802.1ad also enables an option to concatenate another VLAN tag, so both S-TAG and C-TAG will be inserted/stripped for outgoing/incoming traffic, respectively. If not specified, the value is assumed to be 802.1Q. Both the **vf** and **vlan** parameters must be specified.

rate *TXRATE* -- change the allowed transmit bandwidth, in Mbps, for the specified VF. Setting this parameter to 0 disables rate limiting. **vf** parameter must be specified. Please use new API **max_tx_rate** option instead.

max_tx_rate *TXRATE* - change the allowed maximum transmit bandwidth, in Mbps, for the specified VF. Setting this parameter to 0 disables rate limiting. **vf** parameter must be specified.

min_tx_rate *TXRATE* - change the allowed minimum transmit bandwidth, in Mbps, for the specified VF. Minimum TXRATE should be always <= Maximum TXRATE. Setting this parameter to 0 disables rate limiting. **vf** parameter must be specified.

spoofchk *on|off* - turn packet spoof checking on or off for the specified VF.

query_rss *on|off* - toggle the ability of querying the RSS configuration of a specific
VF. VF RSS information like RSS hash key may be considered sensitive
on some devices where this information is shared between VF and PF
and thus its querying may be prohibited by default.

state *auto|enable|disable* - set the virtual link state as seen by the specified VF. Setting to auto means a reflection of the PF link state, enable lets the VF to communicate with other VFs on this host even if the PF link state is down, disable causes the HW to drop any packets sent by the VF.

trust *on|off* - trust the specified VF user. This enables that VF user can set a specific feature which may impact security and/or performance. (e.g. VF multicast promiscuous mode)

node_guid *eui64* - configure node GUID for Infiniband VFs.

port_guid *eui64* - configure port GUID for Infiniband VFs.

master *DEVICE*
set master device of the device (enslave device).

nomaster
unset master device of the device (release device).

addrgenmode *eui64|none|stable_secret|random*
set the IPv6 address generation mode

eui64 - use a Modified EUI-64 format interface identifier

none - disable automatic address generation

stable_secret - generate the interface identifier based on a preset
/proc/sys/net/ipv6/conf/{default,DEVICE}/stable_secret

random - like stable_secret, but auto-generate a new random secret if none is set

link-netnsid
set peer netnsid for a cross-netns interface

type *ETYPE TYPE_ARGS*
Change type-specific settings. For a list of supported types and arguments refer to the description of **ip link**

add above. In addition to that, it is possible to manipulate settings to slave devices:

Bridge Slave Support

For a link with master **bridge** the following additional arguments are supported:

ip link set type bridge_slave [**fdb_flush**] [**state** *STATE*] [**priority** *PRIO*] [**cost** *COST*] [**guard** { **on** | **off** }] [**hairpin** { **on** | **off** }] [**fastleave** { **on** | **off** }] [**root_block** { **on** | **off** }] [**learning** { **on** | **off** }] [**flood** { **on** | **off** }] [**proxy_arp** { **on** | **off** }] [**proxy_arp_wifi** { **on** | **off** }] [**mcast_router** *MULTICAST_ROUTER*] [**mcast_fast_leave** { **on** | **off**}] [**mcast_flood** { **on** | **off** }] [**mcast_to_unicast** { **on** | **off** }] [**group_fwd_mask** MASK] [**neigh_suppress** { **on** | **off** }] [**vlan_tunnel** { **on** | **off** }] [**isolated** { **on** | **off** }] [**backup_port** DEVICE] [**nobackup_port**]

fdb_flush - flush bridge slave's fdb dynamic entries.

state *STATE* - Set port state. *STATE* is a number representing the following states: **0** (disabled), **1** (listening), **2** (learning), **3** (forwarding), **4** (blocking).

priority *PRIO* - set port priority (allowed values are between 0 and 63, inclusively).

cost *COST* - set port cost (allowed values are between 1 and 65535, inclusively).

guard { **on** | **off** } - block incoming BPDU packets on this port.

hairpin { **on** | **off** } - enable hairpin mode on this port. This will allow incoming packets on this port to be reflected back.

fastleave { **on** | **off** } - enable multicast fast leave on this port.

root_block { **on** | **off** } - block this port from becoming the bridge's root port.

learning { **on** | **off** } - allow MAC address learning on this port.

flood { **on** | **off** } - open the flood gates on this port, i.e. forward all unicast frames to this port also. Requires **proxy_arp** and **proxy_arp_wifi** to be turned off.

proxy_arp { **on** | **off** } - enable proxy ARP on this port.

proxy_arp_wifi { **on** | **off** } - enable proxy ARP on this port which meets extended requirements by IEEE 802.11 and Hotspot 2.0 specifications.

group_fwd_mask *MASK* - set the group forward mask. This is the bitmask that is applied to decide whether to forward incoming frames destined to link-local addresses, ie addresses of the form 01:80:C2:00:00:0X (defaults to 0, ie the bridge does not forward any link-local frames coming on this port).

neigh_suppress { **on** | **off** } - controls whether neigh discovery (arp and nd) proxy and suppression is enabled on the port. By default this flag is off.

vlan_tunnel { **on** | **off** } - controls whether vlan to tunnel mapping is enabled on the port. By default this flag is off.

backup_port *DEVICE* - if the port loses carrier all traffic will be redirected to the configured backup port

nobackup_port - removes the currently configured backup port

Bonding Slave Support

For a link with master **bond** the following additional arguments are supported:

ip link set type bond_slave [**queue_id** *ID*]

queue_id *ID* - set the slave's queue ID (a 16bit unsigned value).

MACVLAN and MACVTAP Support

Modify list of allowed macaddr for link in source mode.

ip link set type { macvlan | macvap } [**macaddr** *COMMAND MACADDR* ...]

Commands:

add - add MACADDR to allowed list

set - replace allowed list

del - remove MACADDR from allowed list

flush - flush whole allowed list

ip link show - display device attributes

dev *NAME* **(default)**

NAME specifies the network device to show.

group *GROUP*

GROUP specifies what group of devices to show.

up only display running interfaces.

master *DEVICE*

DEVICE specifies the master device which enslaves devices to show.

vrf *NAME*

NAME speficies the VRF which enslaves devices to show.

type *TYPE*

TYPE specifies the type of devices to show.

Note that the type name is not checked against the list of supported types - instead it is sent as-is to the kernel. Later it is used to filter the returned interface list by comparing it with the relevant attribute in case the kernel didn't filter already. Therefore any string is accepted, but may lead to empty output.

ip link xstats - display extended statistics

type *TYPE*

TYPE specifies the type of devices to display extended statistics for.

ip link afstats - display address-family specific statistics

dev *DEVICE*

DEVICE specifies the device to display address-family statistics for.

ip link help - display help

TYPE specifies which help of link type to display.

GROUP

may be a number or a string from the file **/etc/iproute2/group** which can be manually filled.

EXAMPLES top

```
ip link show
    Shows the state of all network interfaces on the system.

ip link show type bridge
    Shows the bridge devices.

ip link show type vlan
    Shows the vlan devices.

ip link show master br0
    Shows devices enslaved by br0

ip link set dev ppp0 mtu 1400
    Change the MTU the ppp0 device.

ip link add link eth0 name eth0.10 type vlan id 10
    Creates a new vlan device eth0.10 on device eth0.

ip link delete dev eth0.10
    Removes vlan device.

ip link help gre
    Display help for the gre link type.

ip link add name tun1 type ipip remote 192.168.1.1 local
192.168.1.2 ttl 225 encap gue encap-sport auto encap-dport 5555
encap-csum encap-remcsum
    Creates an IPIP that is encapsulated with Generic UDP
    Encapsulation, and the outer UDP checksum and remote checksum
    offload are enabled.

ip link set dev eth0 xdp obj prog.o
    Attaches a XDP/BPF program to device eth0, where the program
    is located in prog.o, section "prog" (default section). In
```

```
           case a XDP/BPF program is already attached, throw an error.

       ip -force link set dev eth0 xdp obj prog.o sec foo
           Attaches a XDP/BPF program to device eth0, where the program
           is located in prog.o, section "foo". In case a XDP/BPF
           program is already attached, it will be overridden by the new
           one.

       ip -force link set dev eth0 xdp pinned /sys/fs/bpf/foo
           Attaches a XDP/BPF program to device eth0, where the program
           was previously pinned as an object node into BPF file system
           under name foo.

       ip link set dev eth0 xdp off
           If a XDP/BPF program is attached on device eth0, detach it
           and effectively turn off XDP for device eth0.

       ip link add link wpan0 lowpan0 type lowpan
           Creates a 6LoWPAN interface named lowpan0 on the underlying
           IEEE 802.15.4 device wpan0.

       ip link add dev ip6erspan11 type ip6erspan seq key 102 local
       fc00:100::2 remote fc00:100::1 erspan_ver 2 erspan_dir ingress
       erspan_hwid 17
           Creates a IP6ERSPAN version 2 interface named ip6erspan00.
```

SEE ALSO top

ip(8), ip-netns(8), ethtool(8), iptables(8)

AUTHOR top

Original Manpage by Michail Litvak <mci@owl.openwall.com>

COLOPHON top

This page is part of the *iproute2* (utilities for controlling TCP/IP networking and traffic) project. Information about the project can be found at ⟨http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2⟩. If you have a bug report for this manual page, send it to netdev@vger.kernel.org, shemminger@osdl.org. This page was obtained from the project's upstream Git repository ⟨https://git.kernel.org/pub/scm/network/iproute2/iproute2.git⟩ on 2021-08-27. (At that time, the date of the most recent commit that was found in the repository was 2021-08-18.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is *not* part of the original manual page), send a mail to man-pages@man7.org