NAME

ip-route - routing table management

SYNOPSIS

ip [*ip-OPTIONS*] **route** { *COMMAND* | **help** }

ip route{ list | flush } *SELECTOR*

ip route save *SELECTOR*

ip route restore

ip route get *ADDRESS* [**from** *ADDRESS* **iif** *STRING*] [**oif** *STRING*] [**tos** *TOS*]

ip route{ add | del | c hange | append | replace } *ROUTE*

SELECTOR := [**root** *PREFIX*] [**match** *PREFIX*] [**exact** *PREFIX*] [**table** *TABLE_ID*] [**proto** *RTPROTO*] [**type** *TYPE*] [**scope** *SCOPE*]

ROUTE := *NODE_SPEC* [*INFO_SPEC*]

NODE_SPEC := [*TYPE*] *PREFIX* [**tos** *TOS*] [**table** *TABLE_ID*] [**proto** *RTPROTO*] [**scope** *SCOPE*] [**metric** *METRIC*]

INFO_SPEC := *NH OPTIONS FLAGS* [**nexthop** *NH*] ...

NH := [**via** *ADDRESS*] [**dev** *STRING*] [**weight** *NUMBER*] *NHFLAGS*

OPTIONS := *FLAGS* [**mtu** *NUMBER*] [**advmss** *NUMBER*] [**rtt** *TIME*] [**rttvar** *TIME*] [**reordering** *NUMBER*] [**window** *NUMBER*] [**cwnd** *NUMBER*] [**ssthresh** *REALM*] [**realms** *REALM*] [**rto_min** *TIME*] [**initcwnd** *NUMBER*] [**initrwnd** *NUMBER*] [**quickack** *BOOL*]

TYPE := [**unicast | local | broadcast | multicast | throw | unreachable | prohibit | blackhole | nat**]

TABLE_ID := [**local| main | default | all |** *NUMBER*]

SCOPE := [**host | link | global |** *NUMBER*]

NHFLAGS := [**onlink | pervasive**]

RTPROTO := [**kernel | boot | static |** *NUMBER*]

DESCRIPTION

ip route is used to manipulate entries in the kernel routing tables.

Route types:

unicast - the route entry describes real paths to the destinations covered by the route prefix.

unreachable - these destinations are unreachable. Packets are discarded and the ICMP message *host unreachable* is generated. The local senders get an *EHOSTUNREACH* error.

blackhole - these destinations are unreachable. Packets are discarded silently. The local senders get an *EINVAL* error.

prohibit - these destinations are unreachable. Packets are discarded and the ICMP message *communication administratively prohibited* is generated. The local senders get an *EACCES* error.

local - the destinations are assigned to this host. The packets are looped back and delivered locally.

broadcast - the destinations are broadcast addresses. The packets are sent as link broadcasts.

throw - a special control route used together with policy rules. If such a route is selected, lookup in this table is terminated pretending that no route was found. Without policy routing it is equivalent to the absence of the route in the routing table. The packets are dropped and the ICMP message *net unreachable* is generated. The local senders get an *ENETUNREACH* error.

nat - a special NAT route. Destinations covered by the prefix are considered to be dummy (or external) addresses which require translation to real (or internal) ones before forwarding. The addresses to translate to are selected with the attribute **via**. **Warning:** Route NAT is no longer supported in Linux 2.6.

anycast - *not implemented* the destinations are *anycast* addresses assigned to this host. They are mainly equivalent to **local** with one difference: such addresses are invalid when used as the source address of any packet.

multicast - a special type used for multicast routing. It is not present in normal routing tables.

Route tables: Linux-2.x can pack routes into several routing tables identified by a number in the range from 1 to 2^31 or by name from the file **/etc/iproute2/rt_tables** By default all normal routes are inserted into the **main** table (ID 254) and the kernel only uses this table when calculating routes. Values (0, 253, 254, and 255) are reserved for built-in use.

Actually, one other table always exists, which is invisible but even more important. It is the **local** table (ID 255). This table consists of routes for local and broadcast addresses. The kernel maintains this table automatically and the administrator usually need not modify it or even look at it.

The multiple routing tables enter the game when *policy routing* is used.

ip route add
    add new route

ip route change
    change route

ip route replace
    change or add new one

to *TYPE PREFIX* **(default)**
the destination prefix of the route. If *TYPE* is omitted, **ip** assumes type **unicast**. Other values of *TYPE* are listed above. *PREFIX* is an IP or IPv6 address optionally followed by a slash and the prefix length. If the length of the prefix is missing, **ip** assumes a full-length host route. There is also a special *PREFIX* **default** - which is equivalent to IP **0/0** or to IPv6 **::/0**.

tos *TOS*

dsfield *TOS*
the Type Of Service (TOS) key. This key has no associated mask and the longest match is understood as: First, compare the TOS of the route and of the packet. If they are not equal, then the packet may still match a route with a zero TOS. *TOS* is either an 8 bit hexadecimal number or an identifier from **/etc/iproute2/rt_dsfield**.

metric *NUMBER*

preference *NUMBER*
: the preference value of the route. *NUMBER* is an arbitrary 32bit number.

table *TABLEID*
: the table to add this route to. *TABLEID* may be a number or a string from the file **/etc/iproute2/rt_tables**. If this parameter is omitted,**ip** assumes the **main** table, with the exception of **local**, **broadcast** and **nat** routes, which are put into the **local** table by default.

dev *NAME*
: the output device name.

via *ADDRESS*
: the address of the nexthop router. Actually, the sense of this field depends on the route type. For normal **unicast** routes it is either the true next hop router or, if it is a direct route installed in BSD compatibility mode, it can be a local address of the interface. For NAT routes it is the first address of the block of translated IP destinations.

src *ADDRESS*
: the source address to prefer when sending to the destinations covered by the route prefix.

realm *REALMID*
: the realm to which this route is assigned. *REALMID* may be a number or a string from the file **/etc/iproute2/rt_realms**.

mtu *MTU*

mtu lock *MTU*
: the MTU along the path to the destination. If the modifier **lock** is not used, the MTU may be updated by the kernel due to Path MTU Discovery. If the modifier **lock** is used, no path MTU discovery will be tried, all packets will be sent without the DF bit in IPv4 case or fragmented to MTU for IPv6.

window *NUMBER*
: the maximal window for TCP to advertise to these destinations, measured in bytes. It limits maximal data bursts that our TCP peers are allowed to send to us.

rtt *TIME*
: the initial RTT ('Round Trip Time') estimate. If no suffix is specified the units are raw values passed directly to the routing code to maintain compatibility with previous releases. Otherwise if a suffix of s, sec or secs is used to specify seconds and ms, msec or msecs to specify milliseconds.

rttvar *TIME* **(2.3.15+ only)**
: the initial RTT variance estimate. Values are specified as with **rtt** above.

rto_min *TIME* **(2.6.23+ only)**
: the minimum TCP Retransmission TimeOut to use when communicating with this destination. Values are specified as with **rtt** above.

ssthresh *NUMBER* **(2.3.15+ only)**
: an estimate for the initial slow start threshold.

cwnd *NUMBER* **(2.3.15+ only)**
the clamp for congestion window. It is ignored if the **lock** flag is not used.

initcwnd *NUMBER* **(2.5.70+ only)**
the initial congestion window size for connections to this destination. Actual window size is this value multiplied by the MSS (``Maximal Segment Size'') for same connection. The default is zero, meaning to use the values specified in RFC2414.

initrwnd *NUMBER* **(2.6.33+ only)**
the initial receive window size for connections to this destination. Actual window size is this value multiplied by the MSS of the connection. The default value is zero, meaning to use Slow Start value.

quickack *BOOL* **(3.11+ only)**
Enable or disable quick ack for connections to this destination.

advmss *NUMBER* **(2.3.15+ only)**
the MSS ('Maximal Segment Size') to advertise to these destinations when establishing TCP connections. If it is not given, Linux uses a default value calculated from the first hop device MTU. (If the path to these destination is asymmetric, this guess may be wrong.)

reordering *NUMBER* **(2.3.15+ only)**
Maximal reordering on the path to this destination. If it is not given, Linux uses the value selected with **sysctl** variable **net/ipv4/tcp_reordering**.

nexthop *NEXTHOP*
the nexthop of a multipath route. *NEXTHOP* is a complex value with its own syntax similar to the top level argument lists:

via *ADDRESS* - is the nexthop router.

dev *NAME* - is the output device.

weight *NUMBER* - is a weight for this element of a multipath route reflecting its relative bandwidth or quality.

scope *SCOPE_VAL*
the scope of the destinations covered by the route prefix. *SCOPE_VAL* may be a number or a string from the file **/etc/iproute2/rt_scopes**. If this parameter is omitted, **ip** assumes scope **global** for all gatewayed **unicast** routes, scope **link** for direct **unicast** and **broadcast** routes and scope **host** for **local** routes.

protocol *RTPROTO*
the routing protocol identifier of this route. *RTPROTO* may be a number or a string from the file **/etc/iproute2/rt_protos**. If the routing protocol ID is not given, **ip assumes protocol boot** (i.e. it assumes the route was added by someone who doesn't understand what they are doing). Several protocol values have a fixed interpretation. Namely:

redirect - the route was installed due to an ICMP redirect.

kernel - the route was installed by the kernel during autoconfiguration.

boot - the route was installed during the bootup sequence. If a routing daemon starts, it will purge all of them.

static - the route was installed by the administrator to override dynamic routing. Routing daemon will respect them and, probably, even advertise them to its peers.

ra - the route was installed by Router Discovery protocol.

The rest of the values are not reserved and the administrator is free to assign (or not to assign) protocol tags.

onlink pretend that the nexthop is directly attached to this link, even if it does not match any interface prefix.

ip route delete
delete route
ip route del has the same arguments as **ip route add**, but their semantics are a bit different.

Key values (**to**, **tos**, **preference** and **table**) select the route to delete. If optional attributes are present, **ip** verifies that they coincide with the attributes of the route to delete. If no route with the given key and attributes was found, **ip route del** fails.

ip route show
list routes
the command displays the contents of the routing tables or the route(s) selected by some criteria.

to *SELECTOR* **(default)**
only select routes from the given range of destinations. *SELECTOR* consists of an optional modifier (**root**, **match** or **exact**) and a prefix. **root** *PREFIX* selects routes with prefixes not shorter than *PREFIX.* F.e. **root** *0/0* selects the entire routing table. **match** *PREFIX* selects routes with prefixes not longer than *PREFIX.* F.e. **match** *10.0/16* selects *10.0/16*, *10/8* and *0/0*, but it does not select *10.1/16* and *10.0.0/24*. And**exact** *PREFIX* (or just *PREFIX*) selects routes with this exact prefix. If neither of these options are present, **ip** assumes **root** *0/0* i.e. it lists the entire table.

tos *TOS*

dsfield *TOS*
only select routes with the given TOS.

table *TABLEID*
show the routes from this table(s). The default setting is to show table **main**. *TABLEID* may either be the ID of a real table or one of the special values:

all - list all of the tables.

cache - dump the routing cache.

cloned

cached
list cloned routes i.e. routes which were dynamically forked from other routes because some route attribute (f.e. MTU) was updated. Actually, it is equivalent to **table cache**.

from *SELECTOR*
the same syntax as for **to**, but it binds the source address range rather than destinations. Note that the**from** option only w orks with cloned routes.

protocol *RTPROTO*
only list routes of this protocol.

scope *SCOPE_VAL*
only list routes with this scope.

type *TYPE*
only list routes of this type.

dev *NAME*
only list routes going via this device.

via *PREFIX*
only list routes going via the nexthop routers selected by *PREFIX*.

src *PREFIX*
only list routes with preferred source addresses selected by *PREFIX*.

realm *REALMID*

realms *FROMREALM/TOREALM*
only list routes with these realms.

ip route flush
flush routing tables
this command flushes routes selected by some criteria.

The arguments have the same syntax and semantics as the arguments of **ip route show**, but routing tables are not listed but purged. The only difference is the default action: **show** dumps all the IP main routing table but **flush** prints the helper page.

With the **-statistics** option, the command becomes verbose. It prints out the number of deleted routes and the number of rounds made to flush the routing table. If the option is given twice, **ip route flush** also dumps all the deleted routes in the format described in the previous subsection.

ip route get
get a single route
this command gets a single route to a destination and prints its contents exactly as the kernel sees it.

to *ADDRESS* **(default)**
the destination address.

from *ADDRESS*
the source address.

tos *TOS*

dsfield *TOS*
the Type Of Service.

iif *NAME*
the device from which this packet is expected to arrive.

oif *NAME*
force the output device on which this packet will be routed.

connected

if no source address (option **from**) was given, relookup the route with the source set to the preferred address received from the first lookup. If policy routing is used, it may be a different route.

Note that this operation is not equivalent to **ip route show**. **show** shows existing routes. **get** resolves them and creates new clones if necessary. Essentially, **get** is equivalent to sending a packet along this path. If the **iif** argument is not given, the kernel creates a route to output packets towards the requested destination. This is equivalent to pinging the destination with a subsequent **ip route ls cache**, however, no packets are actually sent. With the **iif** argument, the kernel pretends that a packet arrived from this interface and searches for a path to forward the packet.

ip route save

save routing table information to stdout
This command behaves like **ip route show** except that the output is raw data suitable for passing to **ip route restore**.

ip route restore

restore routing table information from stdin
This command expects to read a data stream as returned from **ip route save**. It will attempt to restore the routing table information exactly as it was at the time of the save, so any translation of information in the stream (such as device indexes) must be done first. Any existing routes are left unchanged. Any routes specified in the data stream that already exist in the table will be ignored.

EXAMPLES

ip ro

Show all route entries in the kernel.

ip route add default via 192.168.1.1 dev eth0

Adds a default route (for all addresses) via the local gateway 192.168.1.1 that can be reached on device eth0.

SEE ALSO

ip(8)

AUTHOR

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