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

$ man rtnetlink.7

```
RTNETLINK(7)            Linux Programmer's Manual           RTNETLINK(7)

NAME

       rtnetlink - Linux IPv4 routing socket

SYNOPSIS

       #include <asm/types.h>

       #include <linux/netlink.h>

       #include <linux/rtnetlink.h>

       #include <sys/socket.h>

       rtnetlink_socket = socket(AF_NETLINK, int socket_type, NETLINK_ROUTE);

DESCRIPTION

       Rtnetlink  allows  the  kernel's routing tables to be read and altered.

       It is used within the kernel to communicate between various subsystems,

       though  this  usage  is not documented here, and for communication with

       user-space programs.  Network routes, IP  addresses,  link  parameters,

       neighbor setups, queueing disciplines, traffic classes and packet clas?

       sifiers may all be controlled through  NETLINK_ROUTE  sockets.   It  is

       based on netlink messages; see netlink(7) for more information.

   Routing attributes

       Some  rtnetlink  messages  have  optional  attributes after the initial

       header:

           struct rtattr {

               unsigned short rta_len;    /* Length of option */

               unsigned short rta_type;   /* Type of option */
```

```
          /* Data follows */
      };
```

These attributes should be manipulated using only the RTA_* macros or libnetlink, see rtnetlink(3).

Messages

Rtnetlink consists of these message types (in addition to standard netlink messages):

RTM_NEWLINK, RTM_DELLINK, RTM_GETLINK

Create, remove, or get information about a specific network in?terface. These messages contain an ifinfomsg structure followed by a series of rtattr structures.

```
struct ifinfomsg {
    unsigned char  ifi_family; /* AF_UNSPEC */
    unsigned short ifi_type;   /* Device type */
    int            ifi_index;  /* Interface index */
    unsigned int   ifi_flags;  /* Device flags  */
    unsigned int   ifi_change; /* change mask */
};
```

ifi_flags contains the device flags, see netdevice(7); ifi_index is the unique interface index (since Linux 3.7, it is possible to feed a nonzero value with the RTM_NEWLINK message, thus cre?ating a link with the given ifindex); ifi_change is reserved for future use and should be always set to 0xFFFFFFFF.

Routing attributes

rta_type	Value type	Description
IFLA_UNSPEC	-	unspecified
IFLA_ADDRESS	hardware address	interface L2 address
IFLA_BROADCAST	hardware address	L2 broadcast address
IFLA_IFNAME	asciiz string	Device name
IFLA_MTU	unsigned int	MTU of the device
IFLA_LINK	int	Link type
IFLA_QDISC	asciiz string	Queueing discipline

IFLA_STATS	see below	Interface Statistics

The value type for IFLA_STATS is struct rtnl_link_stats (struct net_device_stats in Linux 2.4 and earlier).

RTM_NEWADDR, RTM_DELADDR, RTM_GETADDR

Add, remove, or receive information about an IP address associ?ated with an interface. In Linux 2.2, an interface can carry multiple IP addresses, this replaces the alias device concept in 2.0. In Linux 2.2, these messages support IPv4 and IPv6 ad?dresses. They contain an ifaddrmsg structure, optionally fol?lowed by rtattr routing attributes.

```
struct ifaddrmsg {
    unsigned char ifa_family;    /* Address type */
    unsigned char ifa_prefixlen; /* Prefixlength of address */
    unsigned char ifa_flags;     /* Address flags */
    unsigned char ifa_scope;     /* Address scope */
    unsigned int  ifa_index;     /* Interface index */
};
```

ifa_family is the address family type (currently AF_INET or AF_INET6), ifa_prefixlen is the length of the address mask of the address if defined for the family (like for IPv4), ifa_scope is the address scope, ifa_index is the interface index of the interface the address is associated with. ifa_flags is a flag word of IFA_F_SECONDARY for secondary address (old alias inter?face), IFA_F_PERMANENT for a permanent address set by the user and other undocumented flags.

Attributes

rta_type	Value type	Description
??		
IFA_UNSPEC	-	unspecified
IFA_ADDRESS	raw protocol address	interface address
IFA_LOCAL	raw protocol address	local address
IFA_LABEL	asciiz string	name of the interface
IFA_BROADCAST	raw protocol address	broadcast address

IFA_ANYCAST	raw protocol address	anycast address
IFA_CACHEINFO	struct ifa_cacheinfo	Address information

RTM_NEWROUTE, RTM_DELROUTE, RTM_GETROUTE

Create, remove, or receive information about a network route.

These messages contain an rtmsg structure with an optional se?

quence of rtattr structures following. For RTM_GETROUTE, set?

ting rtm_dst_len and rtm_src_len to 0 means you get all entries

for the specified routing table. For the other fields, except

rtm_table and rtm_protocol, 0 is the wildcard.

```
struct rtmsg {
    unsigned char rtm_family;   /* Address family of route */
    unsigned char rtm_dst_len;  /* Length of destination */
    unsigned char rtm_src_len;  /* Length of source */
    unsigned char rtm_tos;      /* TOS filter */
    unsigned char rtm_table;    /* Routing table ID;
                                   see RTA_TABLE below */
    unsigned char rtm_protocol; /* Routing protocol; see below */
    unsigned char rtm_scope;    /* See below */
    unsigned char rtm_type;     /* See below */
    unsigned int  rtm_flags;
};
```

rtm_type	Route type
RTN_UNSPEC	unknown route
RTN_UNICAST	a gateway or direct route
RTN_LOCAL	a local interface route
RTN_BROADCAST	a local broadcast route (sent as a broadcast)
RTN_ANYCAST	a local broadcast route (sent as a uni? cast)
RTN_MULTICAST	a multicast route
RTN_BLACKHOLE	a packet dropping route
RTN_UNREACHABLE	an unreachable destination

RTN_PROHIBIT	a packet rejection route
RTN_THROW	continue routing lookup in another table
RTN_NAT	a network address translation rule
RTN_XRESOLVE	refer to an external resolver (not im? plemented)

rtm_protocol Route origin

???

RTPROT_UNSPEC	unknown
RTPROT_REDIRECT	by an ICMP redirect (cur? rently unused)
RTPROT_KERNEL	by the kernel
RTPROT_BOOT	during boot
RTPROT_STATIC	by the administrator

Values larger than RTPROT_STATIC are not interpreted by the ker?
nel, they are just for user information. They may be used to
tag the source of a routing information or to distinguish be?
tween multiple routing daemons. See <linux/rtnetlink.h> for the
routing daemon identifiers which are already assigned.

rtm_scope is the distance to the destination:

RT_SCOPE_UNIVERSE	global route
RT_SCOPE_SITE	interior route in the lo? cal autonomous system
RT_SCOPE_LINK	route on this link
RT_SCOPE_HOST	route on the local host
RT_SCOPE_NOWHERE	destination doesn't exist

The values between RT_SCOPE_UNIVERSE and RT_SCOPE_SITE are
available to the user.

The rtm_flags have the following meanings:

RTM_F_NOTIFY	if the route changes, notify the user via rtnetlink
RTM_F_CLONED	route is cloned from another route
RTM_F_EQUALIZE	a multipath equalizer (not yet implemented)

rtm_table specifies the routing table

RT_TABLE_UNSPEC	an unspecified routing table
RT_TABLE_DEFAULT	the default table
RT_TABLE_MAIN	the main table
RT_TABLE_LOCAL	the local table

The user may assign arbitrary values between RT_TABLE_UNSPEC and RT_TABLE_DEFAULT.

Attributes

rta_type	Value type	Description
??		
RTA_UNSPEC	-	ignored
RTA_DST	protocol address	Route destination address
RTA_SRC	protocol address	Route source address
RTA_IIF	int	Input interface index
RTA_OIF	int	Output interface index
RTA_GATEWAY	protocol address	The gateway of the route
RTA_PRIORITY	int	Priority of route
RTA_PREFSRC	protocol address	Preferred source address
RTA_METRICS	int	Route metric
RTA_MULTIPATH		Multipath nexthop data br (see below).
RTA_PROTOINFO		No longer used
RTA_FLOW	int	Route realm
RTA_CACHEINFO	struct rta_cacheinfo	(see linux/rtnetlink.h)
RTA_SESSION		No longer used
RTA_MP_ALGO		No longer used
RTA_TABLE	int	Routing table ID; if set, rtm_table is ignored
RTA_MARK	int	
RTA_MFC_STATS	struct rta_mfc_stats	(see linux/rtnetlink.h)
RTA_VIA	struct rtvia	Gateway in different AF (see below)
RTA_NEWDST	protocol address	Change packet destination address

```
RTA_PREF        char                RFC4191 IPv6 router pref?
                                erence (see below)
RTA_ENCAP_TYPE  short               Encapsulation type for
                                lwtunnels (see below)
RTA_ENCAP                           Defined by RTA_ENCAP_TYPE
RTA_EXPIRES     int                 Expire  time   for   IPv6
                                routes (in seconds)
```

RTA_MULTIPATH contains several packed instances of struct rtnex?

thop together with nested RTAs (RTA_GATEWAY):

```
    struct rtnexthop {
        unsigned short rtnh_len;     /* Length of struct + length
                                        of RTAs */
        unsigned char  rtnh_flags;   /* Flags (see
                                        linux/rtnetlink.h) */
        unsigned char  rtnh_hops;    /* Nexthop priority */
        int            rtnh_ifindex; /* Interface index for this
                                        nexthop */
    }
```

There exist a bunch of RTNH_* macros similar to RTA_* and NL?

HDR_* macros useful to handle these structures.

```
    struct rtvia {
        unsigned short rtvia_family;
        unsigned char  rtvia_addr[0];
    };
```

rtvia_addr is the address, rtvia_family is its family type.

RTA_PREF   may   contain   values   ICMPV6_ROUTER_PREF_LOW,

ICMPV6_ROUTER_PREF_MEDIUM, and ICMPV6_ROUTER_PREF_HIGH defined

incw <linux/icmpv6.h>.

RTA_ENCAP_TYPE may contain values LWTUNNEL_ENCAP_MPLS, LWTUN?

NEL_ENCAP_IP, LWTUNNEL_ENCAP_ILA, or LWTUNNEL_ENCAP_IP6 defined

in <linux/lwtunnel.h>.

Fill these values in!

RTM_NEWNEIGH, RTM_DELNEIGH, RTM_GETNEIGH

Add, remove, or receive information about a neighbor table entry

(e.g., an ARP entry). The message contains an ndmsg structure.

```
struct ndmsg {
    unsigned char ndm_family;
    int           ndm_ifindex;  /* Interface index */
    __u16         ndm_state;    /* State */
    __u8          ndm_flags;    /* Flags */
    __u8          ndm_type;
};
struct nda_cacheinfo {
    __u32         ndm_confirmed;
    __u32         ndm_used;
    __u32         ndm_updated;
    __u32         ndm_refcnt;
};
```

ndm_state is a bit mask of the following states:

NUD_INCOMPLETE	a currently resolving cache entry
NUD_REACHABLE	a confirmed working cache entry
NUD_STALE	an expired cache entry
NUD_DELAY	an entry waiting for a timer
NUD_PROBE	a cache entry that is currently reprobed
NUD_FAILED	an invalid cache entry
NUD_NOARP	a device with no destination cache
NUD_PERMANENT	a static entry

Valid ndm_flags are:

NTF_PROXY	a proxy arp entry
NTF_ROUTER	an IPv6 router

The rtattr struct has the following meanings for the rta_type

field:

NDA_UNSPEC	unknown type
NDA_DST	a neighbor cache n/w layer destination address
NDA_LLADDR	a neighbor cache link layer address
NDA_CACHEINFO	cache statistics

If the rta_type field is NDA_CACHEINFO, then a struct nda_cache?

info header follows.

RTM_NEWRULE, RTM_DELRULE, RTM_GETRULE

Add, delete, or retrieve a routing rule. Carries a struct rtmsg

RTM_NEWQDISC, RTM_DELQDISC, RTM_GETQDISC

Add, remove, or get a queueing discipline. The message contains

a struct tcmsg and may be followed by a series of attributes.

```
struct tcmsg {
    unsigned char    tcm_family;
    int              tcm_ifindex;   /* interface index */
    __u32            tcm_handle;    /* Qdisc handle */
    __u32            tcm_parent;    /* Parent qdisc */
    __u32            tcm_info;
};
```

Attributes

rta_type	Value type	Description
??		
TCA_UNSPEC	-	unspecified
TCA_KIND	asciiz string	Name of queueing discipline
TCA_OPTIONS	byte sequence	Qdisc-specific options follow
TCA_STATS	struct tc_stats	Qdisc statistics
TCA_XSTATS	qdisc-specific	Module-specific statistics
TCA_RATE	struct tc_estimator	Rate limit

In addition, various other qdisc-module-specific attributes are

allowed. For more information see the appropriate include

files.

RTM_NEWTCLASS, RTM_DELTCLASS, RTM_GETTCLASS

Add, remove, or get a traffic class. These messages contain a

struct tcmsg as described above.

RTM_NEWTFILTER, RTM_DELTFILTER, RTM_GETTFILTER

Add, remove, or receive information about a traffic filter.

These messages contain a struct tcmsg as described above.

VERSIONS

rtnetlink is a new feature of Linux 2.2.

BUGS

This manual page is incomplete.

SEE ALSO

cmsg(3), rtnetlink(3), ip(7), netlink(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/.