

NAME

tunnel_key - Tunnel metadata manipulation

SYNOPSIS

tc ... action tunnel_key { unset | SET }

SET := **set src_ip ADDRESS dst_ip ADDRESS id KEY_ID dst_port UDP_PORT tos TOS ttl TTL [csum | nocsum]**

DESCRIPTION

The **tunnel_key** action combined with a shared IP tunnel device, allows to perform IP tunnel en- or decapsulation on a packet, reflected by the operation modes *UNSET* and *SET*. The *UNSET* mode is optional - even without using it, the metadata information will be released automatically when packet processing will be finished. *UNSET* function could be used in cases when traffic is forwarded between two tunnels, where the metadata from the first tunnel will be used for encapsulation done by the second tunnel. *SET* mode requires the source and destination ip *ADDRESS* and the tunnel key id *KEY_ID* which will be used by the ip tunnel shared device to create the tunnel header. The **tunnel_key** action is useful only in combination with a **mirred redirect** action to a shared IP tunnel device which will use the metadata (for *SET*) and unset the metadata created by it (for *UNSET*).

OPTIONS

unset Unset the tunnel metadata created by the IP tunnel device. This function is not mandatory and might be used only in some specific use cases (as explained above).

set Set tunnel metadata to be used by the IP tunnel device. Requires **src_ip** and **dst_ip** options. **id** , **dst_port** , **geneve_opts** , **vlan_opts** and **erspan_opts** are optional.

id Tunnel ID (for example VNI in VXLAN tunnel)

src_ip Outer header source IP address (IPv4 or IPv6)

dst_ip Outer header destination IP address (IPv4 or IPv6)

dst_port
Outer header destination UDP port

geneve_opts
Geneve variable length options. **geneve_opts** is specified in the form CLASS:TYPE:DATA, where CLASS is represented as a 16bit hexadecimal value, TYPE as an 8bit hexadecimal value and DATA as a variable length hexadecimal value. Additionally multiple options may be listed using a comma delimiter.

vlan_opts
Vlan metadata options. **vlan_opts** is specified in the form GBP, as a 32bit number. Multiple options is not supported.

erspan_opts
Erspar metadata options. **erspan_opts** is specified in the form VERSION:INDEX:DIR:HWID, where VERSION is represented as a 8bit number, INDEX as an 32bit number, DIR and HWID as a 8bit number. Multiple options is not supported. Note INDEX is used when VERSION is 1, and DIR and HWID are used when VERSION is 2.

tos Outer header TOS

ttl Outer header TTL

[no]csum

Controls outer UDP checksum. When set to **csum** (which is default), the outer UDP checksum is calculated and included in the packets. When set to **nocsum**, outer UDP checksum is zero. Note that when using zero UDP checksums with IPv6, the other tunnel endpoint must be configured to accept such packets. In Linux, this would be the

udp6zerocsumrx option for the VXLAN tunnel interface.

If using **nocsum** with IPv6, be sure you know what you are doing. Zero UDP checksums provide weaker protection against corrupted packets. See RFC6935 for details.

EXAMPLES

The following example encapsulates incoming ICMP packets on eth0 into a vxlan tunnel, by setting metadata to VNI 11, source IP 11.11.0.1 and destination IP 11.11.0.2, and by redirecting the packet with the metadata to device vxlan0, which will do the actual encapsulation using the metadata:

```
#tc qdisc add dev eth0 handle ffff: ingress
#tc filter add dev eth0 protocol ip parent ffff: \
    flower \
        ip_proto icmp \
        action tunnel_key set \
            src_ip 11.11.0.1 \
            dst_ip 11.11.0.2 \
            id 11 \
        action mirred egress redirect dev vxlan0
```

Here is an example of the **unset** function: Incoming VXLAN traffic with outer IP's and VNI 11 is decapsulated by vxlan0 and metadata is unset before redirecting to tunl1 device:

```
#tc qdisc add dev eth0 handle ffff: ingress
#tc filter add dev vxlan0 protocol ip parent ffff:    flower \
    enc_src_ip 11.11.0.2 enc_dst_ip 11.11.0.1 enc_key_id 11    action tu
```

SEE ALSO

tc(8)