

NAME

bridge – show / manipulate bridge addresses and devices

SYNOPSIS

bridge [*OPTIONS*] *OBJECT* { *COMMAND* | **help** }

OBJECT := { **link** | **fdb** | **mdb** | **vlan** | **vni** | **monitor** }

OPTIONS := { **-V**[ersion] | **-s**[tatistics] | **-n**[etns] name | **-b**[atch] filename | **-com**[pressvlans] | **-c**[olor] | **-p**[retty] | **-j**[son] | **-o**[neline] }

bridge link set dev *DEV* [**cost** *COST*] [**priority** *PRIO*] [**state** *STATE*] [**guard** { **on** | **off** }] [**hairpin** { **on** | **off** }] [**fastleave** { **on** | **off** }] [**root_block** { **on** | **off** }] [**learning** { **on** | **off** }] [**learning_sync** { **on** | **off** }] [**flood** { **on** | **off** }] [**hwmode** { **vepa** | **veb** }] [**bcast_flood** { **on** | **off** }] [**mcast_flood** { **on** | **off** }] [**mcast_max_groups** *MAX_GROUPS*] [**mcast_router** *MULTICAST_ROUTER*] [**mcast_to_unicast** { **on** | **off** }] [**neigh_suppress** { **on** | **off** }] [**neigh_vlan_suppress** { **on** | **off** }] [**vlan_tunnel** { **on** | **off** }] [**isolated** { **on** | **off** }] [**locked** { **on** | **off** }] [**mab** { **on** | **off** }] [**backup_port** *DEVICE*] [**nobackup_port**] [**backup_nhid** *NHID*] [**self**] [**master**]

bridge link [**show**] [**dev** *DEV*] [**master** *DEVICE*]

bridge fdb { **add** | **append** | **del** | **replace** } *LLADDR* **dev** *DEV* { **local** | **static** | **dynamic** } [**self**] [**master**] [**router**] [**use**] [**extern_learn**] [**sticky**] [**src_vni** *VNI*] { [**dst** *IPADDR*] [**vni** *VNI*] } [**port** *PORT*] [**via** *DEVICE*] [**nhid** *NHID*] [**activity_notify**] [**inactive**] [**norefresh**]

bridge fdb [[**show**] [**br** *BRDEV*] [**brport** *DEV*] [**vlan** *VID*] [**state** *STATE*] [**dynamic**]]

bridge fdb get [**to**] *LLADDR* [**br** *BRDEV*] { **brport** | **dev** } *DEV* [**vlan** *VID*] [**vni** *VNI*] [**self**] [**master**] [**dynamic**]

bridge fdb flush dev *DEV* [**brport** *DEV*] [**vlan** *VID*] [**src_vni** *VNI*] [**nhid** *NHID*] [**vni** *VNI*] [**port** *PORT*] [**dst** *IPADDR*] [**self**] [**master**] [**[no]permanent**] [**[no]static**] [**[no]dynamic**] [**[no]added_by_user**] [**[no]extern_learn**] [**[no]sticky**] [**[no]offloaded**] [**[no]router**]

bridge mdb { **add** | **del** | **replace** } **dev** *DEV* **port** *PORT* **grp** *GROUP* [**src** *SOURCE*] [**permanent** | **temp**] [**vid** *VID*] [**filter_mode** { **include** | **exclude** }] [**source_list** *SOURCE_LIST*] [**proto** *PROTO*] [**dst** *IPADDR*] [**dst_port** *DST_PORT*] [**vni** *VNI*] [**src_vni** *SRC_VNI*] [**via** *DEV*]

bridge mdb show [**dev** *DEV*]

bridge mdb get dev *DEV* **grp** *GROUP* [**src** *SOURCE*] [**vid** *VID*] [**src_vni** *SRC_VNI*]

bridge mdb flush dev *DEV* [**port** *PORT*] [**vid** *VID*] [**src_vni** *SRC_VNI*] [**proto** *PROTO*] [**[no]permanent**] [**dst** *IPADDR*] [**dst_port** *DST_PORT*] [**vni** *VNI*]

bridge mst set dev *DEV* *msti* *MSTI* *state* *STP_STATE*

bridge mst [[**show**] [**dev** *DEV*]]

```
bridge vlan { add | del } dev DEV vid VID [ tunnel_info TUNNEL_ID ] [ pvid ] [ untagged ] [ self ] [ master ]
```

```
bridge vlan set dev DEV vid VID [ state STP_STATE ] [ mcast_max_groups MAX_GROUPS ] [ mcast_router MULTICAST_ROUTER ] [ neigh_suppress { on | off } ]
```

```
bridge vlan [ show | tunnelshow ] [ dev DEV ]
```

```
bridge vlan global set dev DEV vid VID [ mcast_snooping MULTICAST_SNOOPING ] [ mcast_querier MULTICAST_QUERIER ] [ mcast_igmp_version IGMP_VERSION ] [ mcast_mld_version MLD_VERSION ] [ mcast_last_member_count LAST_MEMBER_COUNT ] [ mcast_last_member_interval LAST_MEMBER_INTERVAL ] [ mcast_startup_query_count STARTUP_QUERY_COUNT ] [ mcast_startup_query_interval STARTUP_QUERY_INTERVAL ] [ mcast_membership_interval MEMBERSHIP_INTERVAL ] [ mcast_querier_interval QUERIER_INTERVAL ] [ mcast_query_interval QUERY_INTERVAL ] [ mcast_query_response_interval QUERY_RESPONSE_INTERVAL ] [ msti MSTI ]
```

```
bridge vlan global [ show ] [ dev DEV ] [ vid VID ]
```

```
bridge vlan show [ dev DEV ]
```

```
bridge vni { add | del } dev DEV vni VNI [ { group | remote } IPADDR ]
```

```
bridge vni show [ dev DEV ]
```

```
bridge monitor [ all | neigh | link | mdb | vlan ]
```

OPTIONS

-V, -Version

print the version of the **bridge** utility and exit.

-s, -stats, -statistics

output more information. If this option is given multiple times, the amount of information increases. As a rule, the information is statistics or some time values.

-d, -details

print detailed information about bridge vlan filter entries or MDB router ports.

-n, -net, -netns <NETNS>

switches **bridge** to the specified network namespace *NETNS*. Actually it just simplifies executing of:

```
ip netns exec NETNS bridge [ OPTIONS ] OBJECT { COMMAND | help }
```

to

```
bridge -n[etns] NETNS [ OPTIONS ] OBJECT { COMMAND | help }
```

-b, -batch <FILENAME>

Read commands from provided file or standard input and invoke them. First failure will cause termination of bridge command.

- force** Don't terminate bridge command on errors in batch mode. If there were any errors during execution of the commands, the application return code will be non zero.
- com, -compressvlans**
Show compressed VLAN list. It will show continuous VLANs with the range instead of separated VLANs. Default is off.
- c[*color*][=*{always|auto|never}*]**
Configure color output. If parameter is omitted or **always**, color output is enabled regardless of stdout state. If parameter is **auto**, stdout is checked to be a terminal before enabling color output. If parameter is **never**, color output is disabled. If specified multiple times, the last one takes precedence. This flag is ignored if **-json** is also given.
- j, -json**
Output results in JavaScript Object Notation (JSON).
- p, -pretty**
When combined with -j generate a pretty JSON output.
- o, -oneline**
output each record on a single line, replacing line feeds with the '\n' character. This is convenient when you want to count records with **wc(1)** or to **grep(1)** the output.

BRIDGE - COMMAND SYNTAX

OBJECT

- link** - Bridge port.
- fdb** - Forwarding Database entry.
- mdb** - Multicast group database entry.
- vlan** - VLAN filter list.
- vni** - VNI filter list.

COMMAND

Specifies the action to perform on the object. The set of possible actions depends on the object type. As a rule, it is possible to **add**, **delete** and **show** (or **list**) objects, but some objects do not allow all of these operations or have some additional commands. The **help** command is available for all objects. It prints out a list of available commands and argument syntax conventions.

If no command is given, some default command is assumed. Usually it is **list** or, if the objects of this class cannot be listed, **help**.

bridge link - bridge port

link objects correspond to the port devices of the bridge.

The corresponding commands set and display port status and bridge specific attributes.

bridge link set - set bridge specific attributes on a port**dev** *NAME*

interface name of the bridge port

cost *COST*

the STP path cost of the specified port.

priority *PRIO*

the STP port priority. The priority value is an unsigned 8-bit quantity (number between 0 and 255). This metric is used in the designated port and root port selection algorithms.

state *STATE*

the operation state of the port. Except state 0 (disable STP or BPDU filter feature), this is primarily used by user space STP/RSTP implementation. One may enter port state name (case insensitive), or one of the numbers below. Negative inputs are ignored, and unrecognized names return an error.

0 - port is in STP **DISABLED** state. Make this port completely inactive for STP. This is also called BPDU filter and could be used to disable STP on an untrusted port, like a leaf virtual devices.

1 - port is in STP **LISTENING** state. Only valid if STP is enabled on the bridge. In this state the port listens for STP BPDUs and drops all other traffic frames.

2 - port is in STP **LEARNING** state. Only valid if STP is enabled on the bridge. In this state the port will accept traffic only for the purpose of updating MAC address tables.

3 - port is in STP **FORWARDING** state. Port is fully active.

4 - port is in STP **BLOCKING** state. Only valid if STP is enabled on the bridge. This state is used during the STP election process. In this state, port will only process STP BPDUs.

guard on or guard off

Controls whether STP BPDUs will be processed by the bridge port. By default, the flag is turned off allowed BPDU processing. Turning this flag on will disable the bridge port if a STP BPDU packet is received.

If running Spanning Tree on bridge, hostile devices on the network may send BPDU on a port and cause network failure. Setting **guard on** will detect and stop this by disabling the port. The port will be restarted if link is brought down, or removed and reattached. For example if guard is enable on eth0:

ip link set dev eth0 down; ip link set dev eth0 up

hairpin on or hairpin off

Controls whether traffic may be sent back out of the port on which it was received. This option is also called reflective relay mode, and is used to support basic VEPA (Virtual Ethernet Port Aggregator) capabilities. By default, this flag is turned off and the bridge will not forward traffic back

out of the receiving port.

fastleave on or fastleave off

This flag allows the bridge to immediately stop multicast traffic on a port that receives IGMP Leave message. It is only used with IGMP snooping is enabled on the bridge. By default the flag is off.

root_block on or root_block off

Controls whether a given port is allowed to become root port or not. Only used when STP is enabled on the bridge. By default the flag is off.

This feature is also called root port guard. If BPDU is received from a leaf (edge) port, it should not be elected as root port. This could be used if using STP on a bridge and the downstream bridges are not fully trusted; this prevents a hostile guest from rerouting traffic.

learning on or learning off

Controls whether a given port will learn MAC addresses from received traffic or not. If learning is off, the bridge will end up flooding any traffic for which it has no FDB entry. By default this flag is on.

learning_sync on or learning_sync off

Controls whether a given port will sync MAC addresses learned on device port to bridge FDB.

flood on or flood off

Controls whether unicast traffic for which there is no FDB entry will be flooded towards this given port. By default this flag is on.

hwmode

Some network interface cards support HW bridge functionality and they may be configured in different modes. Currently support modes are:

vepa - Data sent between HW ports is sent on the wire to the external switch.

veb - bridging happens in hardware.

bcast_flood on or bcast_flood off

Controls flooding of broadcast traffic on the given port. By default this flag is on.

mcast_flood on or mcast_flood off

Controls whether multicast traffic for which there is no MDB entry will be flooded towards this given port. By default this flag is on.

mcast_max_groups MAX_GROUPS

Sets the maximum number of MDB entries that can be registered for a given port. Attempts to register more MDB entries at the port than this limit allows will be rejected, whether they are done through netlink (e.g. the **bridge** tool), or IGMP or MLD membership reports. Setting a limit to 0 has the effect of disabling the limit. The default value is 0. See also the **ip link** option **mcast_hash_max**.

mcast_router MULTICAST_ROUTER

This flag is almost the same as the per-VLAN flag, see below, except its value can only be set in the range 0-2. The default is **1** where the bridge figures out automatically where an IGMP/MLD querier, MRDISC capable device, or PIM router, is located. Setting this flag to **2** is useful in cases where the multicast router does not indicate its presence in any meaningful way (e.g. older versions of SMCRoute, or mrouted), or when there is a need for forwarding both known and unknown IP multicast to a secondary/backup router.

mcast_to_unicast on or mcast_to_unicast off

Controls whether a given port will replicate packets using unicast instead of multicast. By default this flag is off.

This is done by copying the packet per host and changing the multicast destination MAC to a unicast one accordingly.

mcast_to_unicast works on top of the multicast snooping feature of the bridge. Which means unicast copies are only delivered to hosts which are interested in it and signaled this via IGMP/MLD reports previously.

This feature is intended for interface types which have a more reliable and/or efficient way to deliver unicast packets than broadcast ones (e.g. WiFi).

However, it should only be enabled on interfaces where no IGMPv2/MLDv1 report suppression takes place. IGMP/MLD report suppression issue is usually overcome by the network daemon (supplicant) enabling AP isolation and by that separating all STAs.

Delivery of STA-to-STA IP multicast is made possible again by enabling and utilizing the bridge hairpin mode, which considers the incoming port as a potential outgoing port, too (see **hairpin** option). Hairpin mode is performed after multicast snooping, therefore leading to only deliver reports to STAs running a multicast router.

neigh_suppress on or neigh_suppress off

Controls whether neigh discovery (arp and nd) proxy and suppression is enabled on the port. By default this flag is off.

neigh_vlan_suppress on or neigh_vlan_suppress off

Controls whether per-VLAN neigh discovery (arp and nd) proxy and suppression is enabled on the port. When on, the **bridge link** option **neigh_suppress** has no effect and the per-VLAN state is set using the **bridge vlan** option **neigh_suppress**. By default this flag is off.

vlan_tunnel on or vlan_tunnel off

Controls whether vlan to tunnel mapping is enabled on the port. By default this flag is off.

isolated on or isolated off

Controls whether a given port will be isolated, which means it will be able to communicate with non-isolated ports only. By default this flag is off.

locked on or locked off

Controls whether a port is locked or not. When locked, non-link-local frames received through the port are dropped unless an FDB entry with the MAC source address points to the port. The common use case is IEEE 802.1X where hosts can authenticate themselves by exchanging EAPOL frames with an authenticator. After authentication is complete, the user space control plane can

install a matching FDB entry to allow traffic from the host to be forwarded by the bridge. When learning is enabled on a locked port, the **no_linklocal_learn** bridge option needs to be on to prevent the bridge from learning from received EAPOL frames. By default this flag is off.

mab on or mab off

Controls whether MAC Authentication Bypass (MAB) is enabled on the port or not. MAB can only be enabled on a locked port that has learning enabled. When enabled, FDB entries are learned from received traffic and have the "locked" FDB flag set. The flag can only be set by the kernel and it indicates that the FDB entry cannot be used to authenticate the corresponding host. User space can decide to authenticate the host by replacing the FDB entry and clearing the "locked" FDB flag. Locked FDB entries can roam to unlocked (authorized) ports in which case the "locked" flag is cleared. FDB entries cannot roam to locked ports regardless of MAB being enabled or not. Therefore, locked FDB entries are only created if an FDB entry with the given {MAC, VID} does not already exist. This behavior prevents unauthenticated hosts from disrupting traffic destined to already authenticated hosts. Locked FDB entries act like regular dynamic entries with respect to forwarding and aging. 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

backup_nhid NHID

The FDB nexthop object ID (see **ip-nexthop(8)**) to attach to packets being redirected to a backup port that has VLAN tunnel mapping enabled (via the **vlan_tunnel** option). Setting a value of 0 (default) has the effect of not attaching any ID.

self link setting is configured on specified physical device

master link setting is configured on the software bridge (default)

-t, -timestamp

display current time when using monitor option.

bridge link show - list ports configuration for all bridges.

This command displays ports configuration and flags for all bridges by default.

dev DEV

only display the specific bridge port named DEV.

master DEVICE

only display ports of the bridge named DEVICE. This is similar to "ip link show master <bridge_device>" command.

bridge fdb - forwarding database management

fdb objects contain known Ethernet addresses on a link.

The corresponding commands display fdb entries, add new entries, append entries, and delete old ones.

bridge fdb add - add a new fdb entry

This command creates a new fdb entry.

LLADDR

the Ethernet MAC address.

dev DEV

the interface to which this address is associated.

local - is a local permanent fdb entry, which means that the bridge will not forward frames with this destination MAC address and VLAN ID, but terminate them locally. This flag is default unless "static" or "dynamic" are explicitly specified.

permanent - this is a synonym for "local"

static - is a static (no arp) fdb entry

dynamic - is a dynamic reachable age-able fdb entry

self - the operation is fulfilled directly by the driver for the specified network device. If the network device belongs to a master like a bridge, then the bridge is bypassed and not notified of this operation (and if the device does notify the bridge, it is driver-specific behavior and not mandated by this flag, check the driver for more details). The "bridge fdb add" command can also be used on the bridge device itself, and in this case, the added fdb entries will be locally terminated (not forwarded). In the latter case, the "self" flag is mandatory. The flag is set by default if "master" is not specified.

master - if the specified network device is a port that belongs to a master device such as a bridge, the operation is fulfilled by the master device's driver, which may in turn notify the port driver too of the address. If the specified device is a master itself, such as a bridge, this flag is invalid.

router - the destination address is associated with a router. Valid if the referenced device is a VXLAN type device and has route short circuit enabled.

use - the address is in use. User space can use this option to indicate to the kernel that the fdb entry is in use.

extern_learn - this entry was learned externally. This option can be used to indicate to the kernel that an entry was hardware or user-space controller learnt dynamic entry. Kernel will not age such an entry.

sticky - this entry will not change its port due to learning.

The next command line parameters apply only when the specified device *DEV* is of type VXLAN.

dst IPADDR

the IP address of the destination VXLAN tunnel endpoint where the Ethernet MAC ADDRESS resides.

src_vni VNI

the src VNI Network Identifier (or VXLAN Segment ID) this entry belongs to. Used only when the vxlan device is in external or collect metadata mode. If omitted the value specified at vxlan device creation will be used.

vni VNI

the VXLAN VNI Network Identifier (or VXLAN Segment ID) to use to connect to the remote VXLAN tunnel endpoint. If omitted the value specified at vxlan device creation will be used.

port PORT

the UDP destination PORT number to use to connect to the remote VXLAN tunnel endpoint. If omitted the default value is used.

via DEVICE

device name of the outgoing interface for the VXLAN device driver to reach the remote VXLAN tunnel endpoint.

nhid NHID

ecmp nexthop group for the VXLAN device driver to reach remote VXLAN tunnel endpoints.

activity_notify

enable activity notifications on an existing or a new FDB entry. This keyword only makes sense for non-dynamic entries as dynamic entries are deleted upon inactivity. An entry is assumed to be active unless the **inactive** keyword is specified.

inactive

mark an FDB entry as inactive. This keyword only makes sense in conjunction with the **activity_notify** keyword and usually only when adding a new FDB entry as opposed to replacing an existing one.

norefresh

avoid resetting an FDB entry's activity (i.e., its last updated time). This can be useful, for example, when one wants to enable activity notifications on an existing entry without modifying its last updated time.

bridge fdb append - append a forwarding database entry

This command adds a new fdb entry with an already known *LLADDR*. Valid only for multicast link layer addresses. The command adds support for broadcast and multicast Ethernet MAC addresses. The Ethernet MAC address is added multiple times into the forwarding database and the vxlan device driver sends a copy of the data packet to each entry found.

The arguments are the same as with **bridge fdb add**.

bridge fdb delete - delete a forwarding database entry

This command removes an existing fdb entry.

The arguments are the same as with **bridge fdb add**.

bridge fdb replace - replace a forwarding database entry

If no matching entry is found, a new one will be created instead.

The arguments are the same as with **bridge fdb add**.

bridge fdb show - list forwarding entries.

This command displays the current forwarding table.

With the **-statistics** option, the command becomes verbose. It prints out the last updated and last used time for each entry.

bridge fdb get - get bridge forwarding entry.

lookup a bridge forwarding table entry.

LLADDR

the Ethernet MAC address.

dev DEV

the interface to which this address is associated.

brport DEV

the bridge port to which this address is associated. same as dev above.

br DEV

the bridge to which this address is associated.

self - the address is associated with the port drivers fdb. Usually hardware.

master - the address is associated with master devices fdb. Usually software (default).

bridge fdb flush - flush bridge forwarding table entries.

flush the matching bridge forwarding table entries. Some options below have a negated form when "no" is prepended to them (e.g. permanent and nopermanent).

dev DEV

the target device for the operation. If the device is a bridge port and "master" is set then the operation will be fulfilled by its master device's driver and all entries pointing to that port will be deleted.

brport DEV

the target bridge port for the operation. If the bridge device is specified then only entries pointing to the bridge itself will be deleted. Note that the target device specified by this option will override the one specified by dev above.

vlan *VID*

the target VLAN ID for the operation. Match forwarding table entries only with the specified VLAN ID.

src_vni *VNI*

the src VNI Network Identifier (or VXLAN Segment ID) for the operation. Match forwarding table entries only with the specified VNI. Valid if the referenced device is a VXLAN type device.

nhid *NHID*

the ECMP nexthop group for the operation. Match forwarding table entries only with the specified NHID. Valid if the referenced device is a VXLAN type device.

vni *VNI*

the VXLAN VNI Network Identifier (or VXLAN Segment ID) for the operation. Match forwarding table entries only with the specified VNI. Valid if the referenced device is a VXLAN type device.

port *PORT*

the UDP destination PORT number for the operation. Match forwarding table entries only with the specified PORT. Valid if the referenced device is a VXLAN type device.

dst *IPADDR*

the IP address of the destination VXLAN tunnel endpoint for the operation. Match forwarding table entries only with the specified IPADDR. Valid if the referenced device is a VXLAN type device.

self

the operation is fulfilled directly by the driver for the specified network device. If the network device belongs to a master like a bridge, then the bridge is bypassed and not notified of this operation. The "bridge fdb flush" command can also be used on the bridge device itself. The flag is set by default if "master" is not specified.

master

if the specified network device is a port that belongs to a master device such as a bridge, the operation is fulfilled by the master device's driver. Flush with both 'master' and 'self' is not recommended with attributes that are not supported by all devices (e.g., vlan, vni). Such command will be handled by bridge or VXLAN driver, but will return an error from the driver that does not support the attribute. Instead, run flush twice - once with 'self' and once with 'master', and each one with the supported attributes.

[no]permanent

if specified then only permanent entries will be deleted or respectively if "no" is prepended then only non-permanent entries will be deleted.

[no]static

if specified then only static entries will be deleted or respectively if "no" is prepended then only non-static entries will be deleted.

[no]dynamic

if specified then only dynamic entries will be deleted or respectively if "no" is prepended then only non-dynamic (static or permanent) entries will be deleted.

[no]added_by_user

if specified then only entries with added_by_user flag will be deleted or respectively if "no" is prepended then only entries without added_by_user flag will be deleted.

[no]extern_learn

if specified then only entries with extern_learn flag will be deleted or respectively if "no" is prepended then only entries without extern_learn flag will be deleted.

[no]sticky

if specified then only entries with sticky flag will be deleted or respectively if "no" is prepended then only entries without sticky flag will be deleted.

[no]offloaded

if specified then only entries with offloaded flag will be deleted or respectively if "no" is prepended then only entries without offloaded flag will be deleted.

[no]router

if specified then only entries with router flag will be deleted or respectively if "no" is prepended then only entries without router flag will be deleted. Valid if the referenced device is a VXLAN type device.

bridge mdb - multicast group database management

mdb objects contain known IP or L2 multicast group addresses on a link.

The corresponding commands display mdb entries, add new entries, replace entries and delete old ones.

bridge mdb add - add a new multicast group database entry

This command creates a new mdb entry.

dev *DEV*

the interface where this group address is associated.

port *PORT*

the port whose link is known to have members of this multicast group.

grp *GROUP*

the multicast group address (IPv4, IPv6 or L2 multicast) whose members reside on the link connected to the port.

permanent - the mdb entry is permanent. Optional for IPv4 and IPv6, mandatory for L2.

temp - the mdb entry is temporary (default)

src *SOURCE*

optional source IP address of a sender for this multicast group. If IGMPv3 for IPv4, or MLDv2 for IPv6 respectively, are enabled it will be included in the lookup when forwarding multicast traffic.

vid *VID*

the VLAN ID which is known to have members of this multicast group.

filter_mode *include* or **filter_mode** *exclude*

controls whether the sources in the entry's source list are in INCLUDE or EXCLUDE mode. Can only be set for (*, G) entries.

source_list *SOURCE_LIST*

optional list of source IP addresses of senders for this multicast group, separated by a ','. Whether the entry forwards packets from these senders or not is determined by the entry's filter mode, which becomes a mandatory argument. Can only be set for (*, G) entries.

proto *PROTO*

the routing protocol identifier of this mdb entry. Can be a number or a string from the file /etc/iproute2/route/protocols. If the routing protocol is not given, then **static** is assumed.

The next command line parameters apply only when the specified device *DEV* is of type VXLAN.

dst *IPADDR*

the IP address of the destination VXLAN tunnel endpoint where the multicast receivers reside.

dst_port *DST_PORT*

the UDP destination port number to use to connect to the remote VXLAN tunnel endpoint. If omitted, the value specified at VXLAN device creation will be used.

vni *VNI*

the VXLAN VNI Network Identifier to use to connect to the remote VXLAN tunnel endpoint. If omitted, the value specified at VXLAN device creation will be used or the source VNI when the VXLAN device is in external mode.

src_vni *SRC_VNI*

the source VNI Network Identifier this entry belongs to. Used only when the VXLAN device is in external mode. If omitted, the value specified at VXLAN device creation will be used.

via *DEV*

device name of the outgoing interface for the VXLAN device to reach the remote VXLAN tunnel endpoint.

The 0.0.0.0 and :: MDB entries are special catchall entries used to flood IPv4 and IPv6 unregistered multicast packets, respectively. Therefore, when these entries are programmed, the catchall 00:00:00:00:00:00 FDB entry will only flood broadcast, unknown unicast and link-local multicast.

bridge mdb delete - delete a multicast group database entry

This command removes an existing mdb entry.

The arguments are the same as with **bridge mdb add**.

bridge mdb replace - replace a multicast group database entry

If no matching entry is found, a new one will be created instead.

The arguments are the same as with **bridge mdb add**.

bridge mdb show - list multicast group database entries

This command displays the current multicast group membership table. The table is populated by IGMP and MLD snooping in the bridge driver automatically. It can be altered by **bridge mdb add** and **bridge mdb del** commands manually too.

dev *DEV*

the interface only whose entries should be listed. Default is to list all bridge interfaces.

With the **-details** option, the command becomes verbose. It prints out the ports known to have a connected router.

With the **-statistics** option, the command displays timer values for mdb and router port entries.

bridge mdb get - get multicast group database entry.

This command retrieves a multicast group database entry based on its key.

dev *DEV*

the interface where this group address is associated.

grp *GROUP*

the multicast group address (IPv4, IPv6 or L2 multicast).

src *SOURCE*

the source IP address. Only relevant when retrieving an (S, G) entry.

vid *VID*

the VLAN ID. Only relevant when the bridge is VLAN-aware.

src_vni *SRC_VNI*

the source VNI Network Identifier. Only relevant when the VXLAN device is in external mode.

bridge mdb flush - flush multicast group database entries.

This command flushes the matching multicast group database entries.

dev *DEV*

the interface where this group address is associated.

port *PORT*

the target port for the operation. If the bridge device is specified then only entries pointing to the bridge itself will be deleted.

vid *VID*

the VLAN ID for the operation. Match entries only with the specified VLAN ID.

src_vni *SRC_VNI*

the source VNI Network Identifier for the operation. Match entries only with the specified source VNI.

proto *PROTO*

the routing protocol identifier for the operation. Match entries only with the specified routing protocol. Can be a number or a string from the file `/etc/iproute2/rt_protos`.

[no]permanent

if specified then only permanent entries will be deleted or respectively if "no" is prepended then only non-permanent (temp) entries will be deleted.

dst *IPADDR*

the IP address of the destination VXLAN tunnel endpoint where the multicast receivers reside. Match entries only with the specified destination IP.

dst_port *DST_PORT*

the UDP destination port number to use to connect to the remote VXLAN tunnel endpoint. Match entries only with the specified destination port number.

vni *VNI*

the VXLAN VNI Network Identifier to use to connect to the remote VXLAN tunnel endpoint. Match entries only with the specified destination VNI.

bridge mst - multiple spanning tree port states

In the multiple spanning tree (MST) model, the active paths through a network can be different for different VLANs. In other words, a bridge port can simultaneously forward one subset of VLANs, while blocking another.

Provided that the **mst_enable** bridge option is enabled, a group of VLANs can be forwarded along the same spanning tree by associating them with the same instance (MSTI) using **bridge vlan global set**.

bridge mst set - set multiple spanning tree state

Set the spanning tree state for *DEV*, in the multiple spanning tree instance *MSTI*, to *STP_STATE*.

dev *DEV*

Interface name of the bridge port.

msti *MSTI*

The multiple spanning tree instance.

state *STP_STATE*

The spanning tree state, see the **state** option of **bridge link set** for supported states.

bridge mst show - list MST states

List current MST port states in every MSTI.

dev *DEV*

If specified, only display states of the bridge port with this interface name.

bridge vlan - VLAN filter list

vlan objects contain known VLAN IDs for a link.

The corresponding commands display vlan filter entries, add new entries, and delete old ones.

bridge vlan add - add a new vlan filter entry

This command creates a new vlan filter entry.

dev *NAME*

the interface with which this vlan is associated.

vid *VID*

the VLAN ID that identifies the vlan.

tunnel_info *TUNNEL_ID*

the TUNNEL ID that maps to this vlan. The tunnel id is set in `dst_metadata` for every packet that belongs to this vlan (applicable to bridge ports with `vlan_tunnel` flag set).

pvid the vlan specified is to be considered a PVID at ingress. Any untagged frames will be assigned to this VLAN.

untagged

the vlan specified is to be treated as untagged on egress.

self the vlan is configured on the specified physical device. Required if the device is the bridge device.

master the vlan is configured on the software bridge (default).

bridge vlan delete - delete a vlan filter entry

This command removes an existing vlan filter entry.

The arguments are the same as with **bridge vlan add**. The **pvid** and **untagged** flags are ignored.

bridge vlan set - change vlan filter entry's options

This command changes vlan filter entry's options.

dev *NAME*

the interface with which this vlan is associated.

vid *VID*

the VLAN ID that identifies the vlan.

state *STP_STATE*

the operation state of the vlan. One may enter STP state name (case insensitive), or one of the numbers below. Negative inputs are ignored, and unrecognized names return an error. Note that the state is set only for the vlan of the specified device, e.g. if it is a bridge port then the state will be set only for the vlan of the port.

0 - vlan is in STP **DISABLED** state. Make this vlan completely inactive for STP. This is also called BPDU filter and could be used to disable STP on an untrusted vlan.

1 - vlan is in STP **LISTENING** state. Only valid if STP is enabled on the bridge. In this state the vlan listens for STP BPDUs and drops all other traffic frames.

2 - vlan is in STP **LEARNING** state. Only valid if STP is enabled on the bridge. In this state the vlan will accept traffic only for the purpose of updating MAC address tables.

3 - vlan is in STP **FORWARDING** state. This is the default vlan state.

4 - vlan is in STP **BLOCKING** state. Only valid if STP is enabled on the bridge. This state is used during the STP election process. In this state, the vlan will only process STP BPDUs.

mcast_max_groups *MAX_GROUPS*

Sets the maximum number of MDB entries that can be registered for a given VLAN on a given port. A VLAN-specific equivalent of the per-port option of the same name, see above for details.

Note that this option is only available when **ip link** option **mcast_vlan_snooping** is enabled.

mcast_router *MULTICAST_ROUTER*

configure this vlan and interface's multicast router mode, note that only modes 0 - 2 are available for bridge devices. A vlan and interface with a multicast router will receive all multicast traffic. *MULTICAST_ROUTER* may be either

0 - to disable multicast router.

1 - to let the system detect the presence of routers (default).

2 - to permanently enable multicast traffic forwarding on this vlan and interface.

3 - to temporarily mark this vlan and port as having a multicast router, i.e. enable multicast traffic forwarding. This mode is available only for ports.

neigh_suppress on or **neigh_suppress off**

Controls whether neigh discovery (arp and nd) proxy and suppression is enabled for a given VLAN on a given port. By default this flag is off.

Note that this option only takes effect when **bridge link** option **neigh_vlan_suppress** is enabled for a given port.

bridge vlan show - list vlan configuration.

This command displays the current VLAN filter table.

With the **-details** option, the command becomes verbose. It displays the per-vlan options.

With the **-statistics** option, the command displays per-vlan traffic statistics.

bridge vlan tunnelshow - list vlan tunnel mapping.

This command displays the current vlan tunnel info mapping.

bridge vlan global set - change vlan filter entry's global options

This command changes vlan filter entry's global options.

dev *NAME*

the interface with which this vlan is associated. Only bridge devices are supported for global options.

vid *VID*

the VLAN ID that identifies the vlan.

mcast_snooping *MULTICAST_SNOOPING*

turn multicast snooping for VLAN entry with VLAN ID on (*MULTICAST_SNOOPING* > 0) or off (*MULTICAST_SNOOPING* == 0). Default is on.

mcast_querier *MULTICAST_QUERIER*

enable (*MULTICAST_QUERIER* > 0) or disable (*MULTICAST_QUERIER* == 0) IGMP/MLD querier, ie sending of multicast queries by the bridge. Default is disabled.

mcast_igmp_version *IGMP_VERSION*

set the IGMP version. Default is 2.

mcast_mld_version *MLD_VERSION*

set the MLD version. Default is 1.

mcast_last_member_count *LAST_MEMBER_COUNT*

set multicast last member count, ie the number of queries the bridge will send before stopping forwarding a multicast group after a "leave" message has been received. Default is 2.

mcast_last_member_interval *LAST_MEMBER_INTERVAL*

interval between queries to find remaining members of a group, after a "leave" message is received.

mcast_startup_query_count *STARTUP_QUERY_COUNT*

set the number of queries to send during startup phase. Default is 2.

mcast_startup_query_interval *STARTUP_QUERY_INTERVAL*

interval between queries in the startup phase.

mcast_membership_interval *MEMBERSHIP_INTERVAL*

delay after which the bridge will leave a group, if no membership reports for this group are received.

mcast_querier_interval *QUERIER_INTERVAL*

interval between queries sent by other routers. If no queries are seen after this delay has passed, the bridge will start to send its own queries (as if **mcast_querier** was enabled).

mcast_query_interval *QUERY_INTERVAL*

interval between queries sent by the bridge after the end of the startup phase.

mcast_query_response_interval *QUERY_RESPONSE_INTERVAL*

set the Max Response Time/Maximum Response Delay for IGMP/MLD queries sent by the bridge.

msti *MSTI*

associate the VLAN with the specified multiple spanning tree instance (MSTI).

bridge vlan global show - list global vlan options.

This command displays the global VLAN options for each VLAN entry.

dev *DEV*

the interface only whose VLAN global options should be listed. Default is to list all bridge interfaces.

vid *VID*

the VLAN ID only whose global options should be listed. Default is to list all vlans.

bridge vni - VNI filter list

vni objects contain known VNI IDs for a dst metadata vxlan link.

The corresponding commands display vni filter entries, add new entries, and delete old ones.

bridge vni add - add a new vni filter entry

This command creates a new vni filter entry.

dev *NAME*

the interface with which this vni is associated.

vni *VNI*

the VNI ID that identifies the vni.

remote *IPADDR*

specifies the unicast destination IP address to use in outgoing packets when the destination link layer address is not known in the VXLAN device forwarding database. This parameter cannot be specified with the group.

group *IPADDR*

specifies the multicast IP address to join for this VNI

bridge vni del - delete a new vni filter entry

This command removes an existing vni filter entry.

The arguments are the same as with **bridge vni add**.

bridge vni show - list vni filtering configuration.

This command displays the current vni filter table.

With the **-statistics** option, the command displays per-vni traffic statistics.

dev *NAME*

shows vni filtering table associated with the vxlan device

bridge monitor - state monitoring

The **bridge** utility can monitor the state of devices and addresses continuously. This option has a slightly different format. Namely, the **monitor** command is the first in the command line and then the object list follows:

bridge monitor [**all** | *OBJECT-LIST*]

OBJECT-LIST is the list of object types that we want to monitor. It may contain **link**, **fdb**, **vlan** and **mdb**. If no **file** argument is given, **bridge** opens RTNETLINK, listens on it and dumps state changes in the format described in previous sections.

If a file name is given, it does not listen on RTNETLINK, but opens the file containing RTNETLINK messages saved in binary format and dumps them.

NOTES

This command uses facilities added in Linux 3.0.

Although the forwarding table is maintained on a per-bridge device basis the bridge device is not part of the syntax. This is a limitation of the underlying netlink neighbour message protocol. When displaying the forwarding table, entries for all bridges are displayed. Add/delete/modify commands determine the underlying bridge device based on the bridge to which the corresponding ethernet device is attached.

SEE ALSO

ip(8)

BUGS

Please direct bugreports and patches to: <netdev@vger.kernel.org>

AUTHOR

Original Manpage by Stephen Hemminger