bridge(8) — Linux manual page

NAME top

bridge - show / manipulate bridge addresses and devices

SYNOPSIS top

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

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

OPTIONS := { **-V**[*ersion*] | **-s**[*tatistics*] | **-n**[*etns*] name |
-b[*atch*] filename | **-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** }] [**mcast_flood** { **on** | **off** }] [**mcast_to_unicast** { **on** | **off** }] [**neigh_suppress** { **on** | **off** }] [**vlan_tunnel** { **on** | **off** }] [**isolated** { **on** | **off** }] [**backup_port** *DEVICE*] [**nobackup_port**] [**self**] [**master**]

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

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* }

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 mdb { **add** | **del** } **dev** *DEV* **port** *PORT* **grp** *GROUP* [**src** *SOURCE*] [**permanent** | **temp**] [**vid** *VID*]

bridge mdb 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*]

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

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

OPTIONS [top]

-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.

-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 '\' character. This is convenient when you want to count records with wc(1) or to grep(1) the output.

BRIDGE - COMMAND SYNTAX top

OBJECT

link - Bridge port.

fdb - Forwarding Database entry.

mdb - Multicast group database entry.

vlan - VLAN 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 top

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 an droot 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 disables 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 send 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 if 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.

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_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 signalized 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.

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.

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

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 port configuration and flags for all bridges.

To display port configuration and flags for a specific bridge, use the "ip link show master <bridge_device>" command.

bridge monitor - state monitoring top

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 top

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 top

ip(8)

BUGS top

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

AUTHOR top

Original Manpage by Stephen Hemminger

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