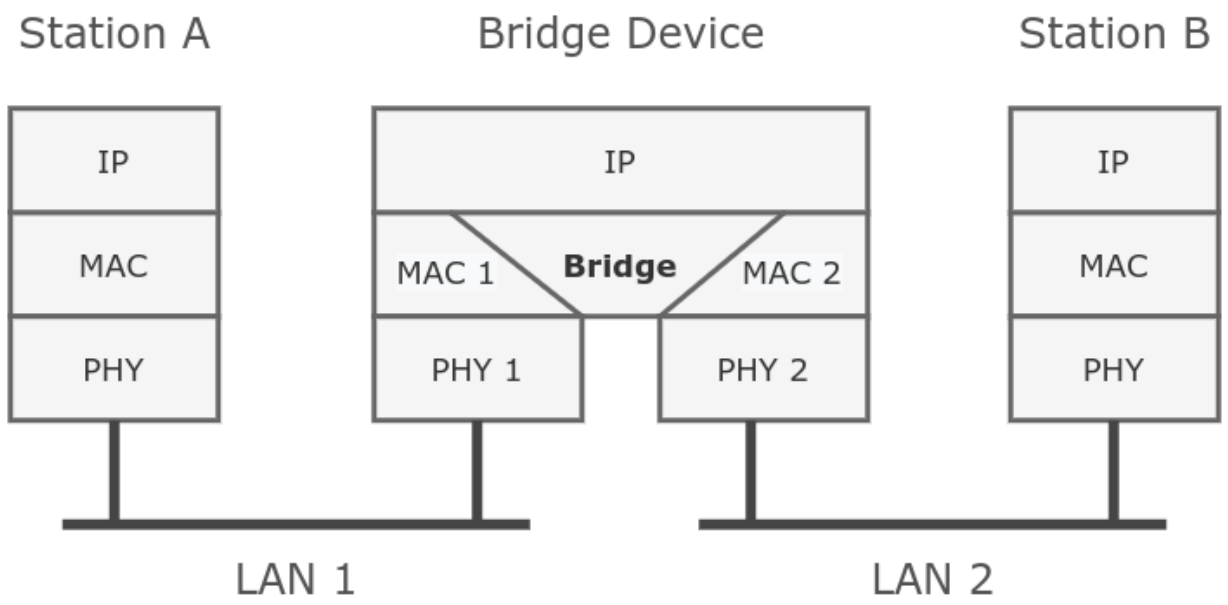
Bridging and Switching

Other resources:

Summary



Ethernet-like networks (Ethernet, Ethernet over IP, IEEE 802.11 in ap-bridge or bridge mode, WDS, VLAN) can be connected together using MAC bridges. The bridge feature allows the interconnection of hosts connected to separate LANs (using EoIP, geographically distributed networks can be bridged as well if any kind of IP network interconnection exists between them) as if they were attached to a single LAN. As bridges are transparent, they do not appear in the traceroute list, and no utility can make a distinction between a host working in one LAN and a host working in another LAN if these LANs are bridged. However, depending on the way the LANs are interconnected, latency and data rate between hosts may vary.

Network loops may emerge (intentionally or not) in complex topologies. Without any special treatment, loops would prevent the network from functioning normally, as they would lead to avalanche-like packet multiplication. Each bridge runs an algorithm that calculates how the loop can be prevented. (R/M)STP allows bridges to communicate with each other, so they can negotiate a loop-free topology. All other alternative connections that would otherwise form loops are put on standby, so that should the main connection fail, another connection could take its place. This algorithm exchanges configuration messages (BPDU - Bridge Protocol Data Unit) periodically, so that all bridges are updated with the newest information about changes in a network topology. (R/M)STP selects a root bridge which is responsible for network reconfiguration, such as blocking and opening ports on other bridges. The root bridge is the bridge with the lowest bridge ID.

Bridge Interface Setup

To combine a number of networks into one bridge, a bridge interface should be created. Later, all the desired interfaces should be set up as its ports. One MAC address from slave (secondary) ports will be assigned to the bridge interface. The MAC address will be chosen automatically, depending on the "port-number", and it can change after a reboot. To avoid unwanted MAC address changes, it is recommended to disable "`auto-mac`" and manually specifying the MAC address by using "`admin-mac`".

Sub-menu: `/interface bridge`

Property	Description
add-dhcp-option82 (*yes* | *no*; Default: **no**)	Whether to add DHCP Option-82 information (Agent Remote ID and Agent Circuit ID) to DHCP packets. Can be used together with Option-82 capable DHCP server to assign IP addresses and implement policies. This property only has an effect when `dhcp-snooping` is set to `yes`.
admin-mac (*MAC address*; Default: **none**)	Static MAC address of the bridge. This property only has an effect when `auto-mac` is set to `no`.
ageing-time (*time*; Default: **00:05:00**)	How long a host's information will be kept in the bridge database.

arp (*disabled* | *enabled* | *local-proxy-arp* | *proxy-arp* | *reply-only*; Default: **enabled**)	Address Resolution Protocol setting • `disabled` - the interface will not use ARP • `enabled` - the interface will use ARP • `local-proxy-arp` - the router performs proxy ARP on the interface and sends replies to the same interface • `proxy-arp` - the router performs proxy ARP on the interface and sends replies to other interfaces • `reply-only` - the interface will only respond to requests originating from matching IP address/MAC address combinations that are entered as static entries in the IP/ARP table. No dynamic entries will be automatically stored in the IP/ARP table. Therefore, for communications to be successful, a valid static entry must already exist.
arp-timeout (*auto* | *integer*; Default: **auto**)	How long the ARP record is kept in the ARP table after no packets are received from IP address. Value `auto` equals to the value of `arp-timeout` in `ip/settings`, default is 30s.
auto-mac (*yes* | *no*; Default: **yes**)	Automatically select one MAC address of bridge ports as a bridge MAC address, bridge MAC will be chosen from the first added bridge port. After a device reboots, the bridge MAC can change depending on the port-number.
comment (*string*; Default:)	Short description of the interface.
dhcp-snooping (*yes* | *no*; Default: **no**)	Enables or disables DHCP Snooping on the bridge.
disabled (*yes* | *no*; Default: **no**)	Changes whether the bridge is disabled.
ether-type (*0x9100* | *0x8100* | *0x88a8*; Default: **0x8100**)	Changes the EtherType, which will be used to determine if a packet has a VLAN tag. Packets that have a matching EtherType are considered as tagged packets. This property only has an effect when `vlan-filtering` is set to `yes`.
fast-forward (*yes* | *no*; Default: **yes**)	Special and faster case of Fast Path which works only on bridges with 2 interfaces (enabled by default only for new bridges). More details can be found in the Fast Forward section.
forward-delay (*time*; Default: **00:00:15**)	The time which is spent during the initialization phase of the bridge interface (i.e., after router startup or enabling the interface) in the listening/learning state before the bridge will start functioning normally.
frame-types (*admit-all* | *admit-only-untagged-and-priority-tagged* | *admit-only-vlan-tagged*; Default: **admit-all**)	Specifies allowed frame types on a bridge port. This property only has an effect when `vlan-filtering` is set to `yes`.
igmp-snooping (*yes* | *no*; Default: **no**)	Enables multicast group and port learning to prevent multicast traffic from flooding all interfaces in a bridge.
igmp-version (*2* | *3*; Default: **2**)	Selects the IGMP version in which IGMP membership queries will be generated when the bridge interface is acting as an IGMP querier. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.
ingress-filtering (*yes* | *no*; Default: **yes**)	Enables or disables VLAN ingress filtering, which checks if the ingress port is a member of the received VLAN ID in the bridge VLAN table. By default, VLANs that don't exist in the bridge VLAN table are dropped before they are sent out (egress), but this property allows you to drop the packets when they are received (ingress). Should be used with `frame-types` to specify if the ingress traffic should be tagged or untagged. This property only has an effect when `vlan-filtering` is set to `yes`. The setting is enabled by default since RouterOS v7.
l2mtu (*read-only*; Default:)	L2MTU indicates the maximum size of the frame without a MAC header that can be sent by this interface. The L2MTU value will be automatically set by the bridge and it will use the lowest L2MTU value of any associated bridge port. This value cannot be manually changed.
last-member-interval (*time*; Default: **1s**)	When the last client on the bridge port unsubscribes to a multicast group and the bridge is acting as an active querier, the bridge will send group-specific IGMP/MLD query, to make sure that no other client is still subscribed. The setting changes the response time for these queries. In case no membership reports are received in a certain time period (`last-member-interval` * `last-member-query-count`), the multicast group is removed from the multicast database (MDB). If the bridge port is configured with fast-leave, the multicast group is removed right away without sending any queries. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.

last-member-query-count (*integer: 0..4294967295*; Default: **2**)	How many times should `last-member-interval` pass until the IGMP/MLD snooping bridge stops forwarding a certain multicast stream. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.
max-hops (*integer: 6..40*; Default: **20**)	Bridge count which BPDU can pass in an MSTP enabled network in the same region before BPDU is being ignored. This property only has an effect when `protocol-mode` is set to `mstp`.
max-message-age (*time: 6s..40s*; Default: **0 0:00:20**)	Changes the Max Age value in BPDU packets, which is transmitted by the root bridge. A root bridge sends a BPDUs with Max Age set to `max-message-age` value and a Message Age of 0. Every sequential bridge will increment the Message Age before sending their BPDUs. Once a bridge receives a BPDU where Message Age is equal or greater than Max Age, the BPDU is ignored. This property only has an effect when `protocol-mode` is set to `stp` or `rstp`.
membership-interval (*time*; Default: **4m20s**)	The amount of time after an entry in the Multicast Database (MDB) is removed if no IGMP/MLD membership reports are received on a bridge port. This property only has an effect when `igmp-snooping` is set to `yes`.
mld-version (*1 | 2*; Default: **1**)	Selects the MLD version in which MLD membership queries will be generated, when the bridge interface is acting as an MLD querier. This property only has an effect when the bridge has an active IPv6 address, `igmp-snooping` and `multicast-querier` is set to `yes`.
mtu (*integer*; Default: **auto**)	Maximum transmission unit, by default, the bridge will set MTU automatically and it will use the lowest MTU value of any associated bridge port. The default bridge MTU value without any bridge ports added is 1500. The MTU value can be set manually, but it cannot exceed the bridge L2MTU or the lowest bridge port L2MTU. If a new bridge port is added with L2MTU which is smaller than the `actual-mtu` of the bridge (set by the `mtu` property), then manually set value will be ignored and the bridge will act as if `mtu=auto` is set. When adding VLAN interfaces on the bridge, and when VLAN is using higher MTU than default 1500, it is recommended to set manually the MTU of the bridge.
multicast-querier (*yes | no*; Default: **no**)	Multicast querier generates periodic IGMP/MLD general membership queries to which all IGMP/MLD capable devices respond with an IGMP/MLD membership report, usually a PIM (multicast) router or IGMP proxy generates these queries. By using this property you can make an IGMP/MLD snooping enabled bridge to generate IGMP/MLD general membership queries. This property should be used whenever there is no active querier (PIM router or IGMP proxy) in a Layer2 network. Without a multicast querier in a Layer2 network, the Multicast Database (MDB) is not being updated, the learned entries will timeout and IGMP/MLD snooping will not function properly. Only untagged IGMP/MLD general membership queries are generated, IGMP queries are sent with IPv4 0.0.0.0 source address, MLD queries are sent with IPv6 link-local address of the bridge interface. The bridge will not send queries if an external IGMP/MLD querier is detected (see the monitoring values `igmp-querier` and `mld-querier`). This property only has an effect when `igmp-snooping` is set to `yes`.
multicast-router (*disabled | permanent | temporary-query*; Default: **temporary-query**)	A multicast router port is a port where a multicast router or querier is connected. On this port, unregistered multicast streams and IGMP/MLD membership reports will be sent. This setting changes the state of the multicast router for a bridge interface itself. This property can be used to send IGMP/MLD membership reports to the bridge interface for further multicast routing or proxying. This property only has an effect when `igmp-snooping` is set to `yes`. • `disabled` - disabled multicast router state on the bridge interface. Unregistered multicast and IGMP/MLD membership reports are not sent to the bridge interface regardless of what is configured on the bridge interface. • `permanent` - enabled multicast router state on the bridge interface. Unregistered multicast and IGMP/MLD membership reports are sent to the bridge interface itself regardless of what is configured on the bridge interface. • `temporary-query` - automatically detect multicast router state on the bridge interface using IGMP/MLD queries.
name (*text*; Default: **bridgeN**)	Name of the bridge interface.

<table>
<tr><td>port-cost-mode (long | short; Default: long)</td><td>Changes the port path-cost and internal-path-cost mode for bridged ports, utilizing automatic values based on interface speed. This setting does not impact bridged ports with manually configured <code>path-cost</code> or <code>internal-path-cost</code> properties. Below are examples illustrating the path-costs corresponding to specific data rates (with proportionate calculations for intermediate rates):

<table>
<tr><th>Data rate</th><th>Long</th><th>Short</th></tr>
<tr><td>10 Mbps</td><td>2,000,000</td><td>100</td></tr>
<tr><td>100 Mbps</td><td>200,000</td><td>19</td></tr>
<tr><td>1 Gbps</td><td>20,000</td><td>4</td></tr>
<tr><td>10 Gbps</td><td>2,000</td><td>2</td></tr>
<tr><td>25 Gbps</td><td>800</td><td>1</td></tr>
<tr><td>40 Gbps</td><td>500</td><td>1</td></tr>
<tr><td>50 Gbps</td><td>400</td><td>1</td></tr>
<tr><td>100 Gbps</td><td>200</td><td>1</td></tr>
</table>

For bonded interfaces, the highest path-cost among all bonded member ports is applied, this value remains unaffected by the total link speed of the bonding.

For virtual interfaces (such as VLAN, EoIP, VXLAN), as well as wifi, wireless, and 60GHz interfaces, a path-cost of 20,000 is assigned for long mode, and 10 for short mode.

For dynamically bridged interfaces (e.g. wifi, wireless, PPP, VPLS), the path-cost defaults to 20,000 for long mode and 10 for short mode. However, this can be manually overridden by the service that dynamically adds interfaces to bridge, for instance, by using the CAPsMAN <code>datapath.bridge-cost</code> setting.

Use port monitor to observe the applied path-cost.

This property has an effect when <code>protocol-mode</code> is set to <code>stp</code>, <code>rstp</code>, or <code>mstp</code>.</td></tr>
<tr><td>priority (integer: 0..65535 decimal format or 0x0000-0xffff hex format; Default: 32768 / 0x8000)</td><td>Bridge priority, used by R/STP to determine root bridge, used by MSTP to determine CIST and IST regional root bridge. This property has no effect when <code>protocol-mode</code> is set to <code>none</code>.</td></tr>
<tr><td>protocol-mode (none | rstp | stp | mstp; Default: rstp)</td><td>Select Spanning tree protocol (STP) or Rapid spanning tree protocol (RSTP) to ensure a loop-free topology for any bridged LAN. RSTP provides a faster spanning tree convergence after a topology change. Select MSTP to ensure loop-free topology across multiple VLANs. Since RouterOS v6.43 it is possible to forward Reserved MAC addresses that are in the 01:80:C2:00:00:0X range, this can be done by setting the <code>protocol-mode</code> to <code>none</code>.</td></tr>
<tr><td>pvid (integer: 1..4094; Default: 1)</td><td>Port VLAN ID (pvid) specifies which VLAN the untagged ingress traffic is assigned to. It applies e.g. to frames sent from bridge IP and destined to a bridge port. This property only has an effect when <code>vlan-filtering</code> is set to <code>yes</code>.</td></tr>
<tr><td>querier-interval (time; Default: 4m15s)</td><td>Changes the timeout period for detected querier and multicast-router ports. This property only has an effect when <code>igmp-snooping</code> is set to <code>yes</code>.</td></tr>
<tr><td>query-interval (time; Default: 2m5s)</td><td>Changes the interval on how often IGMP/MLD general membership queries are sent out when the bridge interface is acting as an IGMP/MLD querier. The interval takes place when the last startup query is sent. This property only has an effect when <code>igmp-snooping</code> and <code>multicast-querier</code> is set to <code>yes</code>.</td></tr>
<tr><td>query-response-interval (time; Default: 10s)</td><td>The setting changes the response time for general IGMP/MLD queries when the bridge is active as an IGMP/MLD querier. This property only has an effect when <code>igmp-snooping</code> and <code>multicast-querier</code> is set to <code>yes</code>.</td></tr>
<tr><td>region-name (text; Default:)</td><td>MSTP region name. This property only has an effect when <code>protocol-mode</code> is set to <code>mstp</code>.</td></tr>
<tr><td>region-revision (integer: 0..65535; Default: 0)</td><td>MSTP configuration revision number. This property only has an effect when <code>protocol-mode</code> is set to <code>mstp</code>.</td></tr>
</table>

startup-query-count (*integer: 0..4294967295*; Default: **2**)	Specifies how many times general IGMP/MLD queries must be sent when bridge interface is enabled or active querier timeouts. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.
startup-query-interval (*time*; Default: **31s250 ms**)	Specifies the interval between startup general IGMP/MLD queries. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.
transmit-hold-count (*integer: 1..10*; Default: **6**)	The Transmit Hold Count used by the Port Transmit state machine to limit the transmission rate.
vlan-filtering (*yes* | *no*; Default: **no**)	Globally enables or disables VLAN functionality for the bridge.

Changing certain properties can cause the bridge to temporarily disable all ports. This must be taken into account whenever changing such properties on production environments since it can cause all packets to be temporarily dropped. Such properties include `vlan-filtering`, `protocol-mode`, `igmp-snooping`, `fast-forward` and others.

Example

To add and enable a bridge interface that will forward L2 packets:

```
[admin@MikroTik] > interface bridge add
[admin@MikroTik] > interface bridge print
Flags: X - disabled, R - running
0 R name="bridge1" mtu=auto actual-mtu=1500 l2mtu=65535 arp=enabled arp-timeout=auto mac-address=5E:D2:42:95:
56:7F protocol-mode=rstp fast-forward=yes
igmp-snooping=no auto-mac=yes ageing-time=5m priority=0x8000 max-message-age=20s forward-delay=15s transmit-
hold-count=6 vlan-filtering=no
dhcp-snooping=no
```

Bridge Monitoring

To monitor the current status of a bridge interface, use the `monitor` command.

Sub-menu: `/interface bridge monitor`

Property	Description
current-mac-address (*MAC address*)	Current MAC address of the bridge
designated-port-count (*integer*)	Number of designated bridge ports
igmp-querier (*none* | *interface & IPv4 address*)	Shows a bridge port and source IP address from the detected IGMP querier. Only shows detected external IGMP querier, local bridge IGMP querier (including IGMP proxy and PIM) will not be displayed. Monitoring value appears only when `igmp-snooping` is enabled.
mld-querier (*none* | *interface & IPv6 address*)	Shows a bridge port and source IPv6 address from the detected MLD querier. Only shows detected external MLD querier, local bridge MLD querier will not be displayed. Monitoring value appears only when `igmp-snooping` is enabled and the bridge has an active IPv6 address.
multicast-router (*yes* | *no*)	Shows if a multicast router is detected on the port. Monitoring value appears only when `igmp-snooping` is enabled.
port-count (*integer*)	Number of the bridge ports
root-bridge (*yes* | *no*)	Shows whether the bridge is the root bridge of the spanning tree

root-bridge-id (*text*)	The root bridge ID, which is in form of bridge-priority.bridge-MAC-address
root-path-cost (*integer*)	The total cost of the path to the root-bridge
root-port (*name*)	Port to which the root bridge is connected to
state (*enabled / disabled*)	State of the bridge

```
[admin@MikroTik] /interface bridge monitor bridge1
                  state: enabled
    current-mac-address: CC:2D:E0:E4:B3:38
            root-bridge: yes
         root-bridge-id: 0x8000.CC:2D:E0:E4:B3:38
         root-path-cost: 0
              root-port: none
             port-count: 2
  designated-port-count: 2
           fast-forward: no
```

Spanning Tree Protocol

RouterOS bridge interfaces are capable of running Spanning Tree Protocol to ensure a loop-free and redundant topology. For small networks with just 2 bridges STP does not bring many benefits, but for larger networks properly configured STP is very crucial, leaving STP-related values to default may result in a completely unreachable network in case of an even single bridge failure. To achieve a proper loop-free and redundant topology, it is necessary to properly set bridge priorities, port path costs, and port priorities.

In RouterOS it is possible to set any value for bridge priority between 0 and 65535, the IEEE 802.1W standard states that the bridge priority must be in steps of 4096. This can cause incompatibility issues between devices that do not support such values. To avoid compatibility issues, it is recommended to use only these priorities: 0, 4096, 8192, 12288, 16384, 20480, 24576, 28672, 32768, 36864, 40960, 45056, 49152, 53248, 57344, 61440

STP has multiple variants, currently, RouterOS supports STP, RSTP, and MSTP. Depending on needs, either one of them can be used, some devices are able to run some of these protocols using hardware offloading, detailed information about which device support it can be found in the Hardware Offloading section. STP is considered to be outdated and slow, it has been almost entirely replaced in all network topologies by RSTP, which is backward compatible with STP. For network topologies that depend on VLANs, it is recommended to use MSTP since it is a VLAN aware protocol and gives the ability to do load balancing per VLAN groups. There are a lot of considerations that should be made when designing an STP enabled network, more detailed case studies can be found in the Spanning Tree Protocol article. In RouterOS, the `protocol-mode` property controls the used STP variant.

RouterOS bridge does not work with PVST and its variants. The PVST BPDUs (with a MAC destination 01:00:0C:CC:CC:CD) are treated by RouterOS bridges as typical multicast packets. In simpler terms, they undergo RouterOS bridge/switch forwarding logic and may get tagged or untagged.

By the IEEE 802.1ad standard, the BPDUs from bridges that comply with IEEE 802.1Q are not compatible with IEEE 802.1ad bridges, this means that the same bridge VLAN protocol should be used across all bridges in a single Layer2 domain, otherwise (R/M)STP will not function properly.

Per-port STP

There might be certain situations where you want to limit STP functionality on single or multiple ports. Below you can find some examples for different use cases.

Be careful when changing the default (R/M)STP functionality, make sure you understand the working principles of STP and BPDUs. Misconfigured (R/M)STP can cause unexpected behavior.

Create edge ports

Setting a bridge port as an edge port will restrict it from sending BPDUs and will ignore any received BPDUs:

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 edge=yes
add bridge=bridge1 interface=ether2
```

Drop received BPDUs

The bridge filter or NAT rules cannot drop BPDUs when the bridge has STP/RSTP/MSTP enabled due to the special processing of BPDUs. However, dropping received BPDUs on a certain port can be done on some switch chips using ACL rules:

On CRS3xx:

```
/interface ethernet switch rule
add dst-mac-address=01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF new-dst-ports="" ports=ether1 switch=switch1
```

On CRS1xx/CRS2xx with Access Control List (ACL) support:

```
/interface ethernet switch acl
add action=drop mac-dst-address=01:80:C2:00:00:00 src-ports=ether1
```

In this example all received BPDUs on **ether1** are dropped.

If you intend to drop received BPDUs on a port, then make sure to prevent BPDUs from being sent out from the interface that this port is connected to. A root bridge always sends out BPDUs and under normal conditions is waiting for a more superior BPDU (from a bridge with a lower bridge ID), but the bridge must temporarily disable the new root-port when transitioning from a root bridge to a designated bridge. If you have blocked BPDUs only on one side, then a port will flap continuously.

Enable BPDU guard

In this example, if **ether1** receives a BPDU, it will block the port and will require you to manually re-enable it.

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 bpdu-guard=yes
add bridge=bridge1 interface=ether2
```

Enable Root guard

In this example, **ether1** is configured with `restricted-role=yes.` It prevented the port from becoming the root port for the CIST or any MSTI, regardless of its best spanning tree priority vector. Such a port will be selected as an Alternate Port (discarding state) and remains so as long as it continues to receive superior BPDUs. It will automatically transition to the forwarding state when it no longer detects a superior root path. Network administrators may enable this setting to safeguard against external bridges influencing the active spanning tree.

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 restricted-role=yes
add bridge=bridge1 interface=ether2

[admin@MikroTik] /interface/bridge/port monitor [find]
                  interface: ether1                   ether2
                     status: in-bridge                in-bridge
                port-number: 1                        2
                       role: alternate-port           designated-port
                  edge-port: no                       yes
        edge-port-discovery: yes                      yes
        point-to-point-port: yes                      yes
               external-fdb: no                       no
               sending-rstp: yes                      yes
                   learning: no                       yes
                 forwarding: no                       yes
           actual-path-cost: 20000                    20000
             root-path-cost: 20000
          designated-bridge: 0x7000.64:D1:54:C7:3A:6E
            designated-cost: 0
    designated-port-number: 1
          hw-offload-group: switch1                  switch1
```

Bridge Settings

Under the bridge settings menu, it is possible to control certain features for all bridge interfaces and monitor global bridge counters.

Sub-menu: `/interface bridge settings`

Property	Description
use-ip-firewall (*yes* | *no*; Default: **no**)	Force bridged traffic to also be processed by prerouting, forward, and postrouting sections of IP routing (see more details on Packet Flow article). This does not apply to routed traffic. This property is required in case you want to assign Simple Queues or global Queue Tree to traffic in a bridge. Property `use-ip-firewall-for-vlan` is required in case bridge `vlan-filtering` is used.
use-ip-firewall-for-pppoe (*yes* | *no*; Default: **no**)	Send bridged un-encrypted PPPoE traffic to also be processed by IP/Firewall. This property only has an effect when `use-ip-firewall` is set to `yes`. This property is required in case you want to assign Simple Queues or global Queue Tree to PPPoE traffic in a bridge.
use-ip-firewall-for-vlan (*yes* | *no*; Default: **no**)	Send bridged VLAN traffic to also be processed by IP/Firewall. This property only has an effect when `use-ip-firewall` is set to `yes`. This property is required in case you want to assign Simple Queues or global Queue Tree to VLAN traffic in a bridge.
allow-fast-path (*yes* | *no*; Default: **yes**)	Whether to enable a bridge Fast Path globally.
bridge-fast-path-active (*yes* | *no*; Default: *)*	Shows whether a bridge FastPath is active globally, Fast Path status per bridge interface is not displayed.
bridge-fast-path-packets (*integer*; Default: *)*	Shows packet count forwarded by bridge Fast Path.

bridge-fast-path-bytes (*integer*; Default: *)*	Shows byte count forwarded by bridge Fast Path.
bridge-fast-forward-packets (*integer*; Default: *)*	Shows packet count forwarded by bridge Fast Forward.
bridge-fast-forward-bytes (*integer*; Default: *)*	Shows byte count forwarded by bridge Fast Forward.

⚠ In case you want to assign Simple Queues or global Queue Trees to traffic that is being forwarded by a bridge, then you need to enable the `use-ip-firewall` property. Without using this property the bridge traffic will never reach the postrouting chain, Simple Queues and global Queue Trees are working in the postrouting chain. To assign Simple Queues or global Queue Trees for VLAN or PPPoE traffic in a bridge you should enable appropriate properties as well.

Port Settings

Port submenu is used to add interfaces in a particular bridge.

Sub-menu: `/interface bridge port`

Property	Description
auto-isolate (*yes* | *no*; Default: **no**)	When enabled, prevents a port moving from discarding into forwarding state if no BPDUs are received from the neighboring bridge. The port will change into a forwarding state only when a BPDU is received. This property only has an effect when `protocol-mode` is set to `rstp` or `mstp` and `edge` is set to `no`.
bpdu-guard (*yes* | *no*; Default: **no**)	Enables or disables BPDU Guard feature on a port. This feature puts the port in a disabled role if it receives a BPDU and requires the port to be manually disabled and enabled if a BPDU was received. Should be used to prevent a bridge from BPDU related attacks. This property has no effect when `protocol-mode` is set to `none`.
bridge (*name*; Default: **none**)	The bridge interface where the respective interface is grouped in.
broadcast-flood (*yes* | *no*; Default: **yes**)	When enabled, bridge floods broadcast traffic to all bridge egress ports. When disabled, drops broadcast traffic on egress ports. Can be used to filter all broadcast traffic on an egress port. Broadcast traffic is considered as traffic that uses **FF:FF:FF:FF:FF:FF** as destination MAC address, such traffic is crucial for many protocols such as DHCP, ARP, NDP, BOOTP (Netinstall), and others. This option does not limit traffic flood to the CPU.
edge (*auto* | *no* | *no-discover* | *yes* | *yes-discover*; Default: **auto**)	Set port as edge port or non-edge port, or enable edge discovery. Edge ports are connected to a LAN that has no other bridges attached. An edge port will skip the learning and the listening states in STP and will transition directly to the forwarding state, this reduces the STP initialization time. If the port is configured to discover edge port then as soon as the bridge detects a BPDU coming to an edge port, the port becomes a non-edge port. This property has no effect when `protocol-mode` is set to `none`. • `no` - non-edge port, will participate in learning and listening states in STP. • `no-discover` - non-edge port with enabled discovery, will participate in learning and listening states in STP, a port can become an edge port if no BPDU is received. • `yes` - edge port without discovery, will transit directly to forwarding state. • `yes-discover` - edge port with enabled discovery, will transit directly to forwarding state. • `auto` - same as `no-discover`, but will additionally detect if a bridge port is a Wireless interface with disabled bridge-mode, such interface will be automatically set as an edge port without discovery.
fast-leave (*yes* | *no*; Default: **no**)	Enables IGMP/MLD fast leave feature on the bridge port. The bridge will stop forwarding multicast traffic to a bridge port when an IGMP/MLD leave message is received. This property only has an effect when `igmp-snooping` is set to `yes`.

frame-types (*admit-all | admit-only-untagged-and-priority-tagged | admit-only-vlan-tagged*; Default: **admit-all**)	Specifies allowed ingress frame types on a bridge port. This property only has an effect when `vlan-filtering` is set to `yes`.
ingress-filtering (*yes | no*; Default: **yes**)	Enables or disables VLAN ingress filtering, which checks if the ingress port is a member of the received VLAN ID in the bridge VLAN table. Should be used with `frame-types` to specify if the ingress traffic should be tagged or untagged. This property only has effect when `vlan-filtering` is set to `yes`. The setting is enabled by default since RouterOS v7.
learn (*auto | no | yes*; Default: **auto**)	Changes MAC learning behavior on a bridge port • `yes` - enables MAC learning • `no` - disables MAC learning • `auto` - detects if bridge port is a Wireless interface and uses a Wireless registration table instead of MAC learning, will use Wireless registration table if the Wireless interface is set to one of `ap-bridge`, `bridge`, `wds-slave` mode and bridge mode for the Wireless interface is disabled.
multicast-router (*disabled | permanent | temporary-query*; Default: **temporary-query**)	A multicast router port is a port where a multicast router or querier is connected. On this port, unregistered multicast streams and IGMP/MLD membership reports will be sent. This setting changes the state of the multicast router for bridge ports. This property can be used to send IGMP/MLD membership reports to certain bridge ports for further multicast routing or proxying. This property only has an effect when `igmp-snooping` is set to `yes`. • `disabled` - disabled multicast router state on the bridge port. Unregistered multicast and IGMP/MLD membership reports are not sent to the bridge port regardless of what is connected to it. • `permanent` - enabled multicast router state on the bridge port. Unregistered multicast and IGMP/MLD membership reports are sent to the bridge port regardless of what is connected to it. • `temporary-query` - automatically detect multicast router state on the bridge port using IGMP/MLD queries.
horizon (*integer 0..429496729*; Default: **none**)	Use split horizon bridging to prevent bridging loops. Set the same value for a group of ports, to prevent them from sending data to ports with the same horizon value. Split horizon is a software feature that disables hardware offloading. Read more about Bridge split horizon.
hw (*yes | no*; Default: **yes**)	Allows to enable or disable hardware offloading on interfaces capable of HW offloading. For software interfaces like EoIP or VLAN this setting is ignored and has no effect. Certain bridge or port functions can automatically disable HW offloading, use the `print` command to see whether the "H" flag is active.
internal-path-cost (*integer: 1..200000000*; Default:)	Path cost to the interface for MSTI0 inside a region. If not manually configured, the bridge automatically determines the internal-path-cost based on the interface speed and the `port-cost-mode` setting. To revert to the automatic determination and remove any manually applied value, simply use an exclamation mark before the `internal-path-cost` property. This property only has effect when `protocol-mode` is set to `mstp`. `/interface bridge port set [find interface=sfp28-1] !internal-path-cost` Use port monitor to observe the applied internal-path-cost.
interface (*name*; Default: **none**)	Name of the interface.
path-cost (*integer: 1..200000000*; Default:)	Path cost to the interface, used by STP and RSTP to determine the best path, and used by MSTP to determine the best path between regions. If not manually configured, the bridge automatically determines the path-cost based on the interface speed and the `port-cost-mode` setting. To revert to the automatic determination and remove any manually applied value, simply use an exclamation mark before the `path-cost` property. This property has no effect when `protocol-mode` is set to `none`. `/interface bridge port set [find interface=sfp28-1] !path-cost` Use port monitor to observe the applied path-cost.

point-to-point (*auto* | *yes* | *no*; Default: **auto**)	Specifies if a bridge port is connected to a bridge using a point-to-point link for faster convergence in case of failure. By setting this property to `yes`, you are forcing the link to be a point-to-point link, which will skip the checking mechanism, which detects and waits for BPDUs from other devices from this single link. By setting this property to `no`, you are expecting that a link can receive BPDUs from multiple devices. By setting the property to `yes`, you are significantly improving (R/M)STP convergence time. In general, you should only set this property to `no` if it is possible that another device can be connected between a link, this is mostly relevant to Wireless mediums and Ethernet hubs. If the Ethernet link is full-duplex, `auto` enables point-to-point functionality. This property has no effect when `protocol-mode` is set to `none`.
priority (*integer: 0..240*; Default: **128**)	The priority of the interface, used by STP to determine the root port, used by MSTP to determine root port between regions.
pvid (*integer 1..4094*; Default: **1**)	Port VLAN ID (pvid) specifies which VLAN the untagged ingress traffic is assigned to. This property only has an effect when `vlan-filtering` is set to `yes`.
restricted-role (*yes* | *no*; Default: **no**)	Enables or disables the restricted role on a port. When enabled, it prevents the port from becoming the root port for the CIST or any MSTI, regardless of its best spanning tree priority vector. Such a port will be selected as an Alternate Port (discarding state) and remains so as long as it continues to receive superior BPDUs. It will automatically transition to the forwarding state when it no longer detects a superior root path. Network administrators may enable this setting to safeguard against external bridges influencing the active spanning tree, a feature also known as root-guard or root-protection. This property has an effect when `protocol-mode` is set to `stp`, `rstp`, or `mstp` (support for STP and RSTP is available since RouterOS v7.14).
restricted-tcn (*yes* | *no*; Default: **no**)	Enables or disables topology change notification (TCN) handling on a port. When enabled, it causes the port not to propagate received topology change notifications to other ports, and any changes caused by the port itself does not result in topology change notification to other ports. This parameter is disabled by default. It can be set by a network administrator to prevent external bridges causing MAC address flushing in local network. This property has an effect when `protocol-mode` is set to `stp`, `rstp`, or `mstp` (support for STP and RSTP is available since RouterOS v7.14).
tag-stacking (*yes* | *no*; Default: **no**)	Forces all packets to be treated as untagged packets. Packets on ingress port will be tagged with another VLAN tag regardless if a VLAN tag already exists, packets will be tagged with a VLAN ID that matches the `pvid` value and will use EtherType that is specified in `ether-type`. This property only has effect when `vlan-filtering` is set to `yes`.
trusted (*yes* | *no*; Default: **no**)	When enabled, it allows forwarding DHCP packets towards the DHCP server through this port. Mainly used to limit unauthorized servers to provide malicious information for users. This property only has an effect when `dhcp-snooping` is set to `yes`.
unknown-multicast-flood (*yes* | *no*; Default: **yes**)	Changes the multicast flood option on bridge port, only controls the egress traffic. When enabled, the bridge allows flooding multicast packets to the specified bridge port, but when disabled, the bridge restricts multicast traffic from being flooded to the specified bridge port. The setting affects all multicast traffic, this includes non-IP, IPv4, IPv6 and the link-local multicast ranges (e. g. 224.0.0.0/24 and ff02::1). Note that when `igmp-snooping` is enabled and IGMP/MLD querier is detected, the bridge will automatically restrict unknown IP multicast from being flooded, so the setting is not mandatory for IGMP/MLD snooping setups. When using this setting together with `igmp-snooping`, the only multicast traffic that is allowed on the bridge port is the known multicast from the MDB table.
unknown-unicast-flood (*yes* | *no*; Default: **yes**)	Changes the unknown unicast flood option on bridge port, only controls the egress traffic. When enabled, the bridge allows flooding unknown unicast packets to the specified bridge port, but when disabled, the bridge restricts unknown unicast traffic from being flooded to the specified bridge port. If a MAC address is not learned in the host table, then the traffic is considered as unknown unicast traffic and will be flooded to all ports. MAC address is learned as soon as a packet on a bridge port is received and the source MAC address is added to the bridge host table. Since it is required for the bridge to receive at least one packet on the bridge port to learn the MAC address, it is recommended to use static bridge host entries to avoid packets being dropped until the MAC address has been learned.

RouterOS can handle a maximum of 1024 bridged interfaces, this limit is fixed and cannot be modified. If you try to add more interfaces as bridge ports, it may lead to unpredictable behavior.

Example

To group ether1 and ether2 in the already created bridge1 interface.

```
[admin@MikroTik] /interface bridge port add bridge=bridge1 interface=ether1
[admin@MikroTik] /interface bridge port add bridge=bridge1 interface=ether2
[admin@MikroTik] /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #     INTERFACE       BRIDGE       HW  PVID PRIORITY  PATH-COST INTERNAL-PATH-COST    HORIZON
 0     ether1          bridge1      yes  100     0x80         10                 10       none
 1     ether2          bridge1      yes  200     0x80         10                 10       none
```

Interface lists

Starting with RouterOS v6.41 it possible to add interface lists as a bridge port and sort them. Interface lists are useful for creating simpler firewall rules. Below is an example how to add an interface list to a bridge:

```
/interface list
add name=LAN1
add name=LAN2
/interface list member
add interface=ether1 list=LAN1
add interface=ether2 list=LAN1
add interface=ether3 list=LAN2
add interface=ether4 list=LAN2
/interface bridge port
add bridge=bridge1 interface=LAN1
add bridge=bridge1 interface=LAN2
```

Ports from an interface list added to a bridge will show up as dynamic ports:

```
[admin@MikroTik] /interface bridge port> pr
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #     INTERFACE       BRIDGE       HW  PVID PRIORITY  PATH-COST INTERNAL-PATH-COST    HORIZON
 0     LAN1            bridge1      yes    1     0x80         10                 10       none
 1  D  ether1          bridge1      yes    1     0x80         10                 10       none
 2  D  ether2          bridge1      yes    1     0x80         10                 10       none
 3     LAN2            bridge1      yes    1     0x80         10                 10       none
 4  D  ether3          bridge1      yes    1     0x80         10                 10       none
 5  D  ether4          bridge1      yes    1     0x80         10                 10       none
```

It is also possible to sort the order of lists in which they appear. This can be done using the `move` command. Below is an example of how to sort interface lists:

```
[admin@MikroTik] > /interface bridge port move 3 0
[admin@MikroTik] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #     INTERFACE       BRIDGE       HW  PVID PRIORITY  PATH-COST INTERNAL-PATH-COST    HORIZON
 0     LAN2            bridge1      yes    1     0x80         10                 10       none
 1  D  ether3          bridge1      yes    1     0x80         10                 10       none
 2  D  ether4          bridge1      yes    1     0x80         10                 10       none
 3     LAN1            bridge1      yes    1     0x80         10                 10       none
 4  D  ether1          bridge1      yes    1     0x80         10                 10       none
 5  D  ether2          bridge1      yes    1     0x80         10                 10       none
```


The second parameter when moving interface lists is considered as "before id", the second parameter specifies before which interface list should be the selected interface list moved. When moving the first interface list in place of the second interface list, then the command will have no effect since the first list will be moved before the second list, which is the current state either way.

Bridge Port Monitoring

To monitor the current status of bridge ports, use the `monitor` command.

Sub-menu: `/interface bridge port monitor`

Property	Description
designated-bridge (*bridge identifier*)	Shows the received bridge identifier.
designated-cost (*integer*)	Shows the received root-path-cost.
designated-port-number (*integer*)	Shows the received port number.
edge-port (*yes* | *no*)	Whether the port is an edge port or not.
edge-port-discovery (*yes* | *no*)	Whether the port is set to automatically detect edge ports.
external-fdb (*yes* | *no*)	Whether the registration table is used instead of a forwarding database.
forwarding (*yes* | *no*)	Shows if the port is not blocked by (R/M)STP.
hw-offload-group (*switchX*)	Switch chip used by the port.
interface (*name*)	Interface name.
learning (*yes* | *no*)	Shows whether the port is capable of learning MAC addresses.
multicast-router (*yes* | *no*)	Shows if a multicast router is detected on the port. Monitoring value appears only when `igmp-snooping` is enabled.
path-cost (*integer: 1.. 200000000*)	Shows the actual port path-cost. Either manually applied or automatically determined based on the interface speed and the `port-cost-mode` setting.
port-number (*integer 1..4095*)	A port-number will be assigned in the order that ports got added to the bridge, but this is only true until reboot. After reboot, the internal port numbering will be used.
point-to-point-port (*yes* | *no*)	Whether the port is connected to a bridge port using full-duplex (yes) or half-duplex (no).
role (*designated* | *root port* | *alternate* | *backup* | *disabled*)	(R/M)STP algorithm assigned the role of the port: • `disabled-port` - not strictly part of STP, a network administrator can manually disable a port • `root-port` - a forwarding port that is the best port facing towards the root bridge • `alternative-port` - an alternate path to the root bridge • `designated-port` - a forwarding port for every LAN segment • `backup-port` - a backup/redundant path to a segment where another bridge port already connects.
root-path-cost (*integer*)	The total cost of the path to the root-bridge
sending-rstp (*yes* | *no*)	Whether the port is using RSTP or MSTP BPDU types. A port will transit to STP type when RSTP/MSTP enabled port receives an STP BPDU. This settings **does not** indicate whether the BDPUs are actually sent.
status (*in-bridge* | *inactive*)	Port status: • `in-bridge` - port is enabled • `inactive` - port is disabled.

```
[admin@MikroTik] /interface bridge port monitor [find interface=sfp-sfpplus2]
              interface: sfp-sfpplus2
                 status: in-bridge
            port-number: 1
                   role: root-port
              edge-port: no
    edge-port-discovery: yes
    point-to-point-port: yes
           external-fdb: no
           sending-rstp: yes
               learning: yes
             forwarding: yes
              path-cost: 2000
         root-path-cost: 4000
      designated-bridge: 0x8000.DC:2C:6E:9E:11:1C
        designated-cost: 2000
  designated-port-number: 2
```

Hosts Table

MAC addresses that have been learned on a bridge interface can be viewed in the host menu. Below is a table of parameters and flags that can be viewed.

Sub-menu: `/interface bridge host`

Property	Description
bridge (*read-only: name*)	The bridge the entry belongs to
disabled (*read-only: flag*)	Whether the static host entry is disabled
dynamic (*read-only: flag*)	Whether the host has been dynamically created
external (*read-only: flag*)	Whether the host has been learned using an external table, for example, from a switch chip or Wireless registration table. Adding a static host entry on a hardware-offloaded bridge port will also display an active external flag
invalid (*read-only: flag*)	Whether the host entry is invalid, can appear for statically configured hosts on already removed interface
local (*read-only: flag*)	Whether the host entry is created from the bridge itself (that way all local interfaces are shown)
mac-address (*read-only: MAC address*)	Host's MAC address
on-interface (*read-only: name*)	Which of the bridged interfaces the host is connected to

Monitoring

To get the active hosts table:

```
[admin@MikroTik] /interface bridge host print
Flags: X - disabled, I - invalid, D - dynamic, L - local, E - external
 #       MAC-ADDRESS        VID ON-INTERFACE           BRIDGE
 0   D   B8:69:F4:C9:EE:D7      ether1                 bridge1
 1   D   B8:69:F4:C9:EE:D8      ether2                 bridge1
 2   DL  CC:2D:E0:E4:B3:38      bridge1                bridge1
 3   DL  CC:2D:E0:E4:B3:39      ether2                 bridge1
```

Static entries

It is possible to add a static MAC address entry into the host table. This can be used to forward a certain type of traffic through a specific port. Another use case for static host entries is to protect the device resources by disabling dynamic learning and relying only on configured static host entries. Below is a table of possible parameters that can be set when adding a static MAC address entry into the host table.

Sub-menu: `/interface bridge host`

Property	Description
bridge (*name*; Default: **none**)	The bridge interface to which the MAC address is going to be assigned.
disabled (*yes* | *no*; Default: **no**)	Disables/enables static MAC address entry.
interface (*name*; Default: **none**)	Name of the interface.
mac-address (*MAC address*; Default:)	MAC address that will be added to the host table statically.
vid (*integer: 1..4094*; Default:)	VLAN ID for the statically added MAC address entry.

For example, if it was required that all traffic destined to **4C:5E:0C:4D:12:43** is forwarded only through **ether2**, then the following commands can be used:

```
/interface bridge host
add bridge=bridge interface=ether2 mac-address=4C:5E:0C:4D:12:43
```

Multicast Table

When IGMP/MLD snooping is enabled, the bridge will start to listen to IGMP/MLD communication, create multicast database (MDB) entries, and make forwarding decisions based on the received information. Packets with link-local multicast destination addresses 224.0.0.0/24 and ff02::1 are not restricted and are always flooded on all ports and VLANs. To see learned multicast database entries, use the `print` command.

Sub-menu: `/interface bridge mdb`

Property	Description
bridge (*read-only: name*)	Shows the bridge interface the entry belongs to.
group (*read-only: ipv4 | ipv6 address*)	Shows a multicast group address.
on-ports (*read-only: name*)	Shows the bridge ports which are subscribed to the certain multicast group.
vid (*read-only: integer*)	Shows the VLAN ID for the multicast group, only applies when `vlan-filtering` is enabled.

```
[admin@MikroTik] /interface bridge mdb print
Flags: D - DYNAMIC
Columns: GROUP, VID, ON-PORTS, BRIDGE
 #   GROUP              VID  ON-PORTS  BRIDGE
 0 D ff02::2              1  bridge1   bridge1
 1 D ff02::6a             1  bridge1   bridge1
 2 D ff02::1:ff00:0       1  bridge1   bridge1
 3 D ff02::1:ff01:6a43    1  bridge1   bridge1
 4 D 229.1.1.1           10  ether2    bridge1
 5 D 229.2.2.2           10  ether3    bridge1
                             ether2
 6 D ff02::2             10  ether5    bridge1
                             ether3
                             ether2
                             ether4
```

Static entries

Since RouterOS version 7.7, it is possible to create static MDB entries for IPv4 and IPv6 multicast groups.

Sub-menu: `/interface bridge mdb`

Property	Description
bridge (*name*; Default:)	The bridge interface to which the MDB entry is going to be assigned.
disabled (*yes* | *no* ; Default: **no**)	Disables or enables static MDB entry.
group (*ipv4* | *ipv6 address*; Default:)	The IPv4 or IPv6 multicast address. Static entries for link-local multicast groups 224.0.0.0/24 and ff02::1 cannot be created, as these packets are always flooded on all ports and VLANs.
ports (*name*; Default:)	The list of bridge ports to which the multicast group will be forwarded.
vid (*integer: 1.. 4094*; Default:)	The VLAN ID on which the MDB entry will be created, only applies when `vlan-filtering` is enabled. When VLAN ID is not specified, the entry will work in shared-VLAN mode and dynamically apply on all defined VLAN IDs for particular ports.

For example, to create a static MDB entry for multicast group 229.10.10.10 on ports ether2 and ether3 on VLAN 10, use the command below:

```
/interface bridge mdb
add bridge=bridge1 group=229.10.10.10 ports=ether2,ether3 vid=10
```

Verify the results with the `print` command:

```
[admin@MikroTik] > /interface bridge mdb print where group=229.10.10.10
Columns: GROUP, VID, ON-PORTS, BRIDGE
 # GROUP         VID  ON-PORTS  BRIDGE
12 229.10.10.10   10  ether2    bridge1
                      ether3
```

In case a certain IPv6 multicast group does not need to be snooped and it is desired to be flooded on all ports and VLANs, it is possible to create a static MDB entry on all VLANs and ports, including the bridge interface itself. Use the command below to create a static MDB entry for multicast group ff02::2 on all VLANs and ports (modify the `ports` setting for your particular setup):

```
/interface bridge mdb
add bridge=bridge1 group=ff02::2 ports=bridge1,ether2,ether3,ether4,ether5

[admin@MikroTik] > /interface bridge mdb print where group=ff02::2
Flags: D - DYNAMIC
Columns: GROUP, VID, ON-PORTS, BRIDGE
 #   GROUP    VID  ON-PORTS  BRIDGE
 0   ff02::2                bridge1
15 D ff02::2    1  bridge1   bridge1
16 D ff02::2   10  bridge1   bridge1
                   ether2
                   ether3
                   ether4
                   ether5
17 D ff02::2   20  bridge1   bridge1
                   ether2
                   ether3
18 D ff02::2   30  bridge1   bridge1
                   ether2
                   ether3
```

Bridge Hardware Offloading

It is possible to switch multiple ports together if a device has a built-in switch chip. While a bridge is a software feature that will consume CPU's resources, the bridge hardware offloading feature will allow you to use the built-in switch chip to forward packets. This allows you to achieve higher throughput if configured correctly.

In previous versions (prior to RouterOS v6.41) you had to use the master-port property to switch multiple ports together, but in RouterOS v6.41 this property is replaced with the bridge hardware offloading feature, which allows your to switch ports and use some of the bridge features, for example, Spanning Tree Protocol.

> ⚠ When upgrading from previous versions (before RouterOS v6.41), the old master-port configuration is automatically converted to the new **Bridge Hardware Offloading** configuration. When downgrading from newer versions (RouterOS v6.41 and newer) to older versions (before RouterOS v6.41) the configuration is not converted back, a bridge without hardware offloading will exist instead, in such a case you need to reconfigure your device to use the old master-port configuration.

Below is a list of devices and feature that supports hardware offloading (+) or disables hardware offloading (-):

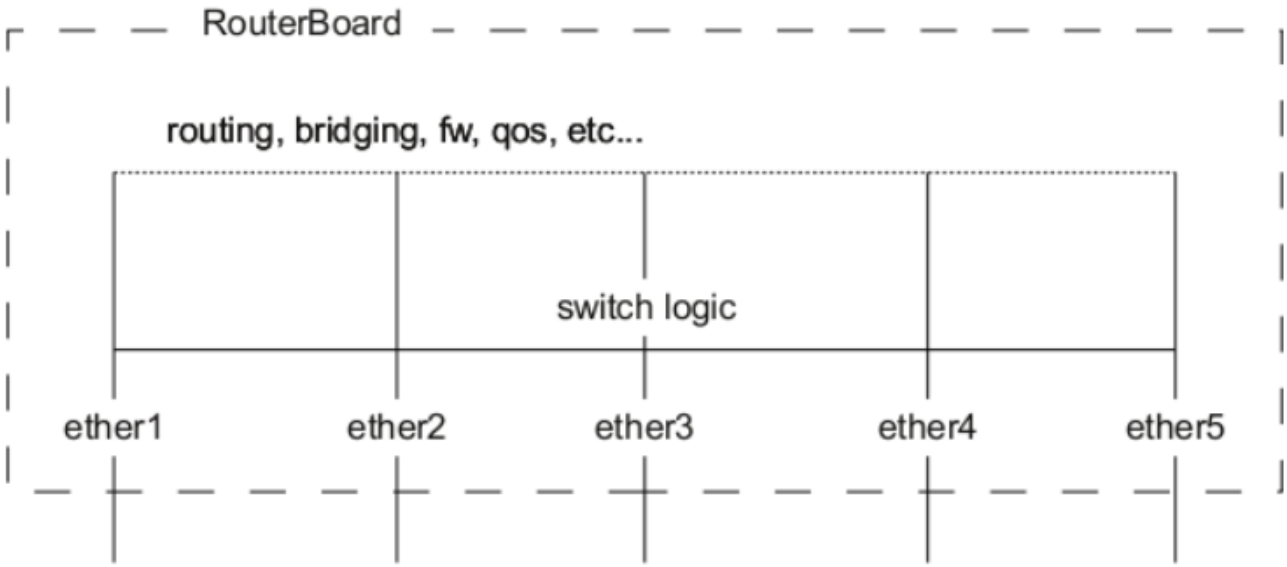
RouterBoard/[Switch Chip] Model	Features in Switch menu	Bridge STP /RSTP	Bridge MSTP	Bridge IGMP Snooping	Bridge DHCP Snooping	Bridge VLAN Filtering	Bonding [4, 5]	Horizon [4]
CRS3xx, CRS5xx series	+	+	+	+	+	+	+	-
CCR2116, CCR2216	+	+	+	+	+	+	+	-
CRS1xx/CRS2xx series	+	+	-	+[2]	+[1]	-	-	-
[QCA8337]	+	+	-	-	+[2]	-	-	-
[Atheros8327]	+	+	-	-	+[2]	-	-	-
[Atheros8316]	+	+	-	-	+[2]	-	-	-
[Atheros8227]	+	+	-	-	-	-	-	-
[Atheros7240]	+	+	-	-	-	-	-	-
[IPQ-PPE]	+[6]	-	-	-	-	-	-	-
[ICPlus175D]	+	-	-	-	-	-	-	-
[MT7621, MT7531]	+	+[3]	+[3]	-	-	+[3]	-	-
[RTL8367]	+	+[3]	+[3]	-	-	+[3]	-	-
[88E6393X, 88E6191X, 88E6190]	+	+	+	+	+	+[3]	+[7]	-

Footnotes:

1. The feature will not work properly in VLAN switching setups. It is possible to correctly snoop DHCP packets only for a single VLAN, but this requires that these DHCP messages get tagged with the correct VLAN tag using an ACL rule, for example, `/interface ethernet switch acl add dst-l3-port=67-68 ip-protocol=udp mac-protocol=ip new-customer-vid=10 src-ports=switch1-cpu`. DHCP Option 82 will not contain any information regarding VLAN-ID.
2. The feature will not work properly in VLAN switching setups.
3. The HW vlan-filtering and R/M/STP was added in the RouterOS 7.1rc1 (for RTL8367) and 7.1rc5 (for MT7621) versions. The switch does not support other ether-type 0x88a8 or 0x9100 (only 0x8100 is supported) and no tag-stacking. Using these features will disable HW offload.
4. The HW offloading will be disabled only for the specific bridge port, not the entire bridge.
5. Only `802.3ad` and `balance-xor` modes can be HW offloaded. Other bonding modes do not support HW offloading.
6. Currently, HW offloaded bridge support for the IPQ-PPE switch chip is still a work in progress. We recommend using, the default, non-HW offloaded bridge (enabled RSTP).
7. The `802.3ad` mode is compatible only with R/M/STP enabled bridge.

> ⚠ When upgrading from older versions (before RouterOS v6.41), only the master-port configuration is converted. For each master-port a bridge will be created. VLAN configuration is not converted and should not be changed, check the Basic VLAN switching guide to be sure how VLAN switching should be configured for your device.

Bridge Hardware Offloading should be considered as port switching, but with more possible features. By enabling hardware offloading you are allowing a built-in switch chip to process packets using its switching logic. The diagram below illustrates that switching occurs before any software related action.



A packet that is received by one of the ports always passes through the switch logic first. Switch logic decides which ports the packet should be going to (most commonly this decision is made based on the destination MAC address of a packet, but there might be other criteria that might be involved based on the packet and the configuration). In most cases the packet will not be visible to RouterOS (only statistics will show that a packet has passed through), this is because the packet was already processed by the switch chip and never reached the CPU.

Though it is possible in certain situations to allow a packet to be processed by the CPU, this is usually called a packet forwarding to the switch CPU port (or the bridge interface in bridge VLAN filtering scenario). This allows the CPU to process the packet and lets the CPU to forward the packet. Passing the packet to the CPU port will give you the opportunity to route packets to different networks, perform traffic control and other software related packet processing actions. To allow a packet to be processed by the CPU, you need to make certain configuration changes depending on your needs and on the device you are using (most commonly passing packets to the CPU are required for VLAN filtering setups). Check the manual page for your specific device:

- CRS1xx/2xx series switches
- CRS3xx, CRS5xx series switches and CCR2116, CCR2216 routers
- non-CRS series switches

Certain bridge and Ethernet port properties are directly related to switch chip settings. Changing such properties can trigger a **switch chip reset**, temporarily disabling all Ethernet ports that are on the switch chip for the settings to take effect. This must be taken into account whenever changing properties in production environments. Such properties include DHCP Snooping, IGMP Snooping, VLAN filtering, L2MTU, Flow Control, and others. The exact settings that can trigger a switch chip reset depend on the device's model.

The CRS1xx/2xx series switches support multiple hardware offloaded bridges per switch chip. All other devices support only one hardware offloaded bridge per switch chip. Use the hw=yes/no parameter to select which bridge will use hardware offloading.

Example

Port switching with bridge configuration and enabled hardware offloading:

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
add bridge=bridge1 interface=ether4 hw=yes
add bridge=bridge1 interface=ether5 hw=yes
```

Make sure that hardware offloading is enabled and active by checking the "H" flag:

```
[admin@MikroTik] /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #     INTERFACE       BRIDGE       HW  PVID PRIORITY  PATH-COST INTERNAL-PATH-COST    HORIZON
 0   H ether2          bridge1      yes    1     0x80         10                 10       none
 1   H ether3          bridge1      yes    1     0x80         10                 10       none
 2   H ether4          bridge1      yes    1     0x80         10                 10       none
 3   H ether5          bridge1      yes    1     0x80         10                 10       none
```

Port switching in RouterOS v6.41 and newer is done using the bridge configuration. Prior to RouterOS v6.41 port switching was done using the master-port property.

Bridge VLAN Filtering

Bridge VLAN Filtering provides VLAN-aware Layer 2 forwarding and VLAN tag modifications within the bridge. This set of features makes bridge operation more similar to a traditional Ethernet switch and allows overcoming Spanning Tree compatibility issues compared to the configuration when VLAN interfaces are bridged. Bridge VLAN Filtering configuration is highly recommended to comply with STP (IEEE 802.1D), RSTP (IEEE 802.1W) standards and is mandatory to enable MSTP (IEEE 802.1s) support in RouterOS.

The main VLAN setting is `vlan-filtering` which globally controls VLAN-awareness and VLAN tag processing in the bridge. If `vlan-filtering=no` is configured, the bridge ignores VLAN tags, works in a shared-VLAN-learning (SVL) mode, and cannot modify VLAN tags of packets. Turning on `vlan-filtering` enables all bridge VLAN related functionality and independent-VLAN-learning (IVL) mode. Besides joining the ports for Layer2 forwarding, the bridge itself is also an interface therefore it has Port VLAN ID (pvid).

Currently, CRS3xx, CRS5xx series switches, CCR2116, CCR2216 routers and RTL8367, 88E6393X, 88E6191X, 88E6190, MT7621 and MT7531 switch chips (since RouterOS v7) are capable of using bridge VLAN filtering and hardware offloading at the same time, other devices will not be able to use the benefits of a built-in switch chip when bridge VLAN filtering is enabled. Other devices should be configured according to the method described in the Basic VLAN switching guide. If an improper configuration method is used, your device can cause throughput issues in your network.

Bridge VLAN table

Bridge VLAN table represents per-VLAN port mapping with an egress VLAN tag action. The `tagged` ports send out frames with a corresponding VLAN ID tag. The `untagged` ports remove a VLAN tag before sending out frames. Bridge ports with `frame-types` set to `admit-all` or `admit-only-untagged-and-priority-tagged` will be automatically added as untagged ports for the `pvid` VLAN.

Sub-menu: `/interface bridge vlan`

Property	Description
bridge (*name*; Default: **none**)	The bridge interface which the respective VLAN entry is intended for.
disabled (*yes* | *no*; Default: **no**)	Enables or disables Bridge VLAN entry.
tagged (*interfaces*; Default: **none**)	Interface list with a VLAN tag adding action in egress. This setting accepts comma-separated values. e.g. `tagged=ether1,ether2`.
untagged (*interfaces*; Default: **none**)	Interface list with a VLAN tag removing action in egress. This setting accepts comma-separated values. e.g. `untagged=ether3,ether4`
vlan-ids (*integer 1..4094*; Default: **1**)	The list of VLAN IDs for certain port configuration. This setting accepts the VLAN ID range as well as comma-separated values. e.g. `vlan-ids=100-115,120,122,128-130`.

The `vlan-ids` parameter can be used to specify a set or range of VLANs, but specifying multiple VLANs in a single bridge VLAN table entry should only be used for ports that are tagged ports. In case multiple VLANs are specified for access ports, then tagged packets might get sent out as untagged packets through the wrong access port, regardless of the PVID value.

Make sure you have added all needed interfaces to the bridge VLAN table when using bridge VLAN filtering. For routing functions to work properly on the same device through ports that use bridge VLAN filtering, you will need to allow access to the bridge interface (this automatically include a switch-cpu port when HW offloaded vlan-filtering is used, e.g. on CRS3xx series switches), this can be done by adding the bridge interface itself to the VLAN table, for tagged traffic you will need to add the bridge interface as a tagged port and create a VLAN interface on the bridge interface. Examples can be found in the inter-VLAN routing and Management port sections.

When allowing access to the CPU, you are allowing access from a certain port to the actual router/switch, this is not always desirable. Make sure you implement proper firewall filter rules to secure your device when access to the CPU is allowed from a certain VLAN ID and port, use firewall filter rules to allow access to only certain services.

Improperly configured bridge VLAN filtering can cause security issues, make sure you fully understand how Bridge VLAN table works before deploying your device into production environments.

Bridge port settings

Each bridge port have multiple VLAN related settings, that can change untagged VLAN membership, VLAN tagging/untagging behavior and packet filtering based on VLAN tag presence.

Sub-menu: `/interface bridge port`

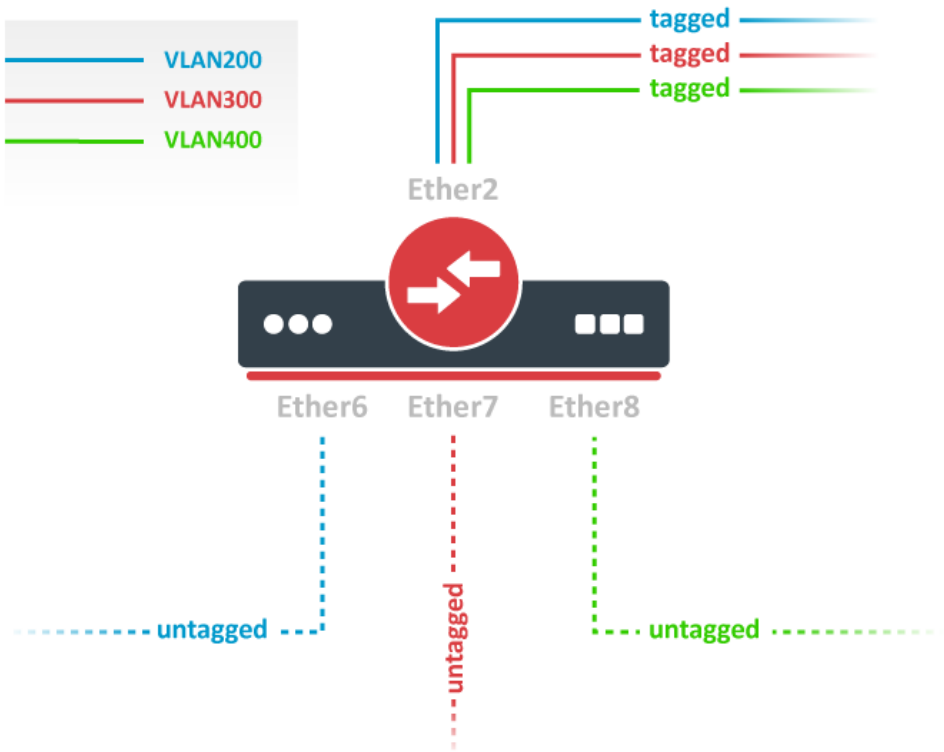
Property	Description
frame-types (*admit-all* | *admit-only-untagged-and-priority-tagged* | *admit-only-vlan-tagged*; Default: **admit-all**)	Specifies allowed ingress frame types on a bridge port. This property only has an effect when `vlan-filtering` is set to `yes`.
ingress-filtering (*yes* | *no*; Default: **yes**)	Enables or disables VLAN ingress filtering, which checks if the ingress port is a member of the received VLAN ID in the bridge VLAN table. Should be used with `frame-types` to specify if the ingress traffic should be tagged or untagged. This property only has effect when `vlan-filtering` is set to `yes`. The setting is enabled by default since RouterOS v7.
pvid (*integer 1..4094*; Default: **1**)	Port VLAN ID (pvid) specifies which VLAN the untagged ingress traffic is assigned to. This property only has an effect when `vlan-filtering` is set to `yes`.
tag-stacking (*yes* | *no*; Default: **no**)	Forces all packets to be treated as untagged packets. Packets on ingress port will be tagged with another VLAN tag regardless if a VLAN tag already exists, packets will be tagged with a VLAN ID that matches the `pvid` value and will use EtherType that is specified in `ether-type`. This property only has effect when `vlan-filtering` is set to `yes`.

Bridge host table

Bridge host table allows monitoring learned MAC addresses. When `vlan-filtering` is enabled, it shows learned VLAN ID as well (enabled independent-VLAN-learning or IVL).

```
[admin@MikroTik] > /interface bridge host print where !local
Flags: X - disabled, I - invalid, D - dynamic, L - local, E - external
 #       MAC-ADDRESS        VID ON-INTERFACE       BRIDGE
 0   D   CC:2D:E0:E4:B3:AA  300 ether3            bridge1
 1   D   CC:2D:E0:E4:B3:AB  400 ether4            bridge1
```

VLAN Example - Trunk and Access Ports



Create a bridge with disabled `vlan-filtering` to avoid losing access to the device before VLANs are completely configured. If you need a management access to the bridge, see the Management access configuration section.

```
/interface bridge
add name=bridge1 vlan-filtering=no
```

Add bridge ports and specify `pvid` for access ports to assign their untagged traffic to the intended VLAN. Use `frame-types` setting to accept only tagged or untagged packets.

```
/interface bridge port
add bridge=bridge1 interface=ether2 frame-types=admit-only-vlan-tagged
add bridge=bridge1 interface=ether6 pvid=200 frame-types=admit-only-untagged-and-priority-tagged
add bridge=bridge1 interface=ether7 pvid=300 frame-types=admit-only-untagged-and-priority-tagged
add bridge=bridge1 interface=ether8 pvid=400 frame-types=admit-only-untagged-and-priority-tagged
```

Add Bridge VLAN entries and specify tagged ports in them. Bridge ports with `frame-types` set to `admit-only-untagged-and-priority-tagged` will be automatically added as untagged ports for the `pvid` VLAN.

```
/interface bridge vlan
add bridge=bridge1 tagged=ether2 vlan-ids=200
add bridge=bridge1 tagged=ether2 vlan-ids=300
add bridge=bridge1 tagged=ether2 vlan-ids=400
```

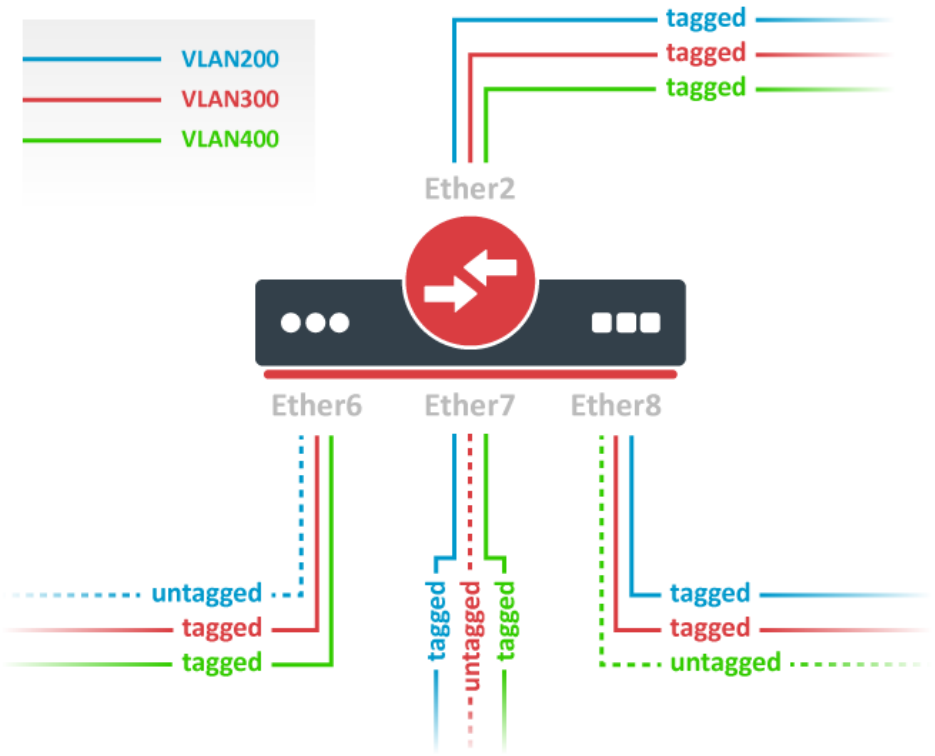
In the end, when VLAN configuration is complete, enable Bridge VLAN Filtering.

```
/interface bridge set bridge1 vlan-filtering=yes
```

Optional step is to set `frame-types=admit-only-vlan-tagged` on the bridge interface in order to disable the default untagged VLAN 1 (`pvid=1`).

```
/interface bridge set bridge1 frame-types=admit-only-vlan-tagged
```

VLAN Example - Trunk and Hybrid Ports



Create a bridge with disabled `vlan-filtering` to avoid losing access to the router before VLANs are completely configured. If you need a management access to the bridge, see the Management access configuration section.

```
/interface bridge
add name=bridge1 vlan-filtering=no
```

Add bridge ports and specify `pvid` on hybrid VLAN ports to assign untagged traffic to the intended VLAN. Use `frame-types` setting to accept only tagged packets on ether2.

```
/interface bridge port
add bridge=bridge1 interface=ether2 frame-types=admit-only-vlan-tagged
add bridge=bridge1 interface=ether6 pvid=200
add bridge=bridge1 interface=ether7 pvid=300
add bridge=bridge1 interface=ether8 pvid=400
```

Add Bridge VLAN entries and specify tagged ports in them. In this example egress VLAN tagging is done on ether6,ether7,ether8 ports too, making them into hybrid ports. Bridge ports with `frame-types` set to `admit-all` will be automatically added as untagged ports for the `pvid` VLAN.

```
/interface bridge vlan
add bridge=bridge1 tagged=ether2,ether7,ether8 vlan-ids=200
add bridge=bridge1 tagged=ether2,ether6,ether8 vlan-ids=300
add bridge=bridge1 tagged=ether2,ether6,ether7 vlan-ids=400
```

In the end, when VLAN configuration is complete, enable Bridge VLAN Filtering.

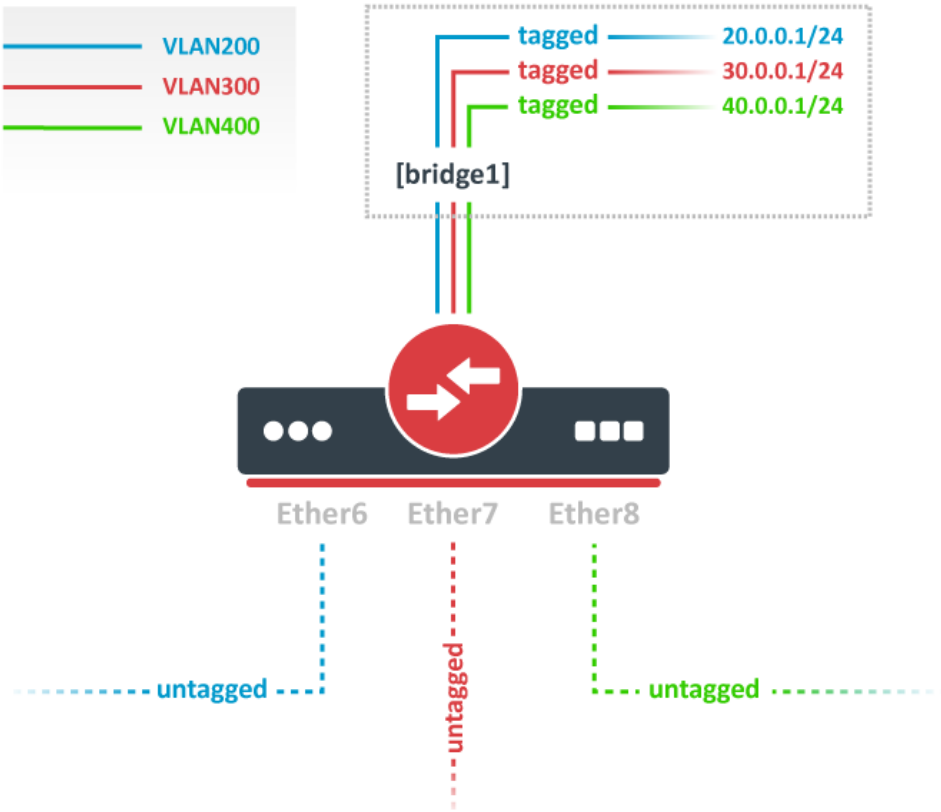
```
/interface bridge set bridge1 vlan-filtering=yes
```

Optional step is to set `frame-types=admit-only-vlan-tagged` on the bridge interface in order to disable the default untagged VLAN 1 (`pvid=1`).

```
/interface bridge set bridge1 frame-types=admit-only-vlan-tagged
```

> You don't have to add access ports as untagged ports, because they will be added dynamically as an untagged port with the VLAN ID that is specified in `pvid`, you can specify just the trunk port as a tagged port. All ports that have the same `pvid` set will be added as untagged ports in a single entry. You must take into account that the bridge itself is a port and it also has a `pvid` value, this means that the bridge port also will be added as an untagged port for the ports that have the same `pvid`. You can circumvent this behavior by either setting different `pvid` on all ports (even the trunk port and bridge itself), or to use `frame-type` set to `accept-only-vlan-tagged`.

VLAN Example - InterVLAN Routing by Bridge



Create a bridge with disabled `vlan-filtering` to avoid losing access to the router before VLANs are completely configured. If you need a management access to the bridge, see the Management access configuration section.

```
/interface bridge
add name=bridge1 vlan-filtering=no
```

Add bridge ports and specify `pvid` for VLAN access ports to assign their untagged traffic to the intended VLAN. Use `frame-types` setting to accept only untagged packets.

```
/interface bridge port
add bridge=bridge1 interface=ether6 pvid=200 frame-types=admit-only-untagged-and-priority-tagged
add bridge=bridge1 interface=ether7 pvid=300 frame-types=admit-only-untagged-and-priority-tagged
add bridge=bridge1 interface=ether8 pvid=400 frame-types=admit-only-untagged-and-priority-tagged
```

Add Bridge VLAN entries and specify tagged ports in them. In this example **bridge1** interface is the VLAN trunk that will send traffic further to do InterVLAN routing. Bridge ports with `frame-types` set to `admit-only-untagged-and-priority-tagged` will be automatically added as untagged ports for the `pvid` VLAN.

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1 vlan-ids=200
add bridge=bridge1 tagged=bridge1 vlan-ids=300
add bridge=bridge1 tagged=bridge1 vlan-ids=400
```

Configure VLAN interfaces on the **bridge1** to allow handling of tagged VLAN traffic at routing level and set IP addresses to ensure routing between VLANs as planned.

```
/interface vlan
add interface=bridge1 name=VLAN200 vlan-id=200
add interface=bridge1 name=VLAN300 vlan-id=300
add interface=bridge1 name=VLAN400 vlan-id=400

/ip address
add address=20.0.0.1/24 interface=VLAN200
add address=30.0.0.1/24 interface=VLAN300
add address=40.0.0.1/24 interface=VLAN400
```

In the end, when VLAN configuration is complete, enable Bridge VLAN Filtering:

```
/interface bridge set bridge1 vlan-filtering=yes
```

Optional step is to set `frame-types=admit-only-vlan-tagged` on the bridge interface in order to disable the default untagged VLAN 1 (`pvid=1`).

```
/interface bridge set bridge1 frame-types=admit-only-vlan-tagged
```

Since RouterOS v7, it is possible to route traffic using the L3 HW offloading on certain devices. See more details on L3 Hardware Offloading.

Management access configuration

There are multiple ways to set up management access on a device that uses bridge VLAN filtering. Below are some of the most popular approaches to properly enable access to a router/switch. Start by creating a bridge without VLAN filtering enabled:

```
/interface bridge
add name=bridge1 vlan-filtering=no
```

Untagged access without VLAN filtering

In case VLAN filtering will not be used and access with untagged traffic is desired, the only requirement is to create an IP address on the bridge interface.

```
/ip address
add address=192.168.99.1/24 interface=bridge1
```

Tagged access without VLAN filtering

In case VLAN filtering will not be used and access with tagged traffic is desired, create a routable VLAN interface on the bridge and add an IP address on the VLAN interface.

```
/interface vlan
add interface=bridge1 name=MGMT vlan-id=99
/ip address
add address=192.168.99.1/24 interface=MGMT
```

Tagged access with VLAN filtering

In case VLAN filtering is used and access with tagged traffic is desired, additional steps are required. In this example, VLAN 99 will be used to access the device. A VLAN interface on the bridge must be created and an IP address must be assigned to it.

```
/interface vlan
add interface=bridge1 name=MGMT vlan-id=99
/ip address
add address=192.168.99.1/24 interface=MGMT
```

For example, if you want to allow access to the device from ports **ether3**, **ether4, sfp-sfpplus1** using tagged VLAN 99 traffic, then you must add this entry to the VLAN table. Note that the **bridge1** interface is also included in the tagged port list:

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1,ether3,ether4,sfp-sfpplus1 vlan-ids=99
```

After that you can enable VLAN filtering:

```
/interface bridge set bridge1 vlan-filtering=yes
```

Untagged access with VLAN filtering

In case VLAN filtering is used and access with untagged traffic is desired, the VLAN interface must use the same VLAN ID as the untagged port VLAN ID (`pvid`). Just like in the previous example, start by creating a VLAN interface on the bridge and add an IP address for the VLAN.

```
/interface vlan
add interface=bridge1 name=MGMT vlan-id=99
/ip address
add address=192.168.99.1/24 interface=MGMT
```

For example, untagged ports **ether2** and **ether3** should be able to communicate with the VLAN 99 interface using untagged traffic. In order to achieve this, these ports should be configured with the `pvid` that matches the VLAN ID on management VLAN. Note that the **bridge1** interface is a tagged port member, you can configure additional tagged ports if necessary (see the previous example).

```
/interface bridge port
set [find interface=ether2] pvid=99
set [find interface=ether3] pvid=99
/interface bridge vlan
add bridge=bridge1 tagged=bridge1 untagged=ether2,ether3 vlan-ids=99
```

After that you can enable VLAN filtering:

```
/interface bridge set bridge1 vlan-filtering=yes
```

Changing untagged VLAN for the bridge interface

In case VLAN filtering is used, it is possible to change the untagged VLAN ID for the bridge interface using the `pvid` setting. Note that creating routable VLAN interfaces and allowing tagged traffic on the bridge is a more flexible and generally recommended option.

First, create an IP address on the bridge interface.

```
/ip address
add address=192.168.99.1/24 interface=bridge1
```

For example, untagged **bridge1** traffic should be able to communicate with untagged **ether2** and **ether3** ports and tagged **sfp-sfpplus1** port in VLAN 99. In order to achieve this, **bridge1**, **ether2**, **ether3** should be configured with the same `pvid` and sfp-sfpplus1 added as a tagged member.

```
/interface bridge
set [find name=bridge1] pvid=99
/interface bridge port
set [find interface=ether2] pvid=99
set [find interface=ether3] pvid=99
/interface bridge vlan
add bridge=bridge1 tagged=sfp-sfpplus1 untagged=bridge1,ether2,ether3 vlan-ids=99
```

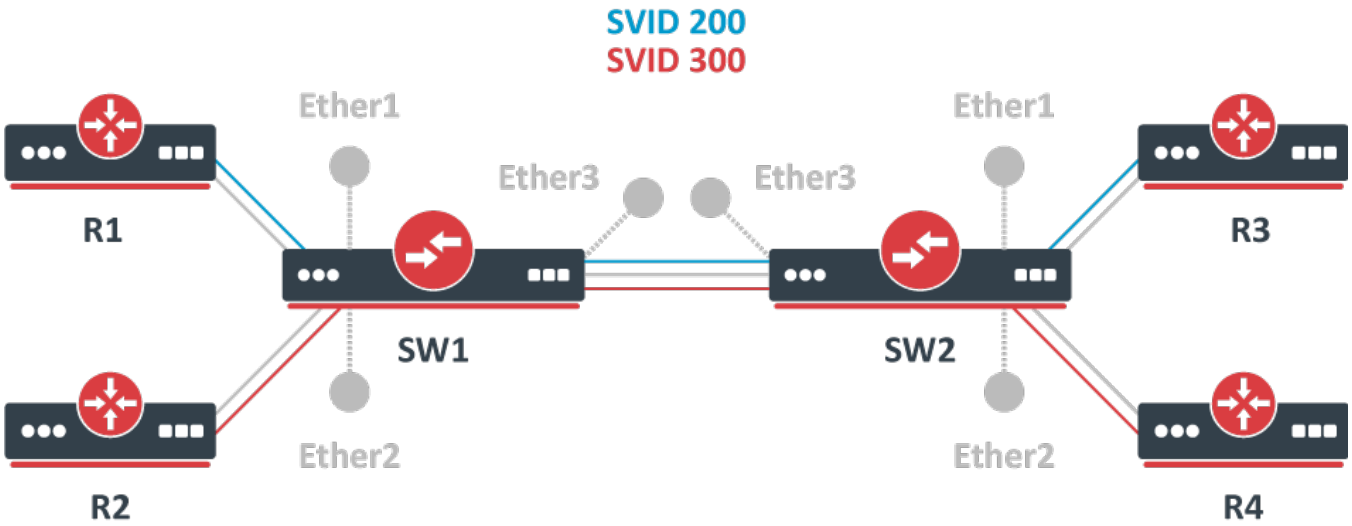
After that you can enable VLAN filtering:

```
/interface bridge set bridge1 vlan-filtering=yes
```

> ⚠ If the connection to the router/switch through an IP address is not required, then steps adding an IP address can be skipped since a connection to the router/switch through Layer2 protocols (e.g. MAC-telnet) will be working either way.

VLAN Tunneling (QinQ)

Since RouterOS v6.43 the RouterOS bridge is IEEE 802.1ad compliant and it is possible to filter VLAN IDs based on Service VLAN ID (0x88a8) rather than Customer VLAN ID (0x8100). The same principles can be applied as with IEEE 802.1Q VLAN filtering (the same setup examples can be used). Below is a topology for a common **Provider bridge**:



In this example, **R1**, **R2**, **R3,** and **R4** might be sending any VLAN tagged traffic by 802.1Q (CVID), but **SW1** and **SW2** needs isolate traffic between routers in a way that **R1** is able to communicate only with **R3**, and **R2** is only able to communicate with **R4**. To do so, you can tag all ingress traffic with an SVID and only allow these VLANs on certain ports. Start by enabling the service tag 0x88a8, introduced by `802.1ad`, on the bridge. Use these commands on **SW1** and **SW2**:

```
/interface bridge
add name=bridge1 vlan-filtering=no ether-type=0x88a8
```

In this setup, **ether1** and **ether2** are going to be access ports (untagged), use the `pvid` parameter to tag all ingress traffic on each port, use these commands on **SW1** and **SW2**:

```
/interface bridge port
add interface=ether1 bridge=bridge1 pvid=200
add interface=ether2 bridge=bridge1 pvid=300
add interface=ether3 bridge=bridge1
```

Specify tagged and untagged ports in the bridge VLAN table, use these commands on **SW1** and **SW2**:

```
/interface bridge vlan
add bridge=bridge1 tagged=ether3 untagged=ether1 vlan-ids=200
add bridge=bridge1 tagged=ether3 untagged=ether2 vlan-ids=300
```

When the bridge VLAN table is configured, you can enable bridge VLAN filtering, use these commands on **SW1** and **SW2:**

```
/interface bridge set bridge1 vlan-filtering=yes
```

By enabling vlan-filtering you will be filtering out traffic destined to the CPU, before enabling VLAN filtering you should make sure that you set up a Management port.

Note, that if you are using the new EtherType/TPID 0x88a8 (service tag) and you also need a VLAN interface for your Service VLAN, you will also have to apply the `use-service-tag` parameter on the VLAN interface.

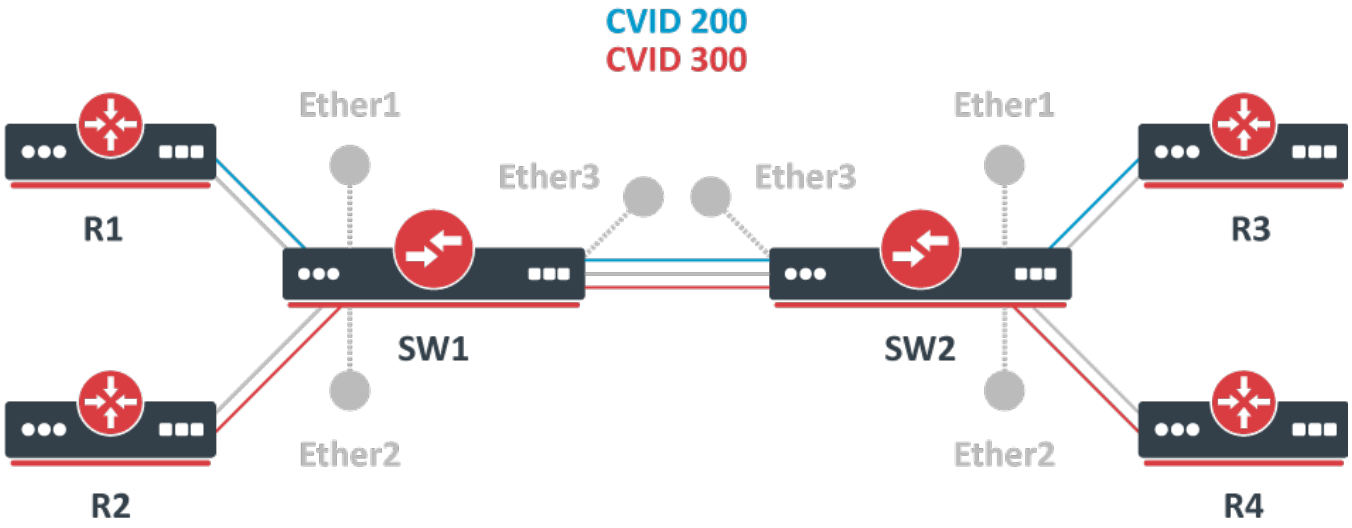
When `ether-type=0x8100` is configured, the bridge checks the outer VLAN tag and sees if it is using EtherType `0x8100`. If the bridge receives a packet with an outer tag that has a different EtherType, it will mark the packet as `untagged`. Since RouterOS only checks the outer tag of a packet, it is not possible to filter 802.1Q packets when the 802.1ad protocol is used.

Currently, only CRS3xx, CRS5xx series switches and CCR2116, CCR2216 routers are capable of hardware offloaded VLAN filtering using the Service tag, EtherType/TPID `0x88a8`.

Devices with switch chip Marvell-98DX3257 (e.g. CRS354 series) do not support VLAN filtering on 1Gbps Ethernet interfaces for other VLAN types (`0x88a8` and `0x9100`).

Tag stacking

Since RouterOS v6.43 it is possible to forcefully add a new VLAN tag over any existing VLAN tags, this feature can be used to achieve a CVID stacking setup, where a CVID (0x8100) tag is added before an existing CVID tag. This type of setup is very similar to the Provider bridge setup, to achieve the same setup but with multiple CVID tags (CVID stacking) we can use the same topology:



In this example **R1**, **R2**, **R3,** and **R4** might be sending any VLAN tagged traffic, it can be 802.1ad, 802.1Q or any other type of traffic, but **SW1** and **SW2** needs isolate traffic between routers in a way that **R1** is able to communicate only with **R3**, and **R2** is only able to communicate with **R4**. To do so, you can tag all ingress traffic with a new CVID tag and only allow these VLANs on certain ports. Start by selecting the proper EtherType, use these commands on **SW1** and **SW2**:

```
/interface bridge
add name=bridge1 vlan-filtering=no ether-type=0x8100
```

In this setup, **ether1** and **ether2** will ignore any VLAN tags that are present and add a new VLAN tag, use the `pvid` parameter to tag all ingress traffic on each port and allow `tag-stacking` on these ports, use these commands on **SW1** and **SW2**:

```
/interface bridge port
add interface=ether1 bridge=bridge1 pvid=200 tag-stacking=yes
add interface=ether2 bridge=bridge1 pvid=300 tag-stacking=yes
add interface=ether3 bridge=bridge1
```

Specify tagged and untagged ports in the bridge VLAN table, you only need to specify the VLAN ID of the outer tag, use these commands on **SW1** and **SW2**:

```
/interface bridge vlan
add bridge=bridge1 tagged=ether3 untagged=ether1 vlan-ids=200
add bridge=bridge1 tagged=ether3 untagged=ether2 vlan-ids=300
```

When the bridge VLAN table is configured, you can enable bridge VLAN filtering, which is required in order for the `pvid` parameter to have any effect, use these commands on **SW1** and **SW2:**

```
/interface bridge set bridge1 vlan-filtering=yes
```

> By enabling vlan-filtering you will be filtering out traffic destined to the CPU, before enabling VLAN filtering you should make sure that you set up a Management port.

MVRP

Multiple VLAN Registration protocol (MVRP) is a protocol based on Multiple Registration Protocol (MRP) which allows to register attributes (VLAN IDs in case of MVRP) with other members of Bridged LAN.

An MRP application can make or withdraw declarations of attributes which result in registration or leaving of those attributes with other MRP participants.

Here's how it works.

MRP consists of two parts:

- **Applicant** - responsible for sending declarations (or leaves). Its behavior can be configured on a per-port basis using the setting called `mvrp-applicant-state`, and per-VLAN using the `mvrp-forbidden` setting.
- **Registrar** - responsible for registering incoming declarations. Its configuration can be set per-port using the `mvrp-registrar-state` setting, and per-VLAN using the `mvrp-forbidden` setting.

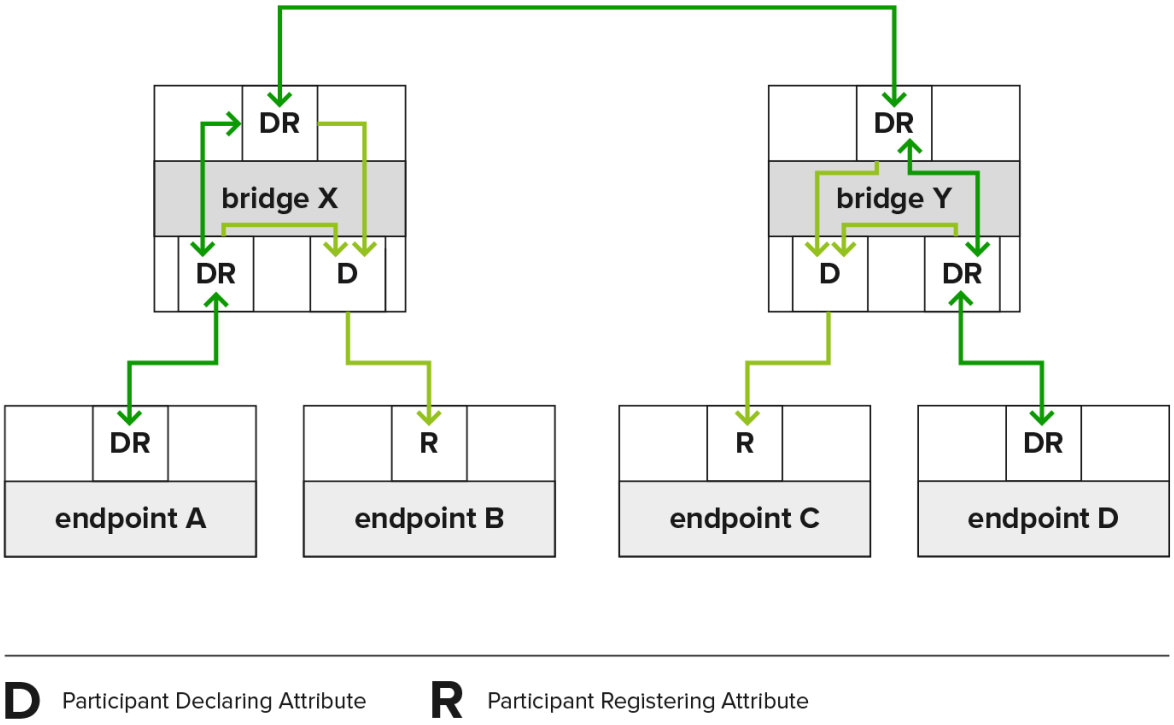
Registration Propagation: Incoming registration on a bridge port dynamically makes that specific port a tagged VLAN member. Additionally, the attributes associated with this registration are spread to all active (forwarding) bridge ports as a declaration.

Declaration Operation: In case of MVRP, the configured VLAN's get declared on each port, but they will only get configured as members of those VLAN's when a declaration is received from the LAN (Registrar will register the VLAN). From the perspective of an end-station, a single declaration will be registered on each upstream port across the entire LAN. When another end-station declares the same attribute, a path of registrations will be made between the two (or more) end stations, see the picture below.

MVRP helps to dynamically propagate VLAN information throughout the bridged network and configure VLANs only on the needed ports. This makes the network efficient by avoiding unnecessary traffic flooding.

As noted before, MVRP is only active on ports that are forwarding. In case of MSTP declarations and registrations are made only if the port is forwarding in the MSTI in which VLAN is mapped.

The point-to-point ports speed up the process of registration (or leaving). Manually configuring `point-to-point=yes` can be advantageous for non-Ethernet interfaces.



Property Reference

Sub-menu: `/interface bridge`

Property	Description
mvrp (*yes* | *no*; Default: **no**)	Enables MVRP for bridge. It ensures that the MAC address 01:80:C2:00:00:21 is trapped and not forwarded, the `vlan-filtering` must be enabled.

Sub-menu: `/interface bridge port`

The port menu enables control over the applicant and registrar settings on a per-port basis.

Property	Description
mvrp-applicant-state (*non-participant* | *normal-participant;* Default: **normal-participant**)	MVRP applicant options: • **non-participant** - port does not send any MRP messages; • **normal-participant** - port participates normally in MRP exchanges.

mvrp-registrar-state (*fixed* | *normal*; Default: **normal**)	MVRP registrar options: • **fixed** - port ignores all MRP messages, and remains Registered (IN) in all configured vlans. • **normal** - port receives MRP messages and handles them according to the standard.

To monitor the currently declared and registered VLAN IDs, use the `monitor` command.

```
[admin@MikroTik] > interface/bridge/port monitor [find interface=sfp-sfpplus1]
            interface: sfp-sfpplus1
               status: in-bridge
          port-number: 1
                 role: designated-port
            edge-port: no
  edge-port-discovery: yes
  point-to-point-port: yes
         external-fdb: no
         sending-rstp: yes
             learning: yes
           forwarding: yes
     actual-path-cost: 2000
     hw-offload-group: switch1
    declared-vlan-ids: 1,10,20-21
  registered-vlan-ids: 1,10,20,30-33
```

Sub-menu: `/interface bridge vlan`

All ports that are members of static VLANs or dynamic untagged VLANs created by the port `pvid` setting are treated as "fixed." Meaning the registrar disregards all MRP messages and remains registered (IN) for those VLANs.

When VLAN is neither manually configured nor created by the port `pvid` setting, incoming registrations on a bridge port can dynamically designate that specific port as a tagged VLAN member. The `mvrp-forbidden` feature allows creating a list of ports that are restricted from registering into a specific VLAN ID.

VLANs that are static or dynamic will be declared by the applicants unless this functionality is disabled by the port's `mvrp-applicant-state`, or by VLAN's `mvrp-forbidden` setting.

Property	Description
mvrp-forbidden (*interfaces*; Default:)	Ports that ignore all MRP messages and remains Not Registered (MT), as well as disables applicant from declaring specific VLAN ID.

Sub-menu: `/interface bridge vlan mvrp`

The MVRP attributes menu can be used to see internal MVRP attribute states, as specified in the IEEE 802.1Q-2011.

Property	Description

applicant-state	The Applicant state machine that declares attributes. Its state can be VO, VP, VN, AN, AA, QA, LA, AO, QO, AP, QP, or LO. Each state consists of two letters. The first letter indicates the state: • V—Very anxious; • A—Anxious; • Q—Quiet; • L—Leaving. The second letter indicates the membership state: • A - Active member; • P - Passive member; • O - Observer; • N - New. For example, VP indicates "Very anxious, Passive member."
registrar-state	The Registrar state machine that records the registration state of attributes declared by other participants. Its state can be IN, LV, or MT: • IN—Registered; • LV—Previously registered, but now being timed out; • MT—Not registered.

```
[admin@Mikrotik] /interface/bridge/vlan/mvrp print where vlan-id=10
Columns: BRIDGE, PORT, VLAN-ID, REGISTRAR-STATE, APPLICANT-STATE, LAST-EVENT
 #  BRIDGE    PORT           VLAN-ID  REGISTRAR-STATE  APPLICANT-STATE  LAST-EVENT
 1  bridge67  sfp-sfpplus1        10  IN               Quiet Active     JoinIn
 9  bridge67  sfp-sfpplus5        10  MT               Quiet Active     JoinEmpty
17  bridge67  sfp-sfpplus9        10  MT               Quiet Active     JoinEmpty
25  bridge67  sfp-sfpplus13       10  IN               Quiet Active     JoinIn
```

Fast Forward

Fast Forward allows forwarding packets faster under special conditions. When Fast Forward is enabled, then the bridge can process packets even faster since it can skip multiple bridge-related checks, including MAC learning. Below you can find a list of conditions that **MUST** be met in order for Fast Forward to be active:

- Bridge has `fast-forward` set to `yes`
- Bridge has only 2 running ports
- Both bridge ports support Fast Path, Fast Path is active on ports and globally on the bridge
- Bridge Hardware Offloading is disabled
- Bridge VLAN Filtering is disabled
- Bridge DHCP snooping is disabled
- `unknown-multicast-flood` is set to `yes`
- `unknown-unicast-flood` is set to `yes`
- `broadcast-flood` is set to `yes`
- MAC address for the bridge matches with a MAC address from one of the bridge slave ports
- `horizon` for both ports is set to `none`

⚠ Fast Forward disables MAC learning, this is by design to achieve faster packet forwarding. MAC learning prevents traffic from flooding multiple interfaces, but MAC learning is not needed when a packet can only be sent out through just one interface.

Fast Forward is disabled when hardware offloading is enabled. Hardware offloading can achieve full write-speed performance when it is active since it will use the built-in switch chip (if such exists on your device), fast forward uses the CPU to forward packets. When comparing throughput results, you would get such results: Hardware offloading > Fast Forward > Fast Path > Slow Path.

It is possible to check how many packets where processed by Fast Forward:

```
[admin@MikroTik] /interface bridge settings> pr
              use-ip-firewall: no
     use-ip-firewall-for-vlan: no
    use-ip-firewall-for-pppoe: no
              allow-fast-path: yes
      bridge-fast-path-active: yes
     bridge-fast-path-packets: 0
       bridge-fast-path-bytes: 0
  bridge-fast-forward-packets: 16423
    bridge-fast-forward-bytes: 24864422
```

If packets are processed by Fast Path, then Fast Forward is not active. Packet count can be used as an indicator of whether Fast Forward is active or not.

Since RouterOS 6.44 it is possible to monitor Fast Forward status, for example:

```
[admin@MikroTik] /interface bridge monitor bridge1
                  state: enabled
    current-mac-address: B8:69:F4:C9:EE:D7
            root-bridge: yes
         root-bridge-id: 0x8000.B8:69:F4:C9:EE:D7
         root-path-cost: 0
              root-port: none
             port-count: 2
  designated-port-count: 2
           fast-forward: yes
```

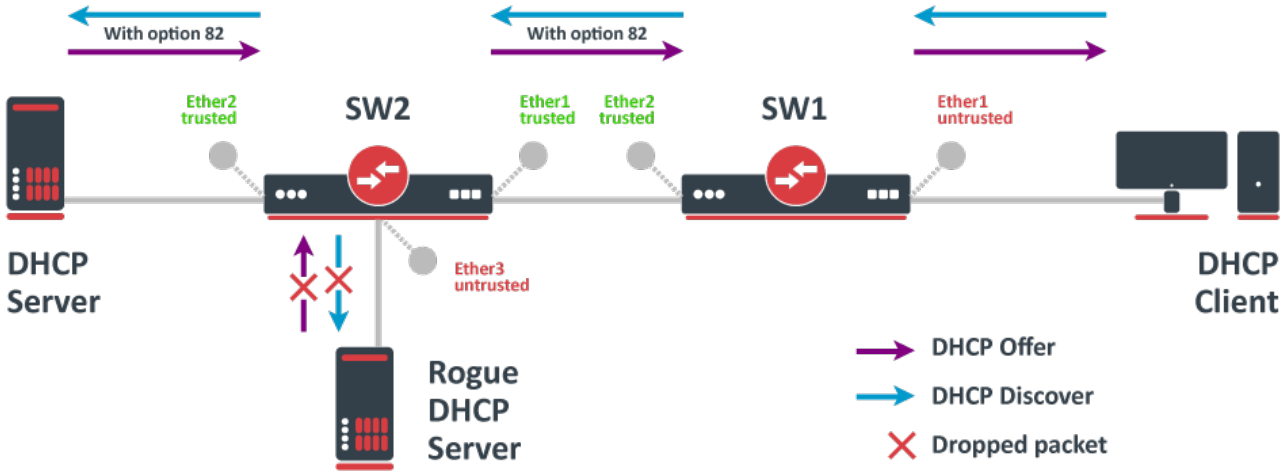
Disabling or enabling fast-forward will temporarily disable all bridge ports for settings to take effect. This must be taken into account whenever changing this property on production environments since it can cause all packets to be temporarily dropped.

IGMP/MLD Snooping

The bridge supports IGMP/MLD snooping. It controls multicast streams and prevents multicast flooding on unnecessary ports. Its settings are placed in the bridge menu and it works independently in every bridge interface. Software-driven implementation works on all devices with RouterOS, but CRS3xx, CRS5xx series switches, CCR2116, CR2216 routers, and 88E6393X, 88E6191X, 88E6190 switch chips also support IGMP/MLD snooping with hardware offloading. See more details on IGMP/MLD snooping manual.

DHCP Snooping and DHCP Option 82

DHCP Snooping and DHCP Option 82 is supported by bridge. The DHCP Snooping is a Layer2 security feature, that limits unauthorized DHCP servers from providing malicious information to users. In RouterOS, you can specify which bridge ports are trusted (where known DHCP server resides and DHCP messages should be forwarded) and which are untrusted (usually used for access ports, received DHCP server messages will be dropped). The DHCP Option 82 is additional information (Agent Circuit ID and Agent Remote ID) provided by DHCP Snooping enabled devices that allow identifying the device itself and DHCP clients.



In this example, SW1 and SW2 are DHCP Snooping, and Option 82 enabled devices. First, we need to create a bridge, assign interfaces and mark trusted ports. Use these commands on **SW1**:

```
/interface bridge
add name=bridge
/interface bridge port
add bridge=bridge interface=ether1
add bridge=bridge interface=ether2 trusted=yes
```

For SW2, the configuration will be similar, but we also need to mark ether1 as trusted, because this interface is going to receive DHCP messages with Option 82 already added. You need to mark all ports as trusted if they are going to receive DHCP messages with added Option 82, otherwise these messages will be dropped. Also, we add ether3 to the same bridge and leave this port untrusted, imagine there is an unauthorized (rogue) DHCP server. Use these commands on **SW2**:

```
/interface bridge
add name=bridge
/interface bridge port
add bridge=bridge interface=ether1 trusted=yes
add bridge=bridge interface=ether2 trusted=yes
add bridge=bridge interface=ether3
```

Then we need to enable DHCP Snooping and Option 82. In case your DHCP server does not support DHCP Option 82 or you do not implement any Option 82 related policies, this option can be disabled. Use these commands on **SW1** and **SW2**:

```
/interface bridge
set [find where name="bridge"] dhcp-snooping=yes add-dhcp-option82=yes
```

Now both devices will analyze what DHCP messages are received on bridge ports. The **SW1** is responsible for adding and removing the DHCP Option 82. The **SW2** will limit rogue DHCP server from receiving any discovery messages and drop malicious DHCP server messages from ether3.

⚠ Currently, CRS3xx, CRS5xx series switches, CCR2116, CR2216 routers, and 88E6393X, 88E6191X, 88E6190 switch chips fully support hardware offloaded DHCP Snooping and Option 82. For CRS1xx and CRS2xx series switches it is possible to use DHCP Snooping along with VLAN switching, but then you need to make sure that DHCP packets are sent out with the correct VLAN tag using egress ACL rules. Other devices are capable of using DHCP Snooping and Option 82 features along with hardware offloading, but you must make sure that there is no VLAN-related configuration applied on the device, otherwise, DHCP Snooping and Option 82 might not work properly. See the Br idge Hardware Offloading section with supported features.

For CRS3xx, CRS5xx series switches and CCR2116, CR2216 routers DHCP snooping will not work when hardware offloading bonding interfaces are created.

Controller Bridge and Port Extender

Controller Bridge (CB) and Port Extender (PE) is an IEEE 802.1BR standard implementation in RouterOS for CRS3xx, CRS5xx series switches and CCR2116, CCR2216 routers. It allows virtually extending the CB ports with a PE device and managing these extended interfaces from a single controlling device. Such configuration provides a simplified network topology, flexibility, increased port density, and ease of manageability. See more details on Controller Bridge and Port Extender manual.

Bridge Firewall

The bridge firewall implements packet filtering and thereby provides security functions that are used to manage data flow to, from, and through the bridge.

Packet flow diagram shows how packets are processed through the router. It is possible to force bridge traffic to go through `/ip firewall filter` rules (see the bridge settings).

There are two bridge firewall tables:

- **filter** - bridge firewall with three predefined chains:
  - **input** - filters packets, where the destination is the bridge (including those packets that will be routed, as they are destined to the bridge MAC address anyway)
  - **output** - filters packets, which come from the bridge (including those packets that has been routed normally)
  - **forward** - filters packets, which are to be bridged (note: this chain is not applied to the packets that should be routed through the router, just to those that are traversing between the ports of the same bridge)
- **nat** - bridge network address translation provides ways for changing source/destination MAC addresses of the packets traversing a bridge. Has two built-in chains:
  - **srcnat** - used for "hiding" a host or a network behind a different MAC address. This chain is applied to the packets leaving the router through a bridged interface
  - **dstnat** - used for redirecting some packets to other destinations

You can put packet marks in bridge firewall (filter and NAT), which are the same as the packet marks in IP firewall configured by '`/ip firewall mangle`'. In this way, packet marks put by bridge firewall can be used in 'IP firewall', and vice versa.

General bridge firewall properties are described in this section. Some parameters that differ between nat and filter rules are described in further sections.

Sub-menu: `/interface bridge filter, /interface bridge nat`

Property	Description
802.3-sap (*integer*; Default:)	DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) are 2 one-byte fields, which identify the network protocol entities which use the link-layer service. These bytes are always equal. Two hexadecimal digits may be specified here to match an SAP byte.
802.3-type (*integer*; Default:)	Ethernet protocol type, placed after the IEEE 802.2 frame header. Works only if 802.3-sap is 0xAA (SNAP - Sub-Network Attachment Point header). For example, AppleTalk can be indicated by the SAP code of 0xAA followed by a SNAP type code of 0x809B.

action (*accept | drop | jump | log | mark-packet | passthrough | return | set-priority*; Default:)	Action to take if the packet is matched by the rule: • accept - accept the packet. The packet is not passed to the next firewall rule • drop - silently drop the packet • jump - jump to the user-defined chain specified by the value of `jump-target` parameter • log - add a message to the system log containing the following data: in-interface, out-interface, src-mac, protocol, src-ip:port->dst-ip:port and length of the packet. After the packet is matched it is passed to the next rule in the list, similar as `passthrough` • mark-packet - place a mark specified by the new-packet-mark parameter on a packet that matches the rule • passthrough - if the packet is matched by the rule, increase counter and go to next rule (useful for statistics) • return - passes control back to the chain from where the jump took place • set-priority - set priority specified by the new-priority parameter on the packets sent out through a link that is capable of transporting priority (VLAN or WMM-enabled wireless interface). Read more
arp-dst-address (*IP address*; Default:)	ARP destination IP address.
arp-dst-mac-address (*MAC address*; Default:)	ARP destination MAC address.
arp-gratuitous (*yes | no*; Default:)	Matches ARP gratuitous packets.
arp-hardware-type (*integer*; Default: **1**)	ARP hardware type. This is normally Ethernet (Type 1).
arp-opcode (*arp-nak | drarp-error | drarp-reply | drarp-request | inarp-reply | inarp-request | reply | reply-reverse | request | request-reverse*; Default:)	ARP opcode (packet type) • arp-nak - negative ARP reply (rarely used, mostly in ATM networks) • drarp-error - Dynamic RARP error code, saying that an IP address for the given MAC address can not be allocated • drarp-reply - Dynamic RARP reply, with a temporary IP address assignment for a host • drarp-request - Dynamic RARP request to assign a temporary IP address for the given MAC address • inarp-reply - InverseARP Reply • inarp-request - InverseARP Request • reply - standard ARP reply with a MAC address • reply-reverse - reverse ARP (RARP) reply with an IP address assigned • request - standard ARP request to a known IP address to find out unknown MAC address • request-reverse - reverse ARP (RARP) request to a known MAC address to find out the unknown IP address (intended to be used by hosts to find out their own IP address, similarly to DHCP service)
arp-packet-type (*integer 0..65535 | hex 0x0000-0xffff*; Default:)	ARP Packet Type.
arp-src-address (*IP address*; Default:)	ARP source IP address.
arp-src-mac-address (*MAC addres*; Default:)	ARP source MAC address.
chain (*text*; Default:)	Bridge firewall chain, which the filter is functioning in (either a built-in one, or a user-defined one).
dst-address (*IP address*; Default:)	Destination IP address (only if MAC protocol is set to IP).
dst-address6 (*IPv6 address*; Default:)	Destination IPv6 address (only if MAC protocol is set to IPv6).
dst-mac-address (*MAC address*; Default:)	Destination MAC address.

dst-port (*integer 0..65535*; Default:)	Destination port number or range (only for TCP or UDP protocols).
in-bridge (*name*; Default:)	Bridge interface through which the packet is coming in.
in-bridge-list (*name*; Default:)	Set of bridge interfaces defined in interface list. Works the same as `in-bridge`.
in-interface (*name*; Default:)	Physical interface (i.e., bridge port) through which the packet is coming in.
in-interface-list (*name*; Default:)	Set of interfaces defined in interface list. Works the same as `in-interface`.
ingress-priority (*integer 0..63*; Default:)	Matches the priority of an ingress packet. Priority may be derived from VLAN, WMM, DSCP or MPLS EXP bit. read more
ip-protocol (*dccp | ddp | egp | encap | etherip | ggp | gre | hmp | icmp | icmpv6 | idpr-cmtp | igmp | ipencap | ipip | ipsec-ah | ipsec-esp | ipv6 | ipv6-frag | ipv6-nonxt | ipv6-opts | ipv6-route | iso-tp4 | l2tp | ospf | pim | pup | rdp | rspf | rsvp | sctp | st | tcp | udp | udp-lite | vmtp | vrrp | xns-idp | xtp*; Default:)	IP protocol (only if MAC protocol is set to IPv4) • dccp - Datagram Congestion Control Protocol • ddp - Datagram Delivery Protocol • egp - Exterior Gateway Protocol • encap - Encapsulation Header • etherip - Ethernet-within-IP Encapsulation • ggp - Gateway-to-Gateway Protocol • gre - Generic Routing Encapsulation • hmp - Host Monitoring Protocol • icmp - IPv4 Internet Control Message Protocol • icmpv6 - IPv6 Internet Control Message Protocol • idpr-cmtp - Inter-Domain Policy Routing Control Message Transport Protocol • igmp - Internet Group Management Protocol • ipencap - IP in IP (encapsulation) • ipip - IP-within-IP Encapsulation Protocol • ipsec-ah - IPsec Authentication Header • ipsec-esp - IPsec Encapsulating Security Payload • ipv6 - Internet Protocol version 6 • ipv6-frag - Fragment Header for IPv6 • ipv6-nonxt - No Next Header for IPv6 • ipv6-opts - Destination Options for IPv6 • ipv6-route - Routing Header for IPv6 • iso-tp4 - ISO Transport Protocol Class 4 • l2tp - Layer Two Tunneling Protocol • ospf - Open Shortest Path First • pim - Protocol Independent Multicast • pup - PARC Universal Packet • rdp - Reliable Data Protocol • rspf - Radio Shortest Path First • rsvp - Reservation Protocol • sctp - Stream Control Transmission Protocol • st - Internet Stream Protocol • tcp - Transmission Control Protocol • udp - User Datagram Protocol • udp-lite - Lightweight User Datagram Protocol • vmtp - Versatile Message Transaction Protocol • vrrp - Virtual Router Redundancy Protocol • xns-idp - Xerox Network Systems Internet Datagram Protocol • xtp - Xpress Transport Protocol
jump-target (*name*; Default:)	If `action=jump` specified, then specifies the user-defined firewall chain to process the packet.

limit (*integer/time,integer*; Default:)	Restricts packet match rate to a given limit. • count - maximum average packet rate, measured in packets per second (pps), unless followed by Time option • time - specifies the time interval over which the packet rate is measured • burst - number of packets to match in a burst
log (*yes* | *no; Default:* **no**)	Add a message to the system log containing the following data: in-interface, out-interface, src-mac, dst-mac, eth-protocol, ip-protocol, src-ip:port->dst-ip:port, and length of the packet.
log-prefix (*text*; Default:)	Defines the prefix to be printed before the logging information.
mac-protocol (*802.2* | *arp* | *homeplug-av* | *ip* | *ipv6* | *ipx* | *length* | *lldp* | *loop-protect* | *mpls-multicast* | *mpls-unicast* | *packing-compr* | *packing-simple* | *pppoe* | *pppoe-discovery* | *rarp* | *service-vlan* | *vlan* | *integer 0..65535* | *hex 0x0000-0xffff*; Default:)	Ethernet payload type (MAC-level protocol). To match protocol type for VLAN encapsulated frames (0x8100 or 0x88a8), a vlan-encap prop erty should be used. • 802.2 - 802.2 Frames (0x0004) • arp - Address Resolution Protocol (0x0806) • homeplug-av - HomePlug AV MME (0x88E1) • ip - Internet Protocol version 4 (0x0800) • ipv6 - Internet Protocol Version 6 (0x86DD) • ipx - Internetwork Packet Exchange (0x8137) • length - Packets with length field (0x0000-0x05DC) • lldp - Link Layer Discovery Protocol (0x88CC) • loop-protect - Loop Protect Protocol (0x9003) • mpls-multicast - MPLS multicast (0x8848) • mpls-unicast - MPLS unicast (0x8847) • packing-compr - Encapsulated packets with compressed IP packing (0x9001) • packing-simple - Encapsulated packets with simple IP packing (0 x9000) • pppoe - PPPoE Session Stage (0x8864) • pppoe-discovery - PPPoE Discovery Stage (0x8863) • rarp - Reverse Address Resolution Protocol (0x8035) • service-vlan - Provider Bridging (IEEE 802.1ad) & Shortest Path Bridging IEEE 802.1aq (0x88A8) • vlan - VLAN-tagged frame (IEEE 802.1Q) and Shortest Path Bridging IEEE 802.1aq with NNI compatibility (0x8100)
new-packet-mark (*string*; Default:)	Sets a new packet-mark value.
new-priority (*integer* | *from-ingress*; Default:)	Sets a new priority for a packet. This can be the VLAN, WMM or MPLS EXP priority Read more. This property can also be used to set an internal priori
out-bridge (*name*; Default:)	Outgoing bridge interface.
out-bridge-list (*name*; Default:)	Set of bridge interfaces defined in interface list. Works the same as `out-bridge`.
out-interface (*name*; Default:)	Interface that the packet is leaving the bridge through.
out-interface-list (*name*; Default:)	Set of interfaces defined in interface list. Works the same as `out-interface`.
packet-mark (*name*; Default:)	Match packets with a certain packet mark.

packet-type (*broadcast* / *host* / *multicast* / *other-host*; Default:)	MAC frame type: • broadcast - broadcast MAC packet • host - packet is destined to the bridge itself • multicast - multicast MAC packet • other-host - packet is destined to some other unicast address, not to the bridge itself
src-address (*IP address*; Default:)	Source IP address (only if MAC protocol is set to IPv4).
src-address6 (*IPv6 address*; Default:)	Source IPv6 address (only if MAC protocol is set to IPv6).
src-mac-address (*MAC address*; Default:)	Source MAC address.
src-port (*integer 0..65535*; Default:)	Source port number or range (only for TCP or UDP protocols).
stp-flags (*topology-change* / *topology-change-ack*; Default:)	The BPDU (Bridge Protocol Data Unit) flags. Bridge exchange configuration messages named BPDU periodically for preventing loops • topology-change - topology change flag is set when a bridge detects port state change, to force all other bridges to drop their host tables and recalculate network topology • topology-change-ack - topology change acknowledgment flag is sent in replies to the notification packets
stp-forward-delay (*integer 0..65535*; Default:)	Forward delay timer.
stp-hello-time (*integer 0..65535*; Default:)	STP hello packets time.
stp-max-age (*integer 0..65535*; Default:)	Maximal STP message age.
stp-msg-age (*integer 0..65535*; Default:)	STP message age.
stp-port (*integer 0..65535*; Default:)	STP port identifier.
stp-root-address (*MAC address*; Default:)	Root bridge MAC address.
stp-root-cost (*integer 0..65535*; Default:)	Root bridge cost.
stp-root-priority (*integer 0..65535*; Default:)	Root bridge priority.
stp-sender-address (*MAC address*; Default:)	STP message sender MAC address.
stp-sender-priority (*integer 0..65535*; Default:)	STP sender priority.
stp-type (*config* / *tcn*; Default:)	The BPDU type: • config - configuration BPDU • tcn - topology change notification
tls-host (*string*; Default:)	Allows matching https traffic based on TLS SNI hostname. Accepts GLOB syntax for wildcard matching. Note that matcher will not be able to match hostname if the TLS handshake frame is fragmented into multiple TCP segments (packets).
vlan-encap (*802.2* / *arp* / *ip* / *ipv6* / *ipx* / *length* / *mpls-multicast* / *mpls-unicast* / *pppoe* / *pppoe-discovery* / *rarp* / *vlan* / *integer 0..65535* / *hex 0x0000-0xffff*; Default:)	Matches the MAC protocol type encapsulated in the VLAN frame.
vlan-id (*integer 0..4095*; Default:)	Matches the VLAN identifier field.
vlan-priority (*integer 0..7*; Default:)	Matches the VLAN priority (priority code point)

Footnotes:

- STP matchers are only valid if the destination MAC address is `01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF` (Bridge Group address), also STP should be enabled.

- ARP matchers are only valid if mac-protocol is `arp` or `rarp`
- VLAN matchers are only valid for `0x8100` or `0x88a8` ethernet protocols
- IP or IPv6 related matchers are only valid if mac-protocol is either set to `ip` or `ipv6`
- 802.3 matchers are only consulted if the actual frame is compliant with IEEE 802.2 and IEEE 802.3 standards. These matchers are ignored for other packets.

Bridge Packet Filter

This section describes specific bridge filter options.

Sub-menu: `/interface bridge filter`

Property	Description
action (*accept* | *drop* | *jump* | *log* | *mark-packet* | *passthrough* | *return* | *set-priority*; Default: **accept**)	Action to take if the packet is matched by the rule: - accept - accept the packet. No action, i.e., the packet is passed through without undertaking any action, and no more rules are processed in the relevant list/chain - drop - silently drop the packet (without sending the ICMP reject message) - jump - jump to the chain specified by the value of the jump-target argument - log - add a message to the system log containing the following data: in-interface, out-interface, src-mac, dst-mac, eth-proto, protocol, src-ip:port->dst-ip:port and length of the packet. After packet is matched it is passed to the next rule in the list, similar as passthrough - mark - mark the packet to use the mark later - passthrough - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for the ability to count packets - return - return to the previous chain, from where the jump took place - set-priority - set priority specified by the new-priority parameter on the packets sent out through a link that is capable of transporting priority (VLAN or WMM-enabled wireless interface). Read more

Bridge NAT

This section describes specific bridge NAT options.

Sub-menu: `/interface bridge nat`

Property	Description
action (*accept* | *drop* | *jump* | *mark-packet* | *redirect* | *set-priority* | *arp-reply* | *dst-nat* | *log* | *passthrough* | *return* | *src-nat*; Default: **accept**)	Action to take if the packet is matched by the rule: - accept - accept the packet. No action, i.e., the packet is passed through without undertaking any action, and no more rules are processed in the relevant list/chain - arp-reply - send a reply to an ARP request (any other packets will be ignored by this rule) with the specified MAC address (only valid in dstnat chain) - drop - silently drop the packet (without sending the ICMP reject message) - dst-nat - change destination MAC address of a packet (only valid in dstnat chain) - jump - jump to the chain specified by the value of the jump-target argument - log - log the packet - mark - mark the packet to use the mark later - passthrough - ignore this rule and go on to the next one. Acts the same way as a disabled rule, except for the ability to count packets - redirect - redirect the packet to the bridge itself (only valid in dstnat chain) - return - return to the previous chain, from where the jump took place - set-priority - set priority specified by the new-priority parameter on the packets sent out through a link that is capable of transporting priority (VLAN or WMM-enabled wireless interface). Read more - src-nat - change source MAC address of a packet (only valid in srcnat chain)
to-arp-reply-mac-address (*MAC address*; Default:)	Source MAC address to put in Ethernet frame and ARP payload, when `action=arp-reply` is selected

to-dst-mac-address (*MAC address*; Default:)	Destination MAC address to put in Ethernet frames, when `action=dst-nat` is selected
to-src-mac-address (*MAC address*; Default:)	Source MAC address to put in Ethernet frames, when `action=src-nat` is selected

See also

- CRS1xx/2xx series switches
- CRS3xx, CRS5xx series switches, and CCR2116, CCR2216 routers
- Switch chip features
- MTU on RouterBOARD
- Layer2 misconfiguration
- Bridge VLAN Table
- Wireless VLAN Trunk
- VLANs on Wireless