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 (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.

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 "port-number", and it can change after a reboot. To avoid unwanted MAC address changes, it is recommended to disable "auto-mac", and to manually specify MAC by using "admin-mac".

### Sub-menu: /interface bridge

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add-dhcp-option82 (yes</td>
<td>no; Default: no)</td>
</tr>
<tr>
<td>admin-mac (MAC address; Default: no)</td>
<td>Static MAC address of the bridge. This property only has an effect when auto-mac is set to no.</td>
</tr>
<tr>
<td>ageing-time (time; Default: 0:00:00)</td>
<td>How long a host's information will be kept in the bridge database.</td>
</tr>
</tbody>
</table>
| 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 reply to requests originated from matching IP address/MAC address combinations which 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. |
<p>| arp-timeout (auto | integer; Default: auto) | How long the ARP record is kept in the ARP table after no packets are received from IP. 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 (0x8100 | 0x8888; 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. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>igmp-version</strong></td>
<td>Selects the IGMP version in which IGMP general membership queries will be generated. This property only has an effect when <strong>igmp-snooping</strong> is set to yes.</td>
</tr>
<tr>
<td><strong>ingress-filtering</strong></td>
<td>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 <strong>frame-types</strong> to specify if the ingress traffic should be tagged or untagged. This property only has an effect when <strong>vlan-filtering</strong> is set to yes.</td>
</tr>
<tr>
<td><strong>l2mtu</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>last-member-interval</strong></td>
<td>If a port has <strong>fast-leave</strong> set to no and a bridge port receives an IGMP Leave message, then IGMP Snooping enabled bridge will send an IGMP query to make sure that no devices have subscribed to a certain multicast stream on a bridge port. If an IGMP Snooping enabled bridge does not receive an IGMP membership report after the amount of <strong>last-member-interval</strong>, then the bridge considers that no one has subscribed to a certain multicast stream and can stop forwarding it. This property only has an effect when <strong>igmp-snooping</strong> is set to yes.</td>
</tr>
<tr>
<td><strong>last-member-query-count</strong></td>
<td>How many times should <strong>last-member-interval</strong> pass until the IGMP Snooping bridge will stop forwarding a certain multicast stream. This property only has an effect when <strong>igmp-snooping</strong> is set to yes.</td>
</tr>
<tr>
<td><strong>max-hops</strong></td>
<td>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 <strong>protocol-mode</strong> is set to <strong>mstp</strong>.</td>
</tr>
<tr>
<td><strong>max-message-age</strong></td>
<td>How long to remember Hello messages received from other STP/RSTP enabled bridges. This property only has an effect when <strong>protocol-mode</strong> is set to <strong>stp</strong> or <strong>rstp</strong>.</td>
</tr>
<tr>
<td><strong>membership-interval</strong></td>
<td>The amount of time after an entry in the Multicast Database (MDB) is removed if an IGMP membership report is not received on a certain port. This property only has an effect when <strong>igmp-snooping</strong> is set to yes.</td>
</tr>
<tr>
<td><strong>mid-version</strong></td>
<td>Selects the MLD version in which MLD general membership queries will be generated. This property only has an effect when the RouterOS IPv6 package is enabled and <strong>igmp-snooping</strong> is set to yes.</td>
</tr>
<tr>
<td><strong>mtu</strong></td>
<td>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 <strong>mtu</strong> property), then manually set value will be ignored and the bridge will act as if <strong>mtu=auto</strong> is set.</td>
</tr>
<tr>
<td><strong>multicast-querier</strong></td>
<td>Multicast querier generates IGMP general membership queries to which all IGMP capable devices respond with an IGMP membership report. Usually a PIM (multicast) router or IGMP proxy generates these queries. When RouterOS IPv6 package is enabled, the bridge will also generate MLD general membership queries. By using this property you can make an IGMP 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 and IGMP Snooping will not function properly. Only untagged IGMP/MLD general membership queries are generated. This property only has an effect when <strong>igmp-snooping</strong> is set to yes. Additionally, the <strong>igmp-snooping</strong> should be disabled/enabled after changing <strong>multicast-querier</strong> property.</td>
</tr>
<tr>
<td><strong>multicast-router</strong></td>
<td>Changes the state of a bridge itself if IGMP membership reports are going to be forwarded to it. This property can be used to forward IGMP membership reports to the bridge for further multicast routing or proxying. This property only has an effect when <strong>igmp-snooping</strong> is set to yes.</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>Name of the bridge interface.</td>
</tr>
<tr>
<td><strong>priority</strong></td>
<td>Bridge priority, used by RSTP to determine root bridge, used by MSTP to determine CIST and IST regional root bridge. This property has no effect when <strong>protocol-mode</strong> is set to none.</td>
</tr>
<tr>
<td><strong>protocol-mode</strong></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 <strong>protocol-mode</strong> to none.</td>
</tr>
<tr>
<td><strong>vlan-filtering</strong></td>
<td>If a port has <strong>fast-leave</strong> set to no and a bridge port receives an IGMP Leave message, then IGMP Snooping enabled bridge will send an IGMP query to make sure that no devices have subscribed to a certain multicast stream on a bridge port. If an IGMP Snooping enabled bridge does not receive an IGMP membership report after the amount of <strong>last-member-interval</strong>, then the bridge considers that no one has subscribed to a certain multicast stream and can stop forwarding it. This property only has an effect when <strong>igmp-snooping</strong> is set to yes.</td>
</tr>
</tbody>
</table>
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 `vlan-filtering` is set to `yes`.

Used to change the interval of how often a bridge checks if it is the active multicast querier. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.

Used to change the interval of how often IGMP general membership queries are sent out. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.

Interval in which an IGMP capable device must reply to an IGMP query with an IGMP membership report. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.

MSTP region name. This property only has an effect when `protocol-mode` is set to `mstp`.

MSTP configuration revision number. This property only has an effect when `protocol-mode` is set to `mstp`.

Specifies how many times must `startup-query-interval` pass until the bridge starts sending out IGMP general membership queries periodically. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.

Used to change the amount of time after a bridge starts sending out IGMP general membership queries after the bridge is enabled. This property only has an effect when `igmp-snooping` and `multicast-querier` is set to `yes`.

The Transmit Hold Count used by the Port Transmit state machine to limit the transmission rate.

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:

```plaintext
[admin@MikroTik] > interface bridge add
[admin@MikroTik] > interface bridge print
Flags: X - disabled, R - running
0 R name="bridge1" mtu=auto actual-mtu=1500 12mtu=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`

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>current-mac-address</td>
<td>Current MAC address of the bridge</td>
</tr>
<tr>
<td>designated-port-count</td>
<td>Number of designated bridge ports</td>
</tr>
<tr>
<td>port-count</td>
<td>Number of the bridge ports</td>
</tr>
<tr>
<td>root-bridge</td>
<td>Shows whether the bridge is the root bridge of the spanning tree</td>
</tr>
<tr>
<td>root-bridge-id</td>
<td>The root bridge ID, which is in form of bridge-priority.bridge-MAC-address</td>
</tr>
</tbody>
</table>
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.

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 article. In RouterOS, the protocol-mode property controls the used STP variant.

---

### 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.

**Create edge ports**

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

```
[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
```

---

**Drop sent BPDUs**

In this example, BPDUs will not be sent out through `ether1`. In case the bridge is the root bridge, then loop detection will not work on this port. If another bridge is connected to `ether1`, then the other bridge will not receive any BPDUs and therefore it might become a second root bridge.

```
[admin@MikroTik] /interface bridge
add name=bridge1
/add interface bridge port
add bridge=bridge1 interface=ether1 edge=yes
add bridge=bridge1 interface=ether2
/add interface bridge filter
add action=drop chain=output dst-mac-address=01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF out-interface=ether1
```

---

**Drop received BPDUs**

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

---

**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.**

---

**Create edge ports**

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

```
[admin@MikroTik] /interface bridge
add name=bridge1
/add interface bridge port
add bridge=bridge1 interface=ether1 edge=yes
add bridge=bridge1 interface=ether2
```

---

**Drop sent BPDUs**

In this example, BPDUs will not be sent out through `ether1`. In case the bridge is the root bridge, then loop detection will not work on this port. If another bridge is connected to `ether1`, then the other bridge will not receive any BPDUs and therefore it might become a second root bridge.

```
[admin@MikroTik] /interface bridge
add name=bridge1
/add interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=ether2
/add interface bridge filter
add action=drop chain=output dst-mac-address=01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF out-interface=ether1
```

---

You can use Interface Lists to specify multiple interfaces.
Drop received BPDUs
The bridge filter input or NAT rules cannot drop received 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.

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

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`

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use-ip-firewall (yes / no)</td>
<td>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 <code>use-ip-firewall-for-pppoe</code> is required in case bridge <code>vlan-filtering</code> is used.</td>
</tr>
<tr>
<td>use-ip-firewall-for-pppoe (yes / no; Default: no)</td>
<td>Send bridged un-encrypted PPPoE traffic to also be processed by IP/Firewall. This property only has an effect when <code>use-ip-firewall</code> 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.</td>
</tr>
<tr>
<td>use-ip-firewall-for-vlan (yes / no; Default: no)</td>
<td>Send bridged VLAN traffic to also be processed by IP/Firewall. This property only has an effect when <code>use-ip-firewall</code> 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.</td>
</tr>
<tr>
<td>allow-fast-path (yes / no; Default: yes)</td>
<td>Whether to enable a bridge Fast Path globally.</td>
</tr>
<tr>
<td>bridge-fast-path-active (yes / no; Default: )</td>
<td>Shows whether a bridge FastPath is active globally, Fast Path status per bridge interface is not displayed.</td>
</tr>
<tr>
<td>bridge-fast-path-packets (integer; Default: )</td>
<td>Shows packet count forwarded by bridge Fast Path.</td>
</tr>
<tr>
<td>bridge-fast-path-bytes (integer; Default: )</td>
<td>Shows byte count forwarded by bridge Fast Path.</td>
</tr>
<tr>
<td>bridge-fast-forward-packets (integer; Default: )</td>
<td>Shows packet count forwarded by bridge Fast Forward.</td>
</tr>
</tbody>
</table>
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 submenu is used to add interfaces in a particular bridge.

**Sub-menu:** `/interface bridge port`

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bridge-fast-forward-bytes</code></td>
<td>Shows byte count forwarded by bridge Fast Forward.</td>
</tr>
<tr>
<td><code>auto-isolate</code></td>
<td>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 <code>protocol-mode</code> is set to <code>rstp</code> or <code>mstp</code> and <code>edge</code> is set to <code>no</code>.</td>
</tr>
<tr>
<td><code>bpdu-guard</code></td>
<td>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 <code>protocol-mode</code> is set to <code>none</code>.</td>
</tr>
<tr>
<td><code>bridge</code></td>
<td>The bridge interface where the respective interface is grouped in.</td>
</tr>
<tr>
<td><code>broadcast-flood</code></td>
<td>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 <code>FF:FF:FF:FF:FF:FF</code> 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.</td>
</tr>
<tr>
<td><code>edge</code></td>
<td>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 <code>protocol-mode</code> is set to <code>none</code>.</td>
</tr>
<tr>
<td><code>fast-leave</code></td>
<td>Enables IGMP Fast leave feature on the port. Bridge will stop forwarding traffic to a bridge port whenever an IGMP Leave message is received for appropriate multicast stream. This property only has an effect when <code>igmp-snooping</code> is set to <code>yes</code>.</td>
</tr>
<tr>
<td><code>frame-types</code></td>
<td>Specifies allowed ingress frame types on a bridge port. This property only has an effect when <code>vlan-filtering</code> is set to <code>yes</code>.</td>
</tr>
<tr>
<td><code>ingress-filtering</code></td>
<td>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 <code>frame-types</code> to specify if the ingress traffic should be tagged or untagged. This property only has an effect when <code>vlan-filtering</code> is set to <code>yes</code>.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>learn (auto / no / yes; Default: auto)</td>
<td>Changes MAC learning behavior on a bridge port</td>
</tr>
<tr>
<td>- yes - enables MAC learning</td>
<td></td>
</tr>
<tr>
<td>- no - disables MAC learning</td>
<td></td>
</tr>
<tr>
<td>- 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.</td>
<td></td>
</tr>
<tr>
<td>multicast-router (disabled / permanent / temporary-query; Default: temporary-query)</td>
<td>Changes the state of a bridge port whether IGMP membership reports are going to be forwarded to this port. By default IGMP membership reports (most importantly IGMP Join messages) are only forwarded to ports that have a multicast router or a IGMP Snooping enabled bridge connected to. Without at least one port marked as a multicast-router IPTV might not work properly, it can be either detected automatically or forced manually.</td>
</tr>
<tr>
<td>- disabled - IGMP membership reports are not forwarded through this port regardless what is connected to it.</td>
<td></td>
</tr>
<tr>
<td>- permanent - IGMP membership reports are forwarded through this port regardless what is connected to it.</td>
<td></td>
</tr>
<tr>
<td>- temporary-query - automatically detect multicast routers and IGMP Snooping enabled bridges.</td>
<td></td>
</tr>
<tr>
<td>horizon (integer; Default: none)</td>
<td>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.</td>
</tr>
<tr>
<td>interface (name; Default: none)</td>
<td>Name of the interface.</td>
</tr>
<tr>
<td>path-cost (integer; Default: 10)</td>
<td>Path cost to the interface, used by STP to determine the best path, used by MSTP to determine the best path between regions. This property has no effect when protocol-mode is set to none.</td>
</tr>
<tr>
<td>point-to-point (auto / yes / no; Default: auto)</td>
<td>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.</td>
</tr>
<tr>
<td>priority (integer; Default: 128)</td>
<td>The priority of the interface, used by STP to determine the root port, used by MSTP to determine root port between regions.</td>
</tr>
<tr>
<td>pvid (integer; Default: 1)</td>
<td>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.</td>
</tr>
<tr>
<td>restricted-role (yes / no; Default: no)</td>
<td>Enable the restricted role on a port, used by STP to forbid a port from becoming a root port. This property only has an effect when protocol-mode is set to mstp.</td>
</tr>
<tr>
<td>restricted-tcn (yes / no; Default: no)</td>
<td>Disable topology change notification (TCN) sending on a port, used by STP to forbid network topology changes to propagate. This property only has an effect when protocol-mode is set to mstp.</td>
</tr>
<tr>
<td>tag-stacking (yes / no; Default: no)</td>
<td>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 an effect when vlan-filtering is set to yes.</td>
</tr>
<tr>
<td>trusted (yes / no; Default: no)</td>
<td>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.</td>
</tr>
</tbody>
</table>
When enabled, the bridge floods unknown multicast traffic to all bridge egress ports. When disabled, drops unknown multicast traffic on egress ports. Multicast addresses that are in the MDB table are considered as learned multicasts and therefore will not be flooded to all ports. Without IGMP Snooping all multicast traffic will be dropped on egress ports. Has an effect only on an egress port. This option does not limit traffic flood to the CPU. Note that local multicast addresses (224.0.0.0/24) are not flooded when `unknown-multicast-flood` is disabled, as a result some protocols that rely on local multicast addresses might not work properly, such protocols are RIPv2m, OSPF, mDNS, VRRP and others. Some protocols do send an IGMP join request and therefore are compatible with IGMP Snooping. some OSPF implementations are compatible with RFC1584, RouterOS OSPF implementation is not compatible with IGMP Snooping. This property should only be used when igmp-snooping is set to yes.

When enabled, bridge floods unknown unicast traffic to all bridge egress ports. 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 learnt as soon as a packet on a bridge port is received, then 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. Has effect only on an egress port. This option does not limit traffic flood to the CPU.

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 1 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 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
```
Bridge Port Monitoring

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

**Sub-menu:** `/interface bridge port monitor`

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>edge-port (yes / no)</td>
<td>Whether the port is an edge port or not.</td>
</tr>
<tr>
<td>edge-port-discovery (yes / no)</td>
<td>Whether the port is set to automatically detect edge ports.</td>
</tr>
<tr>
<td>external-fdb (yes / no)</td>
<td>Whether the registration table is used instead of a forwarding database.</td>
</tr>
<tr>
<td>forwarding (yes / no)</td>
<td>Shows if the port is not blocked by (R/M)STP.</td>
</tr>
<tr>
<td>hw-offload-group (switchX)</td>
<td>Switch chip used by the port.</td>
</tr>
<tr>
<td>learning (yes / no)</td>
<td>Shows whether the port is capable of learning MAC addresses.</td>
</tr>
<tr>
<td>multicast-router (yes / no)</td>
<td>Shows if a multicast router is detected on the port.</td>
</tr>
<tr>
<td>port-number (integer 1..4095)</td>
<td>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.</td>
</tr>
<tr>
<td>point-to-point-port (yes / no)</td>
<td>Whether the port is connected to a bridge port using full-duplex (yes) or half-duplex (no).</td>
</tr>
<tr>
<td>role (designated</td>
<td>root port</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>sending-rstp (yes / no)</td>
<td>Whether the port is sending RSTP or MSTP BPDU types. A port will transit to STP type when RSTP/MSTP enabled port receives an STP BPDU.</td>
</tr>
<tr>
<td>status (in-bridge / inactive)</td>
<td>Port status:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
[admin@MikroTik] /interface bridge port monitor [find interface=ether1]
    interface: ether1
    status: in-bridge
    port-number: 1
    role: designated-port
    edge-port: yes
    edge-port-discovery: yes
    point-to-point-port: yes
    external-fdb: no
    sending-rstp: yes
    learning: yes
    forwarding: yes
```

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`

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>age (read-only: time)</td>
<td>The time since the last packet was received from the host. Appears only for dynamic, non-external, and non-local host entries</td>
</tr>
<tr>
<td>bridge (read-only: name)</td>
<td>The bridge the entry belongs to</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bridge (name; Default: none)</td>
<td>The bridge interface to which the MAC address is going to be assigned.</td>
</tr>
<tr>
<td>disabled (yes</td>
<td>no; Default: no)</td>
</tr>
<tr>
<td>interface (name; Default: none)</td>
<td>Name of the interface.</td>
</tr>
<tr>
<td>mac-address (MAC address; Default: )</td>
<td>MAC address that will be added to the host table statically.</td>
</tr>
<tr>
<td>vid (integer: 1..4094; Default: )</td>
<td>VLAN ID for the statically added MAC address entry.</td>
</tr>
</tbody>
</table>

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

```
/ interface bridge host
add bridge=bridge2 interface=ether2 mac-address=4C:5E:0C:4D:12:43
```

Since RouterOS v6.41 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, \textit{Spanning Tree Protocol}. More details about the outdated master-port property can be found in the \textit{Master-port} page.

When upgrading from previous versions (before RouterOS v6.41), the old master-port configuration is automatically converted to the new \textit{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 (-):

<table>
<thead>
<tr>
<th>RouterBoard[Switch Chip] Model</th>
<th>Features in Switch menu</th>
<th>Bridge STP/RSTP</th>
<th>Bridge MSTP</th>
<th>Bridge IGMP Snooping</th>
<th>Bridge DHCP Snooping</th>
<th>Bridge VLAN Filtering</th>
<th>Bonding</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRS3xx series</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>CRS1xx/CRS2xx series</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>
configuration is converted. For each page.

- A bridge will be master-port, that will property, for more details check the master-port

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.

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 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 series switches
- 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, that will temporarily disable all Ethernet ports that are on the switch chip for the settings to have an effect, this must be taken into account whenever changing properties on production environments. Such properties are DHCP Snooping, IGMP Snooping, VLAN filtering, L2MTU, Flow Control, and others (exact settings that can trigger a switch chip reset depends on the device's model).

Example

Port switching with bridge configuration and enabled hardware offloading since RouterOS v6.41:

```
/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:

<table>
<thead>
<tr>
<th>#</th>
<th>INTERFACE</th>
<th>BRIDGE</th>
<th>HW</th>
<th>PVID</th>
<th>PRIORITY</th>
<th>PATH-COST</th>
<th>INTERNAL-PATH-COST</th>
<th>HORIZON</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ether2</td>
<td>bridge1</td>
<td>yes</td>
<td>1</td>
<td>0x80</td>
<td>10</td>
<td>10</td>
<td>none</td>
</tr>
<tr>
<td>1</td>
<td>ether3</td>
<td>bridge1</td>
<td>yes</td>
<td>1</td>
<td>0x80</td>
<td>10</td>
<td>10</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>ether4</td>
<td>bridge1</td>
<td>yes</td>
<td>1</td>
<td>0x80</td>
<td>10</td>
<td>10</td>
<td>none</td>
</tr>
<tr>
<td>3</td>
<td>ether5</td>
<td>bridge1</td>
<td>yes</td>
<td>1</td>
<td>0x80</td>
<td>10</td>
<td>10</td>
<td>none</td>
</tr>
</tbody>
</table>

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, for more details check the Master-port page.
Bridge VLAN Filtering since RouterOS v6.41 provides VLAN aware Layer2 forwarding and VLAN tag modifications within the bridge. This set of features makes bridge operation more like a traditional Ethernet switch and allows to overcome 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, only CRS3xx series devices 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.

<table>
<thead>
<tr>
<th>Sub-menu: /interface bridge vlan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
<tr>
<td><strong>bridge (name, Default: none)</strong></td>
</tr>
<tr>
<td><strong>disabled (yes / no, Default: no)</strong></td>
</tr>
<tr>
<td><strong>tagged (interfaces, Default: none)</strong></td>
</tr>
<tr>
<td><strong>untagged (interfaces, Default: none)</strong></td>
</tr>
<tr>
<td><strong>vlan-ids (integer 1..4094, Default: 0)</strong></td>
</tr>
</tbody>
</table>

- 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 trunk 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 (or switch-cpu port on CRS3xx series switch), 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 Management port section.

- 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 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).

```plaintext
[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   AGE
0   CC:2D:E0:E4:B3:AA  300 ether3       bridge1 33s
1   CC:2D:E0:E4:B3:AB  400 ether4       bridge1 33s
```

**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.

```
/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.

```
/interface bridge port
  add bridge=bridge1 interface=ether2
  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 and untagged ports in them.

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

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

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

**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.

```
/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.

```
/interface bridge port
  add bridge=bridge1 interface=ether2
  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 and untagged ports in them. In this example egress VLAN tagging is done on ether6, ether7, ether8 ports too, making them into hybrid ports.
In the end, when VLAN configuration is complete, enable Bridge VLAN Filtering.

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

You don't have to add access ports as untagged ports, 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:

```
/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:

```
/interface bridge port
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 and untagged ports in them. In this example `bridge1` interface is the VLAN trunk that will send traffic further to do InterVLAN routing:

```
/interface bridge vlan
add bridge=bridge1 tagged=ether6 untagged=ether8 vlan-ids=200
add bridge=bridge1 tagged=ether7 untagged=ether8 vlan-ids=300
add bridge=bridge1 tagged=ether8 untagged=ether6 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 bridgel vlan-filtering=yes
```

**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
```

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
```

In case VLAN filtering is used and access from the trunk and/or access ports with tagged traffic is desired, additional steps are required. In this example VLAN99 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 router/switch from access ports `ether3`, `ether4`, and from trunk port `sfp-sfpplus1`, then you must add this entry to the VLAN table:

```
/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 bridgel vlan-filtering=yes
```

In case VLAN filtering is used and access from trunk and/or access ports with untagged traffic is desired, you need to allow untagged traffic to access the router/switch. Start by creating an IP address on the bridge interface.

```
/ip address
add address=192.168.88.1/24 interface=bridge1
```

It is required to add VLAN 1 to ports from which you want to allow access to the router/switch, for example, to allow access from access ports `ether3, ether4` add this entry to the VLAN table:

```
/interface bridge vlan
add bridge=bridge1 untagged=ether3,ether4 vlan-ids=1
```

Make sure that PVID on the bridge interface matches the PVID value on these ports:
/interface bridge set bridge1 pvid=1
/interface bridge port set ether3,ether4 pvid=1

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**:

![VLAN Tunneling Diagram]

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 802.1ad VLAN protocol on the bridge, use these commands on SW1 and SW2:

```plaintext
/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:

```plaintext
/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:

```plaintext
/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:

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

In this example, ether3 and ether4 are going to be trunk ports, use the pvid parameter to tag all ingress traffic on each port, use these commands on SW1 and SW2:

```plaintext
/interface bridge port
add interface=ether3 bridge=bridge1 pvid=1
add interface=ether4 bridge=bridge1 pvid=1
```

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

```plaintext
/interface bridge vlan
add bridge=bridge1 tagged=ether1,ether2 untagged=ether4,ether3 vlan-ids=1
add bridge=bridge1 tagged=ether1,ether2 untagged=ether3,ether4 vlan-ids=1
```

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

```plaintext
/interface bridge set bridge1 vlan-filtering=yes
```
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.
```
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:

```bash
[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
```

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

```bash
[admin@MikroTik] /interface bridge monitor bridge1
state: enabled
root-bridge: yes
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.

Starting from RouterOS version 6.41, the bridge supports IGMP 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 CRS1xx/2xx/3xx series switches also support IGMP Snooping with hardware offloading.

To enable IGMP snooping on a bridge, set `igmp-snooping` to **yes**.

```bash
/interface bridge set bridge1 igmp-snooping=yes
```

To monitoring multicast groups, use `print` command in the `'/interface bridge mdb'` menu.
To monitoring ports that are connected to a multicast router, use `monitor` command in the '/interface bridge port' menu.

Starting from RouterOS version 6.43, the bridge supports DHCP Snooping and DHCP Option 82. 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:

```
[admin@MikroTik] /interface bridge add name=bridge
[admin@MikroTik] /interface bridge port add bridge=bridge interface=ether1 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 ether2 to the same bridge and leave this port untrusted, imagine there is an unauthorized (rogue) DHCP server. Use these commands on SW2:

```
[admin@MikroTik] /interface bridge add name=bridge
[admin@MikroTik] /interface bridge port add bridge=bridge interface=ether1 trusted=yes
```

**IGMP membership reports are only forwarded to ports that are connected to a multicast router or to another IGMP Snooping enabled bridge. If no port is marked as a multicast-router then IGMP membership reports will not be forwarded to any port.**
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.

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>802.3-sap (integer; Default: )</td>
<td>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.</td>
</tr>
<tr>
<td>802.3-type (integer; Default: )</td>
<td>Ethernet protocol type, placed after the IEEE 802.3 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 0x808B.</td>
</tr>
</tbody>
</table>

**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-addr, 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>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arp-dst-address</code></td>
<td>(IP address; Default: ) ARP destination IP address.</td>
</tr>
<tr>
<td><code>arp-dst-mac-address</code></td>
<td>(MAC address; Default: ) ARP destination MAC address.</td>
</tr>
<tr>
<td><code>arp-gratuitous</code></td>
<td>(yes / no; Default: ) Matches ARP gratuitous packets.</td>
</tr>
<tr>
<td><code>arp-hardware-type</code></td>
<td>(integer; Default: 1) ARP hardware type. This is normally Ethernet (Type 1).</td>
</tr>
<tr>
<td><code>arp-opcode</code></td>
<td>(arp-nak</td>
</tr>
<tr>
<td><code>arp-packet-type</code></td>
<td>(integer 0..65535</td>
</tr>
<tr>
<td><code>arp-src-address</code></td>
<td>(IP address; Default: ) ARP source IP address.</td>
</tr>
<tr>
<td><code>arp-src-mac-address</code></td>
<td>(MAC address; Default: ) ARP source MAC address.</td>
</tr>
<tr>
<td><code>chain</code></td>
<td>(text; Default: ) Bridge firewall chain, which the filter is functioning in (either a built-in one, or a user-defined one).</td>
</tr>
<tr>
<td><code>dst-address</code></td>
<td>(IP address; Default: ) Destination IP address (only if MAC protocol is set to IP).</td>
</tr>
<tr>
<td><code>dst-mac-address</code></td>
<td>(MAC address; Default: ) Destination MAC address.</td>
</tr>
<tr>
<td><code>dst-port</code></td>
<td>(integer 0..65535; Default: ) Destination port number or range (only for TCP or UDP protocols).</td>
</tr>
<tr>
<td><code>in-bridge</code></td>
<td>(name; Default: ) Bridge interface through which the packet is coming in.</td>
</tr>
<tr>
<td><code>in-interface</code></td>
<td>(name; Default: ) Physical interface (i.e., bridge port) through which the packet is coming in.</td>
</tr>
<tr>
<td><code>in-interface-list</code></td>
<td>(name; Default: ) Set of interfaces defined in interface list. Works the same as in-interface.</td>
</tr>
<tr>
<td><code>ingress-priority</code></td>
<td>(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»</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ip-protocol</td>
<td>Specifies the IP protocol layer to be matched.</td>
</tr>
<tr>
<td>limit</td>
<td>Restricts packet match rate to a given limit.</td>
</tr>
<tr>
<td>log-prefix</td>
<td>Defines the prefix to be printed before the logging information.</td>
</tr>
<tr>
<td>mac-protocol</td>
<td>Ethernet payload type (MAC-level protocol). To match protocol type for VLAN encapsulated frames (0x8100 or 0x88a8), a vlan-encap property should be used.</td>
</tr>
<tr>
<td>out-bridge</td>
<td>Outgoing bridge interface.</td>
</tr>
<tr>
<td>out-interface</td>
<td>Interface that the packet is leaving the bridge through.</td>
</tr>
<tr>
<td>out-interface-list</td>
<td>Set of interfaces defined in interface list. Works the same as out-interface.</td>
</tr>
</tbody>
</table>

**IP protocol (only if MAC protocol is set to IPv4)**

- dcpp - 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
- idrp-cmp - 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
- rsf - Radio Shortest Path First
- rsvp - Reservation Protocol
- sctp - Stream Control Transmission Protocol
- tcp - Transmission Control Protocol
- udp - User Datagram Protocol
- udp-lite - Lightweight User Datagram Protocol
- vmp - Versatile Message Transaction Protocol
- vmtp - Virtual Router Redundancy Protocol
- xns-idp - Xerox Network Systems Internet Datagram Protocol
- xtp - Xpress Transport Protocol

**Note:**
- If `action` is `jump`, specified, then specifies the user-defined firewall chain to process the packet.
- `limit` restricts packet match rate to a given limit.
- `log-prefix` defines the prefix to be printed before the logging information.
- `mac-protocol` specifies the MAC-level protocol. Ethernet payload type (MAC-level protocol). To match protocol type for VLAN encapsulated frames (0x8100 or 0x88a8), a vlan-encap property should be used.
- `out-bridge` specifies the outgoing bridge interface.
- `out-interface` specifies the interface that the packet is leaving the bridge through.
- `out-interface-list` sets the list of interfaces defined in `interface list`. Works the same as `out-interface`. 
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packet-mark</td>
<td>(name; Default:)  Match packets with a certain packet mark.</td>
</tr>
</tbody>
</table>
| 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-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.  
|                     |  - 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 | mple-multicast | mple-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 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
### Bridge filter

This section describes specific bridge filter options.

**Sub-menu:** `/interface bridge filter`

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>action</strong></td>
<td>(accept</td>
</tr>
</tbody>
</table>
| **priority** | 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>  
| CRS1xx/2xx series switches  
Swith chip features  
MTU on RouterBOARD  
Layer2 misconfiguration  
Bridge VLAN Table  
Wireless VLAN Trunk  
VLANs on Wireless |