## NAME

**vlan -** IEEE 802.1Q VLAN network interface

## SYNOPSIS

To compile this driver into the kernel, place the following line in your kernel configuration file:

```
device vlan
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
if_vlan_load="YES"
```

## DESCRIPTION

The **vlan** driver demultiplexes frames tagged according to the IEEE 802.1Q standard into logical **vlan** network interfaces, which allows routing/bridging between multiple VLANs through a single switch trunk port.

Each **vlan** interface is created at runtime using interface cloning. This is most easily done with the ifconfig(8) **create** command or using the *cloned_interfaces* variable in rc.conf(5).

To function, a **vlan** interface must be assigned a parent interface and numeric VLAN tag using ifconfig(8). A single parent can be assigned to multiple **vlan** interfaces provided they have different tags. The parent interface is likely to be an Ethernet card connected to a properly configured switch port. The VLAN tag should match one of those set up in the switched network.

**vlan** initially assumes the same minimum length for tagged and untagged frames. This mode is selected by setting the sysctl(8) variable *net.link.vlan.soft_pad* to 0 (default). However, there are network devices that fail to adjust frame length when it falls below the allowed minimum due to untagging. Such devices should be able to interoperate with **vlan** after changing the value of *net.link.vlan.soft_pad* to 1. In the latter mode, **vlan** will pad short frames before tagging them so that their length is not less than the minimum value after untagging by the non-compliant devices.

## HARDWARE

The **vlan** driver supports efficient operation over parent interfaces that can provide help in processing VLANs. Such interfaces are automatically recognized by their capabilities. Depending on the level of sophistication found in a physical interface, it may do full VLAN processing or just be able to receive and transmit long frames (up to 1522 bytes including an Ethernet header and FCS). The capabilities may be user-controlled by the respective parameters to ifconfig(8), **vlanhwtag**, and **vlanmtu**. However, a physical interface is not obliged to react to them: It may have either capability enabled permanently without a way to turn it off. The whole issue is very specific to a particular device and its driver.

At present, these devices are capable of full VLAN processing in hardware: ae(4), age(4), alc(4), ale(4), bce(4), bge(4), bxe(4), cxgb(4), cxgbe(4), em(4), igb(4), ixgbe(4), jme(4), liquidio(4), msk(4), mxge(4), nge(4), re(4), sge(4), stge(4), ti(4), and vge(4).

Other Ethernet interfaces can run VLANs using software emulation in the **vlan** driver. However, some lack the capability of transmitting and receiving long frames. Assigning such an interface as the parent to **vlan** will result in a reduced MTU on the corresponding **vlan** interfaces. In the modern Internet, this is likely to cause tcp(4) connectivity problems due to massive, inadequate icmp(4) filtering that breaks the Path MTU Discovery mechanism.

These interfaces natively support long frames for **vlan**: axe(4), bfe(4), cas(4), dc(4), et(4), fwe(4), fxp(4), gem(4), le(4), nfe(4), rl(4), sis(4), sk(4), ste(4), vr(4), vte(4), and xl(4).

The **vlan** driver automatically recognizes devices that natively support long frames for **vlan** use and calculates the appropriate frame MTU based on the capabilities of the parent interface. Some other interfaces not listed above may handle long frames, but they do not advertise this ability. The MTU setting on **vlan** can be corrected manually if used in conjunction with such a parent interface.

## SEE ALSO

ifconfig(8), sysctl(8)