

NAME

bridge-utils-interfaces – bridge-utils extensions for the **interfaces(5)** file format

DESCRIPTION

`/etc/network/interfaces` contains network interface configuration information for the **ifup(8)** and **ifdown(8)** commands. This manpage describes the bridge extensions to the standard **interfaces(5)** file format.

The main extension is the `bridge_ports` option, with it you describe that the interface is a bridge and what ports does it have. These ports are the interfaces that are part of the bridge, and they shouldn't have any stanzas defining them on the interfaces file. Other extensions allow you to tune the bridge options or change a bridge behaviour.

We'll see this with an example:

```
auto br0
iface br0 inet static
    address 192.168.1.2
    network 192.168.1.0
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.1
    bridge_ports all
```

Well, after setting this, an `ifup br0`, or the next reboot, should let you have a bridge up and running, after waiting for the ports to get to the forwarding status, of course. This bridge will be using all your `ethX` interfaces, as we have stated on the `bridge_ports` line.

The Debian bridge setup scripts will wait for it to get ready to work. They do this by trying to guess the maximum time that the bridge will need to get to the forwarding status, and by default, they will wait for the bridge to get there, or for the estimated maximum time to go by. This is done so that the services that are loaded after the bridge setup have a working network interface and don't fail because the bridge is still not working. See `bridge_maxwait` if you want to change this behaviour.

An example of how to setup a so called anonymous bridge (a bridge without an assigned IP) would look like this:

```
iface br1 inet manual
    bridge_ports eth1 eth2
    bridge_maxwait 0
```

Here we select the interfaces `eth1` and `eth2` to be added to the bridge interface `br1`, which will be an anonymous bridge, we also tell the scripts not to wait, as we won't be having any service running on that interface (it doesn't even have an IP).

An example of a little more complex bridge setup could be:

```
auto br0
iface br0 inet static
    address 192.168.1.2
    network 192.168.1.0
    netmask 255.255.255.0
    broadcast 192.168.1.255
    bridge_ports all weird0
    bridge_bridgeprio 32767
    bridge_portprio eth0 129
    bridge_fd 5
    pre-up ip addr flush dev eth0
```

In this example we select all the `eth*` devices plus a `weird` device to be added to the bridge, also we change the bridge default priority to a higher one so that this bridge becomes the root (if there are no bridges with higher priority on the net, that is) and also we lower priority of port `eth0` so that it is not used if there are other ports with higher priority to reach the same destination, at the end we lower the default forward delay and we add a pre-up command to remove all addresses on `eth0` as this interface had an address set up before (needed only on weird/broken setups).

If there is a need to set up any of the interfaces participating on the bridge and not the bridge itself, then we must add the commands to set up those settings in a "pre-up" or "up" statement. This means that if we have a wireless card that we want to add to a bridge and we want to set it to master, and select the essid, instead of using the typical `wireless_*` commands we could add to the bridge definition something like this:

```
pre-up iwconfig wlan0 mode master essid myESSID
```

Be aware, however, that using wireless cards as part of a bridge is not a good idea if the card belonging to the bridge is in managed mode. Trying to bridge packets coming out of our LAN through a wireless card that is set in managed mode (the card is a client of an AP) is bound to give problems, as the AP will probably refuse packets with source MAC addresses which are not associated (this will be the case of other machines going through the wireless card of the bridge into the AP).

Multiple stanzas of a bridge definition are currently not supported, so if you want to add a ipv6 and a ipv4 to a bridge do it all in one definition by using the "up" option. If however you still want to use multiple stanzas or would like to read more on this bug you can see it at <http://bugs.debian.org/319832>

IFACE OPTIONS

A little explanation on the new options that can be used on `/etc/network/interfaces` to setup the bridge, so you don't have to go and look at the scripts...

bridge_ports *interface specification*

this option must exist for the scripts to setup the bridge, with it you specify the ports you want to add to your bridge, either using "none" if you want a bridge without any interfaces or you want to add them later using `brctl`, or a list of the interfaces you want to add separated by spaces, for example:

```
bridge_ports eth0 eth4
```

You should not put any lines to configure the interfaces that will be used by the bridge, as this will be setup automatically by the scripts when bringing the bridge up.

If you need to specify the interfaces more flexibly, you can use the following syntax (most useful on a Xen dom0):

```
bridge_ports regex (eth|vif).*
```

This means to evaluate (as in `egrep(1)`) the expressions that follow after "regex" until either the end or a "noregex" statement is reached. The regular expressions are evaluated against all local interfaces and those that match are added.

Specifying "all" is short for "regex eth.* em.* p[0-9].* noregex" and will get all the ethX and biosdevname-format (emX and pX) interfaces added to the bridge.

Carrying this to the extremes, the following is valid syntax:

```
bridge_ports all regex if.0 noregex ext0 regex vif.*
```

This will add all ethX interfaces, the ifX0 interfaces, the ext0 interface and all vifX interfaces.

bridge_ageing *time*

set ageing time, default is 300, can have a fractional part.

bridge_bridgeprio *priority*

set bridge priority, *priority* is between 0 and 65535, default is 32768, affects bridge id, lowest priority bridge will be the root.

bridge_fd *time*

set bridge forward delay to *time* seconds, default is 15, can have a fractional part.

bridge_gcint *time*

set garbage collection interval to *time* seconds, default is 4, can have a fractional part. Available on Linux kernel versions < 2.6.0.

bridge_hello *time*

set hello time to *time* seconds, default is 2, can have a fractional part.

bridge_hw *MAC address*

set the Ethernet MAC address of the bridge to the specified one. There were some concerns of how this was done in the past, see: <http://bugs.debian.org/271406> but we are doing it on a new way now that shouldn't be as bad, see: <http://bugs.debian.org/725786> however you should know what you are doing before using this option.

bridge_maxage *time*

set max message age to *time* seconds, default is 20, can have a fractional part.

bridge_maxwait *time*

forces to *time* seconds the maximum time that the Debian bridge setup scripts will wait for the bridge ports to get to the forwarding status, doesn't allow fractional part. If it is equal to 0 then no waiting is done.

bridge_pathcost *port cost*

set path cost for a port, default is 100, *port* is the name of the interface to which this setting applies.

bridge_portprio *port priority*

set port priority, default is 32, affects port id, *port* is the name of the interface to which this setting applies. On Linux kernels older than 2.6.0 the max value is 255, the default 128. Newer kernels have a maximum value of 63 and a default of 32.

bridge_stp *state*

turn spanning tree protocol on/off, *state* values are on or yes to turn stp on and any other thing to set it off, default has changed to off for security reasons in latest kernels, so you should specify if you want stp on or off with this option, and not rely on your kernel's default behaviour.

bridge_waitport *time [ports]*

wait for a max of *time* seconds for the specified *ports* to become available, if no ports are specified then those specified on `bridge_ports` will be used here. Specifying no ports here should not be used if we are using `regex` or "all" on `bridge_ports`, as it wouldn't work.

FILES

/etc/network/interfaces

definitions of network interfaces See **interfaces(5)** for more information.

KNOWN BUGS/LIMITATIONS

The default values shown here are current values as of this writing, but as they are not related to this package but to the bridge code itself, can change anytime.

AUTHOR

This manpage was written by Santiago Garcia Mantinan <manty@debian.org> based on *interfaces(5)*.

SEE ALSO

brctl(8), **interfaces(5)**, **ifup(8)**, **iwconfig(8)**, **run-parts(8)**.