

---

# Libvirt client IPv6

## Overview

In **Debian** default Libvirt installation IPv6 is not configured. To enable IPv6 changes need to be made in 2 areas:

1. Virtual Network setup `virbr0`
2. Network filter (`nwfilter`)

## Virtual Network setup `virbr0`

The default virtual network can be inspected and changed in the Virtual Machine Manager (Edit → Connection Details → Virtual Networks) or via `virsh`.

### Default Virtual Network Configuration

Inspect the current/default configuration with the command below. As you can see there are no entries related to IPv6 included.

```
# virsh net-dumpxml default
-----
<network>
  <name>default</name>
  <uuid>b7dc4445-2ab9-4ad0-a48f-bd8aec3a99ec</uuid>
  <forward mode="nat">
    <nat>
      <port start="1024" end="65535"/>
    </nat>
  </forward>
  <bridge name="virbr0" stp="on" delay="0"/>
  <mac address="52:54:00:f5:c4:41"/>
  <ip address="192.168.122.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.122.2" end="192.168.122.254"/>
    </dhcp>
  </ip>
</network>
```

### Edit Virtual Network Configuration

To edit and update the configuration with the following commands, where `default` is the name of the only/standard configuration:

```
# virsh net-destroy default
# virsh net-edit default
# virsh net-start default
# systemctl restart libvirtd (** optional ** to restart whole libvirt)
```

With `virsh net-edit default` apply the following configuration:

```
# virsh net-dumpxml default
<network connections='1'>
  <name>default</name>
  <uuid>b7dc4445-2ab9-4ad0-a48f-bd8aec3a99ec</uuid>
  <forward mode='nat'>
    <nat ipv6='yes'>
      <port start='1024' end='65535' />
    </nat>
  </forward>
  <bridge name='virbr0' stp='on' delay='0' />
  <mac address='52:54:00:f5:c4:41' />
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254' />
    </dhcp>
  </ip>
  <ip family='ipv6' address='2001:1c00:2e07:fa0a:aaaa::1' prefix='80'>
    <dhcp>
      <range start='2001:1c00:2e07:fa0a:aaaa::1000'
end='2001:1c00:2e07:fa0a:aaaa::1100' />
    </dhcp>
  </ip>
</network>
```

## Network filter (nfilter)

In **Debian**, when the default configuration has applied the `clean-traffic` network filter (`nfilter`) to VM network interface, **all IPv6 traffic is completely blocked by default**. These network filters define the `nft` firewall setting that the host apply on the `virbr0` bridge interface.

The `clean-traffic` filter is designed to prevent IP/MAC spoofing by only allowing IPv4 traffic matching the VM's assigned IP address. To use IPv6 without removing this security filter, you must explicitly chain or include the `allow-ipv6` sub-filter.

Here is how to properly configure it.

### Step 1: Ensure the Filter Configurations Exist

On Debian, these default filters are provided by the `libvirt-daemon-config-nwfilter` package. Check if they are available in your system's network filters:

```
virsh nwfilter-list
```

You should see both `clean-traffic` and `allow-ipv6` in the output.

## Step 2: Check active filter in your Virtual Machine's XML Configuration

Open your VM configuration for editing (replace your `-vm-name` with your actual VM's name). Locate your network `<interface>` block. It will probably lists `clean-traffic`.

```
# virsh edit your-vm-name
```

Example Configuration looks like:

```
<interface type='bridge'>
  <mac address='52:54:00:12:34:56' />
  <source bridge='br0' />
  <model type='virtio' />
  <filterref filter='clean-traffic' />
</interface>
```

## Step 2: Edit your Virtual Machine's XML Configuration

You need to add the `allow-ipv6` filter reference inside the interface definition of your domain XML.

1. Open your VM configuration for editing (replace your `-vm-name` with your actual VM's name):

```
```bash
virsh edit your-vm-name
```
```

2. Locate your network `<interface>` block. 3. If it only lists `clean-traffic`, modify it to include `allow-ipv6`.

### Example Configuration:

```
<interface type='bridge'>
  <mac address='52:54:00:12:34:56' />
  <source bridge='br0' />
  <model type='virtio' />
  <!-- Reference clean-traffic -->
  <filterref filter='clean-traffic' />
  <!-- Explicitly add the allow-ipv6 filter -->
  <filterref filter='allow-ipv6' />
```

```
</interface>
```

Alternative: Custom No-Spoofing Filter (Advanced) If you want strict anti-spoofing for both IPv4 and IPv6 (instead of a blanket “allow all IPv6” via allow-ipv6), you can combine clean-traffic with specific IPv6 variables, or reference a custom filter. [1] (<https://github.com/fsinf/libvirt-fsinf>)

If using the default allow-ipv6, keep in mind that it acts as an open pass for IPv6 traffic through that interface. If you require strict IPv6 source filtering, you must define the `<parameter name='IPV6' value='...'/>` inside the `<filterref>` block using an anti-spoofing filter like no-ipv6-spoofing. [1] (<https://libvirt.org/formatnwfilter.html>), [2] (<https://github.com/fsinf/libvirt-fsinf>)

Step 3: Apply Changes For the changes to take effect on the guest networking stack: [1] (<https://codeberg.org/fkooman/frkovm>) 1. Save and exit the editor. 2. Completely stop and start the VM (a warm reboot inside the guest is usually not enough to reload host-side filter rules):

```
virsh destroy your-vm-name virsh start your-vm-name
```

## virsh edit error

### virsh edit Debian13-ESP32-ARMHF

error: XML document failed to validate against schema: Unable to validate doc against /usr/share/libvirt/schemas/domain.rng Extra element devices in interleave Element domain failed to validate content

Failed. Try again? [y,n,i,f,?]:

This error happens because libvirt's XML parser is extremely strict about tag ordering and schema nesting. The specific message Extra element devices in interleave almost always means one of two things occurred during your edit: [1, 2]

1. Tag placement error: You accidentally pasted the `<filterref>` rules outside the closing `</devices>` tag or broke the hierarchy of the `<interface>` block.
2. Duplicate/Mismatched tags: A closing tag like `</interface>` or `</devices>` was deleted or duplicated by mistake. [3]

## How to Fix It Right Now

At the prompt Failed. Try again? [y,n,i,f,?]:, type y and press Enter to open the editor back up. [4] Carefully inspect your changes. The `<filterref>` elements must be placed inside the `<interface>` tag block, and the `<interface>` block must remain safely inside the `<devices>` block.

## Correct Structure:

```
<domain type='kvm'>
```

```
<!-- ... other configurations ... -->
```

```
<devices>
```

```
<!-- ... other devices like disks, controllers ... -->
```

```
<interface type='bridge'>
```

```
<mac address='52:54:00:12:34:56' />
```

```
<source bridge='br0' />
```

```
<model type='virtio' />
```

```
<!-- FILTERS GO HERE, INSIDE THE INTERFACE BLOCK -->
```

```
<filterref filter='clean-traffic' />
```

```
<filterref filter='allow-ipv6' />
```

```
</interface> <!-- This must close before any other device starts -->
```

```
</devices> <!-- This closes the entire devices section -->
```

```
</domain>
```

## Common Pitfalls to Check:

\* \* Did you place it after `</devices>`? If your network block was at the very bottom of the `<devices>` section, it is easy to accidentally paste your text after `</devices>`, which causes this exact validation error. [3] \* Did you accidentally close `<interface />` early? Ensure your opening interface tag does not end with a slash (like `<interface type='bridge' />`). It must be an open tag `<interface type='bridge'>` that is explicitly closed with `</interface>` at the end of the block. \*

## The Escape Hatch (If your XML looks completely fine)

If you look at the XML and everything is structurally perfect, libvirt might be choking on a different pre-existing element or space in a filepath somewhere else in your file. [4, 5, 6] You can force libvirt to ignore the validation schema and save your changes anyway by typing `i` at the prompt: [4, 7]

Failed. Try again? [y,n,i,f,?]: i

(Note: Only do this if you are certain your `<interface>` tags are typed and closed correctly, otherwise libvirt might drop the bad network configuration entirely.) [2] If you hit `y` and look at the file, feel free to paste your `<interface>` block here if you want me to spot the formatting error for you!

[1] <https://lists.libvirt.org> [2] <https://wiki.libvirt.org> [3] <https://lists.libvirt.org> [4] <https://bugzilla.redhat.com> [5] <https://forum.qubes-os.org> [6] <https://bugzilla.redhat.com> [7] <https://forum.qubes-os.org>

The problem is that the Libvirt Domain XML schema only allows exactly one `<filterref>` element per network interface. While a custom network filter file can have multiple `<filterref>` nodes chained inside it, a virtual machine's interface configuration cannot accept multiple parallel filter definitions. When you try to add a second `<filterref>`, the XML validation engine gets thrown off completely, loses track of its current element tree, and bubbles up that confusing Extra element devices in interleave schema error. [1, 2] You have two clean ways to fix this.

## Option 1: Combine filters into a custom file (Recommended & Cleanest)

Instead of forcing two filters onto the interface, create a custom network filter file on the host machine that explicitly chains both functions together.

1. Create a new custom filter file on your Debian host:

```
sudo nano /etc/libvirt/nwfilter/clean-traffic-ipv6.xml
```

2. Paste the following configuration, which cleanly references both default behaviors:

```
<filter name='clean-traffic-ipv6' chain='root'>
  <filterref filter='clean-traffic' />
  <filterref filter='allow-ipv6' />
</filter>
```

3. Define and register the new filter in libvirt:

```
virsh nwfilter-define /etc/libvirt/nwfilter/clean-traffic-ipv6.xml
```

4. Update your VM XML to reference this single combined filter via `virsh edit Debian13-ESP32-ARMHF`:

```
<interface type='network'>
  <mac address='52:54:00:7e:d3:ee' />
  <source network='default' />
  <model type='virtio' />
  <filterref filter='clean-traffic-ipv6' />
  <address type='pci' domain='0x0000' bus='0x01' slot='0x00'
function='0x0' />
</interface>
```

## Option 2: Force it past the validation schema

If you prefer not to create a custom file, you can instruct libvirt to completely bypass the structural validation check by choosing the “ignore validation” (i) option at the prompt:

Failed. Try again? [y,n,i,f,?]: i

Why this works: The underlying libvirt processing engine handles multiple filter references perfectly fine at runtime, but its domain.rng checking tool is too strict and doesn't know it. Forcing the save will let the rules process normally. (Note: The custom file layout in Option 1 is heavily preferred as it keeps your configuration safe from being rejected or stripped out during future automated modifications or virt-manager edits). Would you like to verify your operational IPv6 firewall rules on the host using ebtables or nftables once the filter is applied?

[1] <https://libvirt.org> [2] <https://unix.stackexchange.com>

To include strict anti-spoofing protection for both IPv4 and IPv6, you should replace `<filterref filter='allow-ipv6'/>` with `<filterref filter='no-ipv6-spoofing'/>`. Do not keep `allow-ipv6` in the file. The `allow-ipv6` filter acts as a blanket permit for all IPv6 traffic, which would completely override and disable the security rules created by `no-ipv6-spoofing`. Here is exactly how your `/etc/libvirt/nwfilter/clean-traffic-ipv6.xml` file should look:

```
<filter name='clean-traffic-ipv6' chain='root'>

<filterref filter='clean-traffic' />
<filterref filter='no-ipv6-spoofing' />

</filter>
```

## Important Requirement: Passing the IPv6 Parameter

The `no-ipv6-spoofing` filter requires libvirt to know exactly which IPv6 address belongs to the virtual machine. Unlike IPv4 (where libvirt can automatically learn the IP via DHCP snooping), you must explicitly pass the allowed IPv6 address as a parameter inside your VM's XML configuration. Update your VM interface via `virsh edit Debian13-ESP32-ARMHF` to look like this:

```
<interface type='network'>

<mac address='52:54:00:7e:d3:ee' />
<source network='default' />
<model type='virtio' />
<filterref filter='clean-traffic-ipv6'>
  <!-- Replace with the actual IPv6 address assigned to this VM -->
  <parameter name='IPV6' value='2001:db8:1::50' />
</filterref>
<address type='pci' domain='0x0000' bus='0x01' slot='0x00' function='0x0' />

</interface>
```

## How this functions at the host level:

\* `clean-traffic` builds filters to block any IPv4 or MAC spoofing attempts. \* `no-ipv6-spoofing` builds filters that drop any outgoing IPv6 packets whose source address does not match the exact IPV6

parameter value you provided.

Would you like to know how to specify multiple IPv6 addresses if your VM uses both a global unicast address (GUA) and a link-local address, or are you assigning a single static address?

From:

<https://wiki.oscardegroot.nl/> - **HomeWiki**

Permanent link:

<https://wiki.oscardegroot.nl/doku.php?id=linux:apps:kvm:client-ipv6&rev=1789505837>

Last update: **2026/09/15 20:57**

