Kernel Stack

This chapter contains the following topics:

About Kernel Stack

Kernel Stack (kstack) uses well known Linux APIs to manage the routes and front panel ports.

Open Containers, like the Guest Shell, are Linux environments that are decoupled from the host software. You can install or modify software within that environment without impacting the host software packages.

Guidelines and Limitations

- Guest shell and the host Bash Shell use Kernel Stack (kstack).
- The Guest Shell and the host Bash Shell start in the default network namespace.
  - Use the **setns** system call to access other network namespaces
  - The **nsenter** and **ip netns exec** utilities can be used to execute within the context of a different network namespace.
- The interface state may be read from `/proc/net/dev` or retrieved using other typical Linux utilities such as **ip**, **ifconfig**, or **netstat**. The counters are for packets that have initiated or terminated on the switch.
- Use **ethtool –S** to get extended statistics from the net devices, which include packets that are switched through the interface.
- You can run packet capture applications like **tcpdump** to capture packets that initiate or terminate on the switch.
- There is no support for networking state changes (interface creation or deletion, IP address configuration, MTU change, and so on) from the Guest Shell.

- IPv4 and IPv6 are supported.
- Raw PF_PACKET is supported.
- Only on stack (Netstack or kstack) at a time can use well-known ports (0-15000), regardless of the network namespace.
- There is no IP connectivity between applications using Nestack and applications running kstack on the same switch. This limitation holds true regardless of whether the kstack applications are being run from the host Bash Shell or within a container.
- Applications within the Guest Shell are not allowed to send packets directly over an Ethernet out-of-band channel (EOBC) interface to communicate with the line cards or standby Sup.
- The management interface (mgmt0) is represented as eth1 in the kernel netdevices.

Changing the Port Range

Netstack and kstack divide the port range between them. The default port ranges are as follows:

- Kstack—From 15001 through 58000
- Netstack—From 58001 through 65535

Note Ports within the range from 63536 through 65535 are reserved for NAT.

Procedure

	Command or Action	Purpose
Step 1	[**no**] **sockets local-port-range** *start-port end-port*	This command modifies the port range for kstack. This command does not modify the Netstack range.

Example

The following example sets the kstack port range:

```
switch# sockets local-port-range 15001 25000
```

What to do next

After you have entered the command, be aware of the following issues:

- Reload the switch after entering the command.
- Leave a minimum of 7000 ports unallocated which are used by Netstack.
- Specify the *start-port* as 15001 or the *end-port* as 65535 to avoid holes in the port range.

Netdevice Property Changes

Netdevices representing the front channel port interfaces are always in the ADMIN UP state. The final, effective state is determined by the link carrier state.

The following example shows the following interfaces in NX-OS, where eth1/17 is shown as **up** and eth1/1 is shown as **down**:

```
root@kstack-switch# sh int ethernet 1/17 brief
Eth1/17       --      eth  routed up      none                      1000(D) --


root@kstack-switch# sh int ethernet 1/1 brief
Eth1/1        --      eth  routed down    Link not connected        auto(D) --
```

The following example shows these same interfaces, but this time as shown in the Bash shell using the **ip link show** command:

```
bash-4.3# ip link show Eth1-17
49: Eth1-17: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 100
    link/ether 00:42:68:58:f8:eb brd ff:ff:ff:ff:ff:ff


bash-4.3# ip link show Eth1-1
33: Eth1-1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode
DEFAULT group default qlen 100
    link/ether 00:42:68:58:f8:eb brd ff:ff:ff:ff:ff:ff
```

In this example, Eth1-1 is shown as being **UP**, but is shown as **NO-CARRIER** and **state DOWN**.

The following example shows these same interfaces, but this time as shown in the Bash shell using the **ifconfig** command:

```
bash-4.3# ifconfig Eth1-17
Eth1-17   Link encap:Ethernet  HWaddr 00:42:68:58:f8:eb
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7388 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:1869164 (1.7 MiB)


bash-4.3# ifconfig Eth1-1
Eth1-1    Link encap:Ethernet  HWaddr 00:42:68:58:f8:eb
          inet addr:99.1.1.1  Bcast:99.1.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

The output from the **ifconfig** command provides different information, where the **RUNNING** keyword is used to represent the final state. By default, all netdevices show the keyword **UP**, which represents the ADMIN state of the netdevice in the kernel.

The following IPv4 and IPv6 behaviors are applicable to netdevices configured on the switch:

- **IPv4 address on netdevices** — The IPv4 addresses are plumbed to the kernel space only when the interface is in the **UP** state. Once plumbed, the IPv4 address continues to stay with the netdevice in the

kernel even if the interface goes **DOWN**. It will be removed only after you have entered the following CLI command to explicitly remove the IP address from the NX-OS interface:

```
Interface Eth1/1
           no ip address IP-address
```

- **IPv6 address on netdevices** — Netdevices are always in the Admin **UP** state, so the IPv6 addresses will not get flushed from the kernel when the interface goes down.