



Nokia Service Router Linux  
7220 Interconnect Router  
7250 Interconnect Router  
Release 25.3

## Troubleshooting Toolkit

---

3HE 21392 AAAA TQZZA  
Edition: 01  
March 2025

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

---

This document includes Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

This document is intended for use by Nokia's customers ("You"/"Your") in connection with a product purchased or licensed from any company within Nokia Group of Companies. Use this document as agreed. You agree to notify Nokia of any errors you may find in this document; however, should you elect to use this document for any purpose(s) for which it is not intended, You understand and warrant that any determinations You may make or actions You may take will be based upon Your independent judgment and analysis of the content of this document.

Nokia reserves the right to make changes to this document without notice. At all times, the controlling version is the one available on Nokia's site.

No part of this document may be modified.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

Copyright and trademark: Nokia is a registered trademark of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective owners.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

© 2025 Nokia.

# Table of contents

- 1 About this guide..... 4**
  - 1.1 Precautionary and information messages.....4
  - 1.2 Conventions..... 4
  
- 2 What’s new..... 6**
  
- 3 Interactive traffic-monitoring tool.....7**
  - 3.1 Using the interactive traffic-monitoring tool.....7
    - 3.1.1 Monitoring ICMP Packets..... 9
    - 3.1.2 Displaying verbose output.....10
    - 3.1.3 Capturing packets to a file..... 11
    - 3.1.4 Capturing bidirectional transit traffic..... 11
  
- 4 Switch fabric statistics..... 13**
  - 4.1 Displaying switch fabric statistics..... 13
  
- 5 Packet-trace tool..... 14**
  - 5.1 Configuring packet-trace tool commands..... 14
    - 5.1.1 Configuring the packet-trace tool (using Scapy file format)..... 14
    - 5.1.2 Configuring the packet-trace tool (using base64 format)..... 17
    - 5.1.3 Configuring the packet-trace tool (using pcap format)..... 18

# 1 About this guide

This document describes how to use and configure diagnostic tools for the Nokia Service Router Linux (SR Linux).

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how to use and configure diagnostic tools.

**Note:**

This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Software Release Notes* for information about features supported in each load.

Configuration and command outputs shown in this guide are examples only; actual displays may differ depending on supported functionality and user configuration.

## 1.1 Precautionary and information messages

The following are information symbols used in the documentation.



**DANGER:** Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.



**WARNING:** Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.



**Caution:** Caution indicates that the described activity or situation may reduce your component or system performance.



**Note:** Note provides additional operational information.



**Tip:** Tip provides suggestions for use or best practices.

## 1.2 Conventions

Nokia SR Linux documentation uses the following command conventions.

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in Courier text.
- An open right angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface).

Example: **start** > **connect to**

- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ( [ ] ) indicate optional elements.
- Braces ( { } ) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

## 2 What's new

There have been no updates in this document since it was last released.

### 3 Interactive traffic-monitoring tool

SR Linux features an interactive traffic-monitoring tool that allows you to capture and monitor traffic based on 5-tuple match criteria. The match criteria is injected into a packet capture filter ACL entry that is applied to all subinterfaces; information from matching packets can be displayed on screen or directed to a file.

#### 3.1 Using the interactive traffic-monitoring tool

##### Procedure

You can specify the match criteria either by using the **tools system traffic-monitor** CLI command, or by defining capture-filter ACL entries.

If you use the **tools system traffic-monitor** command to specify the match criteria, SR Linux dynamically creates a capture-filter entry with the match criteria. Packets that match the capture-filter entry are sent to the traffic-monitoring tool running on the CPM and displayed until the traffic-monitoring tool is exited, at which time the dynamically created capture-filter entries are removed.

Use the following syntax to configure the **tools system traffic-monitor** command:

**tools system traffic-monitor** [**source-address** <ip-addr/len>] [**destination-address** <ip-addr/len>] [**protocol** <proto-val>] [**source-port** <value | range>] [**destination-port** <value | range>] [**verbose**] [**output-file** <file-name>] [**hex-output**]

The command parameters are described in [Table 1: Traffic monitoring command parameters](#).

Table 1: Traffic monitoring command parameters

Command/parameter	Description
<b>tools system traffic-monitor</b>	Initiates an interactive monitor session
<b>source-address</b> <ip-addr/len>	Source IP address (IPv4 or IPv6) prefix and netmask length value. For example: 10.10.11.0/24
<b>destination-address</b> <ip-addr/len>	Destination IP address (IPv4 or IPv6) prefix and netmask length value. For example: 10.10.20.0/24
<b>protocol</b> <proto-val>	Specifies the protocol type value to match (required if either port values are specified)
<b>source-port</b> <value   range>	Source port integer value or port range in the format of port1..port2
<b>destination-port</b> <value   range>	Destination port integer value or port range in the format of port1..port2

Command/parameter	Description
<b>verbose</b>	Displays detailed output
<b>output-file</b> <file-name>	Directs output to a file
<b>hex-output</b>	Displays output in hex format

If you specify the match criteria by defining capture-filter ACL entries, starting the traffic-monitoring tool with the **tools system traffic-monitor** command causes the system to send packets that match the defined capture-filter entries to the CPM and display them until the traffic-monitoring tool is exited. Unlike the dynamically created capture-filter entries, the defined capture-filter entries are not removed from the system when the traffic-monitoring tool is exited.

There is one packet capture filter for IPv4 traffic and another packet capture filter for IPv6 traffic. The default IPv4 packet capture filter copies no IPv4 packets, and the default IPv6 packet capture filter copies no IPv6 packets. To configure a packet capture filter, you create an IPv4 or IPv6 ACL filter named capture and specify match conditions and actions.



**Note:** The name **capture** is reserved for packet capture filters; an ACL named **capture** cannot be associated with any interface.

The following is an example of a capture-filter ACL entry:

**Example: Capture filter ACL entry**

```
--{ * candidate shared default }--[ ]--
# info acl acl-filter capture type ipv4
acl {
  acl-filter capture type ipv4 {
    entry 1 {
      match {
        ipv4 {
          protocol icmp
          destination-ip {
            address 10.1.1.1
            mask 255.255.255.255
          }
          source-ip {
            address 10.2.2.2
            mask 255.255.255.255
          }
        }
      }
      action {
        copy {
        }
      }
    }
  }
}
```

Capture filters are applied to traffic after any subinterface filters, but before CPM filters. If a packet is dropped by a subinterface filter, it is not evaluated by a capture filter.

Only a single instance of the traffic-monitoring tool can be running at a time.



If no capture-filter entries are already defined, you must specify the match criteria with the **tools system traffic-monitor** command. If capture-filter entries are already defined, match criteria specified with the **tools system traffic-monitor** command is ignored.

### 3.1.1 Monitoring ICMP Packets

#### Procedure

The following is an example of using the traffic-monitoring tool to monitor ICMP packets. In this example, information about ICMP packets with source address 10.1.1.1/32 and destination address 10.2.2.2/32 is displayed in the monitor window, including the arrival time and source port (ethernet-1/20.1) of each packet. The traffic-monitoring tool captures ICMP packets until you press Ctrl-C.

#### Example: Traffic-monitoring tool

```
# tools system traffic-monitor destination-address 10.1.1.1/32 source-address 10.2.2.2/32 protocol
icmp
Capturing on 'monit'
1 0.000 ethernet-1/20.1 2.2.2.2 1.1.1.1 ICMP 146 Echo (ping) reply id=0x28a8, seq=119/30464, ttl=63
2 1.803 ethernet-1/20.1 2.2.2.2 1.1.1.1 ICMP 146 Echo (ping) reply id=0x28a8, seq=119/30464, ttl=63
3 2.895 ethernet-1/20.1 2.2.2.2 1.1.1.1 ICMP 146 Echo (ping) reply id=0x28a8, seq=119/30464, ttl=63
4 3.749 ethernet-1/20.1 2.2.2.2 1.1.1.1 ICMP 146 Echo (ping) reply id=0x28a8, seq=119/30464, ttl=63
5 4.250 ethernet-1/20.1 2.2.2.2 1.1.1.1 ICMP 146 Echo (ping) reply id=0x28a8, seq=119/30464, ttl=63
6 5.759 ethernet-1/20.1 2.2.2.2 1.1.1.1 ICMP 146 Echo (ping) reply id=0x28a8, seq=119/30464, ttl=63
7 6.644 ethernet-1/20.1 2.2.2.2 1.1.1.1 ICMP 146 Echo (ping) reply id=0x28a8, seq=119/30464, ttl=63
^C
7 packets captured
Command execution aborted : 'tools system traffic-monitor destination-address 10.1.1.1/32 source-
address 10.2.2.2/32 protocol icmp'
```

When you execute the **tools system traffic-monitor** command in the example above, it dynamically creates the following traffic monitoring policy:

```
acl {
  acl-filter capture type ipv4 {
    entry 1 {
      match {
        ipv4 {
          protocol icmp
          destination-ip {
            address 10.1.1.1
            mask 255.255.255.255
          }
          source-ip {
            address 10.2.2.2
            mask 255.255.255.255
          }
        }
      }
      action {
        copy {
        }
      }
    }
  }
}
```

When you terminate the command by pressing Ctrl-C, the dynamically created traffic monitoring policy is removed from all ingress interfaces.

### 3.1.2 Displaying verbose output

#### Procedure

If you include the **verbose** option in the **tools system traffic-monitor** command, it displays the header fields and additional information from the shim header, followed by the original packet.

#### Example: Verbose output

The following example shows verbose output for an ICMP packet:

```
# tools system traffic-monitor destination-address 10.1.1.1/32 source-address 10.2.2.2/32 protocol
icmp verbose
Frame 1: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface 0
  Interface id: 0 (monit)
    Interface name: monit
  Encapsulation type: Ethernet (1)
  Arrival Time: Jan  4, 2098 19:53:01.144789891 UTC
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: -255263715.144789891 seconds
  [Time delta from previous captured frame: 0.000000000 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 0.000000000 seconds]
  Frame Number: 1
  Frame Length: 146 bytes (1168 bits)
  Capture Length: 146 bytes (1168 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:srlinux:eth:ethertype:ip:icmp:data]
Srlinux Packet
  Ingress Port: ethernet-1/20.1
  Padding: 000000
Ethernet II, Src: 00:00:5e:00:53:d2, Dst: 00:00:5e:00:53:41
  Destination: 00:00:5e:00:53:41
    Address: 00:00:5e:00:53:41
      .... 0. .... = LG bit: Globally unique address (factory default)
      .... 0 .... = IG bit: Individual address (unicast)
  Source: 00:00:5e:00:53:d2
    Address: 00:00:5e:00:53:d2
      .... 0. .... = LG bit: Globally unique address (factory default)
      .... 0 .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.20.20.20, Dst: 10.10.10.10
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  Total Length: 84
  Identification: 0xa166 (41318)
  Flags: 0x0000
    0... .... = Reserved bit: Not set
    .0. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  ...0 0000 0000 0000 = Fragment offset: 0
  Time to live: 63
  Protocol: ICMP (1)
  Header checksum: 0x9e07 [validation disabled]
```

```
[Header checksum status: Unverified]
Source: 10.2.2.2
Destination: 10.1.1.1
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0xd01f [correct]
[Checksum Status: Good]
Identifier (BE): 10408 (0x28a8)
Identifier (LE): 43048 (0xa828)
Sequence number (BE): 1352 (0x0548)
Sequence number (LE): 18437 (0x4805)
Timestamp from icmp data: Jan  4, 2098 19:53:01.000000000 UTC
[Timestamp from icmp data (relative): 0.144789891 seconds]
Data (48 bytes)
0000  5a 30 02 00 00 00 00 10 11 12 13 14 15 16 17  20.....
0010  18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27  ..... !"#%&'
0020  28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37  (*)*+,-./01234567
Data: 5a3002000000000101112131415161718191a1b1c1d1e1f...
[Length: 48]
```

### 3.1.3 Capturing packets to a file

#### Procedure

You can direct the captured packets to a file, which can be used as a source for the SR Linux packet trace utility or for Wireshark.

#### Example: Directing captured packets

The following example directs information about ICMP packets with source address 10.1.1.1/32 and destination address 10.2.2.2/32 to a .pcap file.

```
# tools system traffic-monitor destination-address 10.1.1.1/32 source-address 10.2.2.2/32
  protocol icmp output-file /home/linuxadmin/ICMP.pcap
Capturing on 'monit'
6 packets captured
Command execution aborted : 'tools system traffic-monitor destination-address 10.1.1.1/32
source-address 10.2.2.2/32 protocol icmp output-file /home/linuxadmin/ICMP.pcap '
```

Before opening the .pcap file, remove the shim header (the first 48 bytes of the file). For example:

```
$ editcap -C 0:48 /home/linuxadmin/ICMP.pcap /home/linuxadmin/ICMP_chopped.pcap
```

### 3.1.4 Capturing bidirectional transit traffic

#### Procedure

The 5-tuple matching criteria defined in a **tools system traffic-monitor** command applies in one direction only. To capture traffic in both directions, you define capture filters for each direction, then start the traffic-monitoring tool, which applies both capture filters on all ports.

### Example: Capturing bidirectional transit traffic

The following example defines two capture filter entries: one that matches traffic with source address 10.0.0.0/8 and one that matches traffic with destination address 10.0.0.0/8.

```
--{ + candidate shared default }--[ ]--
# info acl acl-filter capture type ipv4
  acl {
    acl-filter capture type ipv4 {
      entry 1 {
        match {
          ipv4 {
            protocol icmp
            destination-ip {
              prefix 10.0.0.0/8
            }
            source-ip {
              prefix 10.0.0.0/8
            }
          }
        }
      }
    }
  }
}
```

When you start the traffic-monitoring tool, it captures packets matching both filter entries. For example:

```
# tools system traffic-monitor
1 0.000000000 ethernet-1/1.1 10.1.1.1 10.2.2.2 ICMP 146 Echo (ping) request id=0x8e3d, seq=23/5888, ttl=64
2 0.000946459 ethernet-1/2.1 10.2.2.2 10.1.1.1 ICMP 146 Echo (ping) reply id=0x8e3d, seq=23/5888, ttl=64 (request in 1)
3 1.001009926 ethernet-1/1.1 10.1.1.1 10.2.2.2 ICMP 146 Echo (ping) request id=0x8e3d, seq=24/6144, ttl=64
4 1.002960526 ethernet-1/2.1 10.2.2.2 10.1.1.1 ICMP 146 Echo (ping) reply id=0x8e3d, seq=24/6144, ttl=64 (request in 3)
5 2.002007753 ethernet-1/1.1 10.1.1.1 10.2.2.2 ICMP 146 Echo (ping) request id=0x8e3d, seq=25/6400, ttl=64
6 2.002949632 ethernet-1/2.1 10.2.2.2 10.1.1.1 ICMP 146 Echo (ping) reply id=0x8e3d, seq=25/6400, ttl=64 (request in 5)
7 3.001983363 ethernet-1/1.1 10.1.1.1 10.2.2.2 ICMP 146 Echo (ping) request id=0x8e3d, seq=26/6656, ttl=64
8 3.003938511 ethernet-1/2.1 10.2.2.2 10.1.1.1 ICMP 146 Echo (ping) reply id=0x8e3d, seq=26/6656, ttl=64 (request in 7)
9 4.003997530 ethernet-1/1.1 10.1.1.1 10.2.2.2 ICMP 146 Echo (ping) request id=0x8e3d, seq=27/6912, ttl=64
104.004949967 ethernet-1/2.1 10.2.2.2 10.1.1.1 ICMP 146 Echo (ping) reply id=0x8e3d, seq=27/6912, ttl=64 (request in 9)
```

## 4 Switch fabric statistics

The switch fabric statistics tool allows you to monitor and troubleshoot common switch fabric issues at different points in the fabric.

The tool can be used to determine the current utilization level. Utilization data is displayed on a per-slot and line-card basis and includes aggregate line card/slot to switch fabric utilization (bidirectional).

See the *SR Linux Data Model Reference Guide* for details on all switch fabric statistic related commands and descriptions of all parameters.

### 4.1 Displaying switch fabric statistics

#### Procedure

Use this procedure to display switch fabric statistics:

```
# enter show
```

```
# tools platform show-fabric-bandwidth
```

#### Example: Displaying switch fabric statistics

```
/platform/show-fabric-bandwidth:
```

Slot	to-fabric Gbps	from-fabric Gbps
1	2369	2370
2	2393	2393
Total	4762	4764

## 5 Packet-trace tool

The packet-trace tool is a troubleshooting command that allows the specification of a probe packet that is injected into the specified interface forwarding context. The tool records the forwarding destination or egress port for the probe packet, as well as any matched ACL records.

The packet-trace tool calculates the egress interfaces for an IP forward flow, while taking into account ECMP and LAG hashing.

The tool reports the following output:

- supplied input parameters
- calculated egress interface and port through which a packet with the specified fields is forwarded
- applied ACL (both ingress and egress)
- reason for a discarded packet



**Note:** Packet-trace is not supported on 7220 IXR-D4 and 7220 IXR-D5 systems.

### 5.1 Configuring packet-trace tool commands

The **packet-trace** command is a tools command that reports the forwarding behavior for a test packet specified in one of the following formats:

- Scapy file format: file specifying the packet format in Scapy packet definition form
- base64 format: string specifying the packet to send in base64 format
- pcap file format: file containing pcap data

Only physical interface types can be used as the ingress interface for injected packets.

#### 5.1.1 Configuring the packet-trace tool (using Scapy file format)

##### Procedure

Use this command to report the forwarding behavior for a specified test packet (file format) that contains a packet formatted in Scapy packet definition form:

```
# tools system packet-trace file <input file in Scapy format> interface <interface name>
```

Packet trace command parameters for specifying an input file are described in [Table 2: Packet trace command parameters using an input file](#).

Table 2: Packet trace command parameters using an input file

Command/parameter	Description
<b>tools system packet-trace</b>	Reports the forwarding behavior for a specified test packet (file format)
<b>file</b> <file name>	File containing the packet format in Scapy packet definition form. The format of the packet definition should match that of the Linux utility Scapy.
<b>interface</b> <interface name>	The name of the configured interface to inject the probe packet

**Example: Scapy input file**

```
# bash cat /tmp/p1.txt
Ether(dst="00:00:5E:00:53:D2",src="00:00:5E:00:53:41")/Dot1Q(vlan=100)/
IP(dst="10.1.5.1",src="192.0.2.1")/UDP(sport=6722,dport=6789)"Hi"/
Raw(RandString(size=512))
```

**Example: command**

```
# tools system packet-trace file /tmp/p1.txt interface ethernet-1/1
```

**Example: output (bridged)**

```
Ether(dst="00:00:5E:00:53:D2",src="00:00:5E:00:53:41")/Dot1Q(vlan=100)/IP(dst=
"10.1.5.1",src="192.35.1.1")/UDP(sport=6722,dport=6789)"Hi"/Raw(RandString(size=512))
Generated packet:
###[ Ethernet ]###
  dst      = 00:00:5e:00:53:d2
  src      = 00:00:5e:00:53:41
  type     = VLAN
###[ 802.1Q ]###
  prio     = 0
  id       = 0
  vlan     = 100
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 542
  id       = 1
  flags    =
  frag     = 0
  ttl      = 64
  proto    = udp
  chksum   = 0x4ea8
  src      = 192.0.2.1
  dst      = 10.1.5.1
  \options \
###[ UDP ]###
  sport    = 6722
  dport    = smc_https
  len      = 522
```

```

    checksum    = 0x251e
###[ Raw ]###
    load       = 'Hi9cmfMxg4lBV6iXRKbe3t2dUJyiGzb7s2GcTQ8YQ0A2PYnF8ntm45l
GqCezZ6ncYF4ijs7hjqxSUjIJdq4YhhRrNSnyUsHkhehhSif
TpT1EEiQN0zNLWgF6DPdcQ078REyyjnI9hqzTNAk0Xhg0mLtg55rkufD8ny0otgBgnz2mpQ0igLSEtYe84VDfdi
Cs5lWTVhGTYCCLxsCYmXEozmSsWqBagdwHe1Ia0voCZ3deUUL6B7paA0b8ua5bZa44G7Z7LneJZ0YxH2Vjb
SqmeukaxyMrkg7NUIxs3aVIwD2jPqra3CBaxokvarX5TyIzNuK2qYeAwnjdzBZo2iZTonXom
JjoDWB2cqG61iEGPLNg5juC7PTa9fglirYgEI2T9rTm8gpTjG6ZgN90g3w0x0xBgwYsNfuXMqp7u9wR8fvfNa4Mm
ZseCC6UUKneSKK0zDyxyHgtSEKwHQQA0H0h6wZttNQRfzST4YB0cFM1tTeo6mCgwApLYX8THGImjvis'
/system/packet-trace-base64:
=====
Ingress information for Packet 3 Ingress Interface ethernet-1/1
=====
Type           : Bridged
Interface      : ethernet-1/1 (4401020001)
Net Instance   : macvrf1
=====
Egress information for Packet 3
=====
Interface      : Flooded in macvrf1
Egress Net Instance : macvrf1
===== (routed)=====

```

**Example: output (routed)**

```

smac='00:00:5E:00:53:D2'
dmac= '00:00:5E:00:53:41'
Ether(src=smac,dst=dmac)/IP(src='192.0.2.1',dst='10.1.5.1')/ICMP()

Generated packet:
###[ Ethernet ]###
dst = 00:00:5e:00:53:41
src = 00:00:5e:00:53:d2
type = IPv4
###[ IP ]###
    version = 4
    ihl = 5
    tos = 0x0
    len = 28
    id = 1
    flags =
    frag = 0
    ttl = 64
    proto = icmp
    checksum = 0x7edc
    src = 192.0.2.1
    dst = 10.1.5.1
    \options \
###[ ICMP ]###
    type = echo-request
    code = 0
    checksum = 0xf7ff
    id = 0x0
    seq = 0x0
/system/packet-trace-base64:
=====
Ingress information for Packet 1 Ingress Interface ethernet-4/29
=====
Type           : Routed
Interface      : lag5 (14000000005)
Sub interface  : lag5.1

```



```

Net Instance      : red
Out Interface     : ethernet-4/22
NextHop ip       : 192.0.52.1
=====
Egress information for Packet 1 Egress Interface ethernet-4/22
=====
Interface        : ethernet-4/22 (4404020016)
Sub interface    : ethernet-4/22.1
Mac Address      : 00:01:03:FF:00:08
=====
    
```

### 5.1.2 Configuring the packet-trace tool (using base64 format)

#### Procedure

Use this command to report the forwarding behavior for a specified test packet using packets specified in base64 format:

```
# tools system packet-trace-base64 interface <interface name> packet <value>
```

Packet trace command parameters for specifying base64 format are described in [Table 3: Packet trace command parameters using base64 string format](#).

Table 3: Packet trace command parameters using base64 string format

Command/parameter	Description
<b>tools system packet-trace-base64</b>	Reports the forwarding behavior for a specified test packet (packet specified in base64 format)
<b>interface &lt;interface name&gt;</b>	The name of the configured interface to inject the probe packet
<b>packet &lt;value&gt;</b>	Packet format in base64 string format

#### Example: command (for routed)

```
# tools system packet-trace-base64 interface ethernet-1/1 packet
"RQAA0gABAABABnS4AQEBAQICAgiAFABQAAAAAAAAAABQAiAAqscAAEdFVCAvIEhUVFAvMS4wDQoNCg=="
```

#### Example: output (routed)

```

tools system packet-trace-base64 interface ethernet-1/3 packet
"MjBFMDlDNz1CQUUzMDAwMTA3RkYwMDAwMDgwMDQ1MDAwMDJFMDAwMDAwMDA0MDExNTA3REMwMz
kwMTA3NjQwMTA1MDExQTQyMUE4NTAwMUE1ODVGMDAwMTAyMDMwNDA1MDYwNzA4MDkwQTBCEMwRDBFMEYxMD
ExM0VGMjA5OTM="
/system/packet-trace-base64:
=====
Ingress information for Packet 77 Ingress Interface ethernet-1/2
=====
Type           : Routed
Interface      : ethernet-1/2 (4401020002)
Sub interface  : ethernet-1/2.1
Instance id   : 1
Out Interface  : ethernet-1/1
    
```

```

Nexthop ip      : 10.1.5.1
=====
Egress information for Packet 77 Egress Interface ethernet-1/1
=====
Interface      : ethernet-1/1 (4401020001)
Sub interface  : ethernet-1/1.1
Mac Address    : 00:00:5e:00:53:41
=====
    
```

**Example: command (for bridged)**

```

tools system packet-trace-base64 packet U0Dv0urSAAED/wBBgQAAZAgARQACHgAB
AABAEU6owCMBAWQBBQEAqhFAGpUvkhqQzVDODlhCuttADVsRHNWRnk4WDRpYXRQbw81UllWenFweE9p
NEJNeXpPQWo5UktOVTRGNkFwTENhNVljNlFVMHVIRTY2UUJzUkh5TWh0SHhQUTZ0aFFTRk5LeXFKNGVn
VVZINDJl0FdBSmt1NlFZeHFicFRmZjEwdHVWdENwCENNMMQ5R1RCeHpseUY3aDZrQjBLMHRXNkF1a2Y0
QllNS3Jld2M5aUVGNRUc1pPbEs0WVFEdkpxRjF0Q1BMMktXNjlnS212bXJmbTLZT2tHWE01MG9haTdp
R2l0amNzRHdkV3VBZEJ40HJvek5tbnVQc2FCYVdPeVBWUjJBT0hVa1BrOW1mclDwYTFDvXV0cU8xZzJk
RVExRXhBNFhaYUlnNlJLZjJvc2swMVJZektac0dKZEFUVnBaSkQzM2tnY2c4UDJnM0dYZFYzZnp4VTNH
bEtEQzhRUUlzQTJvYUJ0ODM4TWniNmW3MudZdGNuZlNDdGZFYlB0TU90S2xSejLhYWZhb3JaQzVMNFdw
TjZXRdVzZWlkeLZtYwdrWUM2VThYY2dKWGpDSXJpR01lQjlobnY4RmFjNkLDZnpR0HF1ZE5iZ21TTG9M
N0l0Tk4xZ1NmQ2JkeUE0RVFabHBGYlFEeVFFYUFJZUUYcG9lbWRPU2x4a0FWYzBQU3kzZEXEYWE=int
erface ethernet-1/1
    
```

**Example: output (bridged)**

```

A:rifa# tools system packet-trace-base64 packet U0Dv0urSAAED/wBBgQAAZAgARQACHgAB
AABAEU6owCMBAWQBBQEAqhFAGpUvkhqQzVDODlhCuttADVsRHNWRnk4WDRpYXRQbw81UllWenFweE9p
NEJNeXpPQWo5UktOVTRGNkFwTENhNVljNlFVMHVIRTY2UUJzUkh5TWh0SHhQUTZ0aFFTRk5LeXFKNGVn
VVZINDJl0FdBSmt1NlFZeHFicFRmZjEwdHVWdENwCENNMMQ5R1RCeHpseUY3aDZrQjBLMHRXNkF1a2Y0
QllNS3Jld2M5aUVGNRUc1pPbEs0WVFEdkpxRjF0Q1BMMktXNjlnS212bXJmbTLZT2tHWE01MG9haTdp
R2l0amNzRHdkV3VBZEJ40HJvek5tbnVQc2FCYVdPeVBWUjJBT0hVa1BrOW1mclDwYTFDvXV0cU8xZzJk
RVExRXhBNFhaYUlnNlJLZjJvc2swMVJZektac0dKZEFUVnBaSkQzM2tnY2c4UDJnM0dYZFYzZnp4VTNH
bEtEQzhRUUlzQTJvYUJ0ODM4TWniNmW3MudZdGNuZlNDdGZFYlB0TU90S2xSejLhYWZhb3JaQzVMNFdw
TjZXRdVzZWlkeLZtYwdrWUM2VThYY2dKWGpDSXJpR01lQjlobnY4RmFjNkLDZnpR0HF1ZE5iZ21TTG9M
N0l0Tk4xZ1NmQ2JkeUE0RVFabHBGYlFEeVFFYUFJZUUYcG9lbWRPU2x4a0FWYzBQU3kzZEXEYWE= int
erface ethernet-1/1
/system/packet-trace-base64:
=====
Ingress information for Packet 4 Ingress Interface ethernet-1/1
=====
Type           : Bridged
Interface      : ethernet-1/1 (4401020001)
Net Instance   : macvrf1
=====
Egress information for Packet 4
=====
Interface      : Flooded in macvrf1
Egress Net Instance : macvrf1
=====
    
```

**5.1.3 Configuring the packet-trace tool (using pcap format)**

**Procedure**

Use the following command to report the forwarding behavior for a specified test packet using packets specified in pcap format:

**# tools system packet-trace pcap-file <file name> [interface <interface name>] [max-packet-count <value>] [packet-number <value>]**

Packet trace command parameters for specifying pcap format are described in [Table 4: Packet trace command parameters using pcap format](#).

Table 4: Packet trace command parameters using pcap format

Command/parameter	Description
<b>tools system packet-trace</b>	Reports the forwarding behavior for a specified test packet (file format)
<b>pcap-file &lt;file name&gt;</b>	Input file in pcap format
<b>interface &lt;interface name&gt;</b>	The name of the configured interface to inject the probe packet
<b>max-packet-count &lt;value&gt;</b>	Number of packets to read from the file (default: 100)
<b>packet-number &lt;value&gt;</b>	Use packet with the specified packet number from the pcap file

**Example: packet-trace command using pcap format**

```
# tools system packet-trace pcap-file data.pcap max-packet-count 1 packet-number 1
interface ethernet-1/2
```

**Example: output of packet trace in pcap format**

```
+-----+
| Number   Time      Ingress  Source   Destina  Protoco  Length  Info  |
|          |          | port    |         | tion    | l       |      |
+-----+-----+-----+-----+-----+-----+-----+
| 1        0.046971  etherne  192.1.7.1  10.1.5.1  UDP      2545    6722  |
|          |          | t-1/2.1  1         | 8        |         |         | \u2192  | |
|          |          |          |          |          |         |         | 6789 Le  |
|          |          |          |          |          |         |         | n=2454  |
+-----+-----+-----+-----+-----+-----+
Enter packet number (default: [1]): 1
###[ Ethernet ]###
  dst      = 00:00:5e:00:53:41
  src      = 00:00:5e:00:53:d2
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl     = 5
  tos     = 0x0
  len     = 2482
  id      = 0
  flags   =
  frag    = 0
  ttl     = 64
  proto   = udp
  chksum  = 0xd09
  src     = 192.0.2.1
  dst     = 10.1.5.1
  \options \
```

```

###[ UDP ]###
sport      = 6722
dport      = smc_https
len        = 2462
chksum     = 0xcd6b
###[ Raw ]###
Load      =
'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f
!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0....'
/system/packet-trace-base64:
=====
Ingress information for Packet 10 Ingress Interface ethernet-1/2
=====
Type                : Routed
Interface           : ethernet-1/2 (4401020002)
Sub interface       : ethernet-1/2.1
Instance id        : 1
Out Interface       : ethernet-1/1
Next hop ip        : 192.0.2.54
=====
Egress information for Packet 10 Egress Interface ethernet-1/1
=====
Interface           : ethernet-1/1 (4401020001)
Sub interface       : ethernet-1/1.8
Mac Address         : 00:00:5e:00:53:41

```



# Customer document and product support



## **Customer documentation**

[Customer documentation welcome page](#)



## **Technical support**

[Product support portal](#)



## **Documentation feedback**

[Customer documentation feedback](#)