

Chapter 13

Source Class Usage

Source Class Usage (SCU) is a method of monitoring traffic in Juniper Networks routers. It extends the functionality of an existing accounting method called Destination Class Usage (DCU). Like DCU, SCU gives you a way to define certain traffic as a group and monitor that group traffic using command-line interface (CLI) show commands, accounting profiles, or Simple Network Management Protocol (SNMP). Instead of using the standard destination address lookup required by DCU, SCU performs the lookup on the source address. This functionality of SCU gives you greater flexibility in selecting which traffic to meter.

DCU commonly profiles traffic traveling from the customer edge to the provider core. SCU primarily tracks packets moving from the provider core to the customer edge.

This guide explains three methods of using SCU: setting up standard SCU for CLI monitoring, using SCU with Layer 3 VPNs, and establishing accounting profiles based on SCU source classes.

This feature guide covers these topics:

Overview on page 440

System Requirements on page 440

Terms and Acronyms on page 441

Configuring SCU on page 441

Example: SCU Configuration on page 444

Check Your Work on page 446

Configuring SCU with Layer 3 VPNs on page 451

Example: SCU in a Layer 3 VPN Configuration on page 453

Check Your Work on page 459

Configuring Accounting Profiles with SCU on page 460

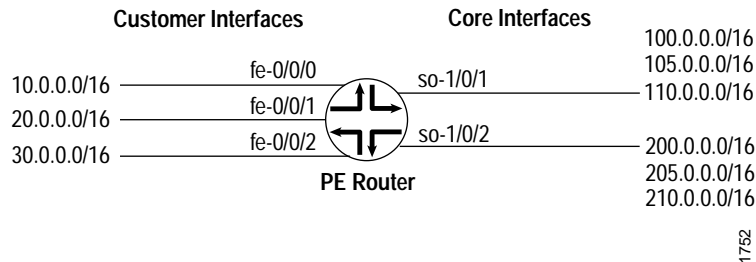
For More Information on page 462

Revision History on page 462

Overview

SCU is a logical extension of the DCU concept. DCU was created so that Juniper Networks customers could count on a per-interface basis how much traffic was sent to specified prefixes. Figure 45 shows a service provider edge (PE) router diagram.

Figure 45: DCU/SCU Concept



The Fast Ethernet interfaces contain inbound traffic from customers, and the SONET interfaces are connected to outbound public network prefixes. With DCU configured on the Fast Ethernet interfaces, you can track how much traffic is sent to a specific prefix in the core of the network originating from one of the specified interfaces (in this case, the Fast Ethernet interfaces).

However, DCU limits your ability to keep track of traffic moving in the reverse direction. It can account for all traffic that arrives on a core interface and heads toward a specific customer, but it cannot count traffic that arrives on a core interface from a specific prefix. For example, DCU can process cumulative traffic headed toward interface fe-0/0/0, but cannot differentiate between traffic coming only from 100.0.0.0/16 and traffic coming from all prefixes.

You can track source-based traffic by using SCU, which allows you to monitor the amount of traffic originating from a specific prefix. With this feature, usage can be tracked and customers can be billed for the traffic they receive.

System Requirements

To implement SCU, your system must meet these requirements:

- JUNOS Release 5.4 or later

- Three Juniper Networks routers for basic SCU and five routers for SCU with Layer 3 VPNs. One router containing an Internet Processor II (such as the M5, M10, M20, M40e, or M160 Internet routers, or the T640 Internet routing node) acts as a source class usage transit router, and the other routers or routing nodes are used to generate traffic or participate in the Layer 3 VPN.

- A Tunnel PIC for SCU with Layer 3 VPNs

Terms and Acronyms

Destination Class Usage (DCU)—A method of grouping certain types of traffic and monitoring these groups through CLI show commands, accounting profiles, or SNMP. DCU uses a destination address lookup when determining group membership. For more information about DCU, see the *JUNOS Internet Software Configuration Guide: Policy Framework*.

Source Class Usage (SCU)—A method of grouping certain types of traffic and monitoring these groups through CLI show commands, accounting profiles, or SNMP. SCU uses a source address lookup when determining group membership. For more information about SCU, see the *JUNOS Internet Software Configuration Guide: Policy Framework*.

Source Address (SA)—The IP address of a device sending a packet. This address is included in the IP header and is analyzed by the router for a variety of services, including source-based filtering, policing, class of service (CoS), and SCU.

Destination Address (DA)—The IP address of a device intended as the receiver for a packet. This address is included in the IP header and is the main address analyzed by the router during routing table lookups and DCU.

Configuring SCU

To configure SCU, complete the following:

Configure Route Filters and Source Classes in a Routing Policy on page 441

Apply the Policy to the Forwarding Table on page 442

Enable Accounting on Inbound and Outbound Interfaces on page 442

To view examples of basic SCU, go to the following sections:

Example: SCU Configuration on page 444

Check Your Work on page 446

Configure Route Filters and Source Classes in a Routing Policy

Begin configuring SCU by creating prefix route filters in a policy statement. These prefixes indicate the source addresses to monitor. Within the policy statement, you must define and name the source classes attached to the filters.

```
[edit policy-options]
policy-statement policy-name {
  term term-name
    from {
      route-filter address/prefix;
    }
    then source-class class-name;
  }
}
```

An alternate configuration method, `forwarding-class`, is even more flexible. It allows your route filters to apply to an SCU profile, a DCU profile, or both simultaneously. Additionally, if you have only one term, you can implement the from and to statements at the `[edit policy-options policy-statement policy-name]` hierarchy level.

```
[edit policy-options]
policy-statement policy-name {
  from {
    route-filter 105.15.0.0/16 orlonger;
  }
  then forwarding-class gold-class;
}
```

A third option is the existing DCU parameter of `destination-class`. For more information on DCU, see the *JUNOS Internet Software Configuration Guide: Policy Framework*.

Apply the Policy to the Forwarding Table

Next, apply the policy you created to the forwarding table. When you apply the policy, the network prefixes you defined are marked with the appropriate source class.

```
[edit]
routing-options {
  forwarding-table {
    export policy-name;
  }
}
```

Enable Accounting on Inbound and Outbound Interfaces

Unlike DCU, which only requires implementation on a single interface, accounting for SCU must be enabled on two interfaces: the inbound and outbound physical or logical interfaces traversed by the source class. You must define explicitly the two interfaces on which SCU monitored traffic is expected to arrive and depart. This is because SCU performs two lookups in the routing table: a source address (SA) and a destination address (DA) lookup. In contrast, DCU only has a single destination address lookup. By specifying the addresses involved in the additional SCU SA lookup, you minimize the performance impact on your router.

An individual SCU interface can be configured as an input interface, an output interface, or both. Currently, SCU can be enabled in an IPv4 (family inet) network. You configure SCU accounting with the following commands:

```
[edit]
interfaces
  interface-name {
    unit unit-number {
      family inet {
        accounting {
          source-class-usage {
            (input | output | [input output]);
          }
          destination-class-usage;
        }
      }
    }
  }
}
```

After the full SCU configuration is enabled, every packet arriving on an SCU input interface is subjected to an SA-based lookup and then a DA-based lookup. In addition, an individual set of counters for every configured SCU class is maintained by the router on a per-interface and per-protocol family basis.

When you enable SCU or DCU, keep the following information in mind:

In JUNOS Release 5.6 and later, you can use a source class or a destination class as a match condition in a firewall filter. To configure, include the destination-class or source-class statement at the [edit firewall filter *firewall-name* term *term-name* from] hierarchy level. For more information about firewall filters, see the *JUNOS Internet Software Configuration Guide: Policy Framework*.

You can assign up to 126 source classes and 126 destination classes.

A source or destination class is applied to a packet only once during the routing table lookup. When a network prefix matches a class-usage policy, SCU is assigned to packets first; DCU is assigned only if SCU has not been assigned. Be careful when using both class types, since misconfiguration can result in uncounted packets. The following example explores one potential mishap:

A packet arrives on a router interface configured for both SCU and DCU. The packet's source address matches an SCU class and its destination matches a DCU class. Consequently, the packet is subjected to a source lookup, marked with the SCU class, and the DCU class is ignored. As a result, the packet is forwarded to the outbound interface with only the SCU class still intact.

However, the outbound interface lacks an SCU configuration. As the packet is ready to leave the router, the router notices the output interface is not configured for SCU and the packet is not counted by SCU. Likewise, even though the prefix matched the DCU prefix, the DCU counters do not increment since DCU was superseded by SCU at the inbound interface.

To solve this problem, make sure you configure both the inbound and outbound interfaces completely or configure only one class type per interface per direction.

Classes cannot be mapped to directly connected prefixes configured on local interfaces. This is true for DCU and SCU classes.

If you use multiple terms within a single policy, you only need to configure the policy name and apply it to the forwarding table once. This makes it easier to change options within your terms without having to reconfigure the main policy.

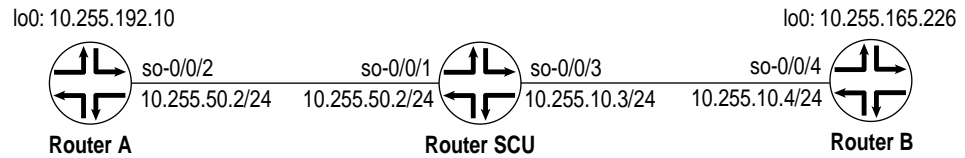
Use the correct interface when tracking SCU statistics, setting accounting profiles, and establishing classes:

Execute CLI show commands and accounting profiles at the desired outbound interface to track SCU traffic. SCU counters increment at the SCU output interface.

Apply your classes to the inbound and outbound interfaces by means of the input and output SCU interface parameters.

Example: SCU Configuration

Figure 46: SCU Topology Diagram



1761

Figure 46 shows a basic SCU configuration with three routers. Source routers A and B use loopback addresses as the prefixes to be monitored. Most of the configuration tasks and actual monitoring occurs on transit router SCU.

Begin your configuration on router A. The loopback address on router A contains the origin of the prefix that is to be assigned to source class A on router SCU. However, no SCU processing happens on this router. Therefore, configure router A for basic Open Shortest Path First (OSPF) routing and include your loopback interface and interface so-0/0/2 in the OSPF process.

```
Router A: [edit]
interfaces {
  so-0/0/2 {
    unit 0 {
      family inet {
        address 10.255.50.2/24;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 10.255.192.10/32;
      }
    }
  }
}
protocols
  ospf {
    area 0.0.0.0 {
      interface so-0/0/2.0;
      interface lo0.0;
    }
  }
}
```

Router SCU handles the bulk of the activity in this example. On router SCU, enable source class usage on the inbound and outbound interfaces at the [edit interfaces *interface-name* unit *unit-number* family inet accounting] hierarchy level. Make sure you specify the expected traffic: input, output, or, in this case, both.

Next, configure a route filter policy statement that matches the prefixes of the loopback addresses from routers A and B. Include statements in the policy that classify packets from Router A in one group named scu-class-a and packets from Router B in a second class named scu-class-b. Notice the efficient use of a single policy containing multiple terms.

Last, apply the policy to the forwarding table.

```
Router SCU [edit]
interfaces {
  so-0/0/1 {
    unit 0 {
      family inet {
        accounting {
          source-class-usage {
            input;
            output;
          }
        }
        address 10.255.50.1/24;
      }
    }
  }
  so-0/0/3 {
    unit 0 {
      family inet {
        accounting {
          source-class-usage {
            input;
            output;
          }
        }
        address 10.255.10.3/24;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 10.255.6.111/32;
      }
    }
  }
}
protocols
  ospf {
    area 0.0.0.0 {
      interface so-0/0/1.0;
      interface so-0/0/3.0;
    }
  }
}
routing-options {
  forwarding-table {
    export scu-policy;
  }
}
```

```

policy-options {
  policy-statement scu-policy {
    term 0 {
      from {
        route-filter 10.255.192.0/24 orlonger;
      }
      then source-class scu-class-a;
    }
    term 1 {
      from {
        route-filter 10.255.165.0/24 orlonger;
      }
      then source-class scu-class-b;
    }
  }
}

```

Complete the configuration tasks on router B. Just as router A provides a source prefix, router B's loopback address matches the prefix assigned to scu-class-b on router SCU. Again, no SCU processing happens on this router, so configure router B for basic OSPF routing and include your loopback interface and interface so-0/0/4 in the OSPF process.

Router B:

```

interfaces {
  so-0/0/4 {
    unit 0 {
      family inet {
        address 10.255.10.4/24;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 10.255.165.226/32;
      }
    }
  }
}
protocols
  ospf {
    area 0.0.0.0 {
      interface so-0/0/4.0;
      interface lo0.0;
    }
  }
}

```

Check Your Work

To verify that SCU is functioning properly, use the following commands:

```
show interfaces interface-name statistics
```

```
show interfaces interface-name (extensive | detail)
```

```
show route (extensive | detail)
```

```
show interfaces source-class source-class-name interface-name
```

```
clear interface interface-name statistics
```

You should always verify SCU statistics at the outbound SCU interface on which you configured the output statement. You can follow three steps to check the functionality of SCU:

1. Clear all counters on your SCU-enabled router and verify that they are empty.
2. Send a ping from one edge router to another edge router to generate SCU traffic across the SCU-enabled router.
3. Verify that the counters are incrementing correctly on the outbound interface.

The following section shows the output of these commands as used with the configuration example.

Step One: Clear the Counters

```
user@scu> clear interfaces statistics all

user@scu> show interfaces so-0/0/1.0 statistics
Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119)
Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
Protocol inet, MTU: 4470
Source class                               Packets          Bytes
      scu-class-a                           0                0
      scu-class-b                           0                0
Addresses, Flags: Is-Preferred Is-Primary
Destination: 10.255.50/24, Local: 10.255.50.1

user@scu> show interfaces so-0/0/3.0 statistics
Logical interface so-0/0/3.0 (Index 6) (SNMP ifIndex 113)
Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
Protocol inet, MTU: 4470
Source class                               Packets          Bytes
      scu-class-a                           0                0
      scu-class-b                           0                0
Addresses, Flags: Is-Preferred Is-Primary
Destination: 10.255.10/24, Local: 10.255.10.3

user@scu> show interfaces source-class scu-class-a so-0/0/3.0
Protocol inet
Source class                               Packets          Bytes
      scu-class-a                           0                0

user@scu> show interfaces source-class scu-class-b so-0/0/1.0
Protocol inet
Source class                               Packets          Bytes
      scu-class-b                           0                0
```

Step Two: Ping from the Edge Routers

```
user@routerB> ping 10.255.192.10 source 10.255.165.226 rapid 10000

user@routerA> ping 10.255.165.226 source 10.255.192.10 rapid 10000
```

Step Three: Verify the Counters

```
user@scu> show interfaces source-class scu-class-a so-0/0/3.0
Protocol inet
Source class                               Packets          Bytes
      scu-class-a                           20000           1680000
```

```

user@scu> show interfaces source-class scu-class-a so-0/0/1.0
Protocol inet
Source class
scu-class-b
Packets
20000
Bytes
1680000

user@scu> show interfaces so-0/0/3.0 statistics
Logical interface so-0/0/3.0 (Index 6) (SNMP ifIndex 113)
Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
Protocol inet, MTU: 4470
Source class
scu-class-a
scu-class-b
Packets
20000
0
Bytes
1680000
0
Addresses, Flags: Is-Preferred Is-Primary
Destination: 10.255.10/24, Local: 10.255.10.3

user@scu> show interfaces so-0/0/1.0 statistics
Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119)
Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
Protocol inet, MTU: 4470
Source class
scu-class-a
scu-class-b
Packets
0
20000
Bytes
0
1680000
Addresses, Flags: Is-Preferred Is-Primary
Destination: 10.255.50/24, Local: 10.255.50.1

user@scu> show route extensive 10.255.192.0

inet.0: 26 destinations, 28 routes (25 active, 0 holddown, 1 hidden)
10.255.192.0/18 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.255.192.0/18 -> {so-0/0/1.0}
Source class: scu-class-a
*OSPF Preference: 150
Next hop: via so-0/0/1.0,, selected
State: <Active Int Ext>
Age: 2:49:31 Metric: 0 Tag: 0
Task: OSPF
Announcement bits (1): 0-KRT
AS path: I

user@scu> show route extensive 10.255.165.0

inet.0: 26 destinations, 28 routes (25 active, 0 holddown, 1 hidden)
10.255.165.0/20 (1 entry, 1 announced)
TSI:
KRT in-kernel 10.255.165.0/20 -> {so-0/0/3.0}
Source class: scu-class-b
*OSPF Preference: 150
Next hop: via so-0/0/3.0,, selected
State: <Active Int Ext>
Age: 2:49:31 Metric: 0 Tag: 0
Task: OSPF
Announcement bits (1): 0-KRT
AS path: I

```

```

user@scu> show interfaces so-0/0/1 detail
Physical interface: so-0/0/1, Enabled, Physical link is Up
Interface index: 12, SNMP ifIndex: 17, Generation: 11
Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC3,
Loopback: None, FCS: 16, Payload scrambler: Enabled
Device flags   : Present Running
Interface flags: Point-To-Point SNMP-Traps
Link flags     : Keepalives
Hold-times     : Up 0 ms, Down 0 ms
Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
Keepalive statistics:
  Input : 46 (last seen 00:00:01 ago)
  Output: 45 (last sent 00:00:00 ago)
LCP state: Opened
NCP state: inet: Opened, inet6: Not-configured, iso: Not-configured, mpls:
Not-configured
CHAP state: Not-configured
Last flapped   : 2002-04-19 11:49:22 PDT (03:10:09 ago)
Statistics last cleared: 2002-04-19 14:52:04 PDT (00:07:27 ago)
Traffic statistics:
  Input bytes   :                1689276                40 bps
  Output bytes  :                1689747                48 bps
  Input packets:                20197                   0 pps
  Output packets:               20200                   0 pps
Queue counters:      Queued packets  Transmitted packets  Dropped packets
  0 best-effort      20053                20053                0
  1 expedited-fo     0                    0                    0
  2 assured-forw     0                    0                    0
  3 network-cont     146                  146                  0
SONET alarms   : None
SONET defects  : None

Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119) (Generation 3)
Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
Protocol inet, MTU: 4470
Flags: SCU-in, SCU-out
Generation: 6 Route table: 0

```

Source class	Packets	Bytes
scu-class-a	0	0
scu-class-b	20000	1680000

```

Filters: Input: icmp-so-0/0/1.0-i, Output: icmp-so-0/0/1.0-o
Addresses, Flags: Is-Preferred Is-Primary
Destination: 10.255.50/24, Local: 10.255.50.1, Broadcast: Unspecified,
Generation: 8

```

```

user@scu> show interfaces so-0/0/1 extensive
Physical interface: so-0/0/1, Enabled, Physical link is Up
Interface index: 12, SNMP ifIndex: 17, Generation: 11
Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC3,
Loopback: None, FCS: 16, Payload scrambler: Enabled
Device flags   : Present Running
Interface flags: Point-To-Point SNMP-Traps
Link flags     : Keepalives
Hold-times     : Up 0 ms, Down 0 ms
Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
Keepalive statistics:
  Input : 51 (last seen 00:00:04 ago)
  Output: 50 (last sent 00:00:05 ago)
LCP state: Opened
NCP state: inet: Opened, inet6: Not-configured, iso: Not-configured, mpls:
Not-configured

```

```

CHAP state: Not-configured
Last flapped   : 2002-04-19 11:49:22 PDT (03:11:05 ago)
Statistics last cleared: 2002-04-19 14:52:04 PDT (00:08:23 ago)
Traffic statistics:
Input bytes   :          1689884          264 bps
Output bytes  :          1690388          280 bps
Input packets :           20215           0 pps
Output packets:           20217           0 pps
Input errors:
  Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Giants: 0,
  Bucket drops: 0, Policed discards: 0, L3 incompletes: 0,
  L2 channel errors: 0, L2 mismatch timeouts: 0, HS link CRC errors: 0,
  HS link FIFO overflows: 0
Output errors:
  Carrier transitions: 0, Errors: 0, Drops: 0, Aged packets: 0,
  HS link FIFO underflows: 0
Queue counters:
  Queued packets  Transmitted packets  Dropped packets
0 best-effort    20053                20053            0
1 expedited-fo    0                    0                0
2 assured-forw    0                    0                0
3 network-cont   164                  164              0
SONET alarms   : None
SONET defects  : None
SONET PHY:
  Seconds      Count  State
  PLL Lock     0      0 OK
  PHY Light    0      0 OK
SONET section:
  BIP-B1       0      0
  SEF          0      0 OK
  LOS          0      0 OK
  LOF          0      0 OK
  ES-S         0
  SES-S        0
  SEFS-S       0
SONET line:
  BIP-B2       0      0
  REI-L        0      0
  RDI-L        0      0 OK
  AIS-L        0      0 OK
  BERR-SF      0      0 OK
  BERR-SD      0      0 OK
  ES-L         0
  SES-L        0
  UAS-L        0
  ES-LFE       0
  SES-LFE      0
  UAS-LFE      0
SONET path:
  BIP-B3       0      0
  REI-P        0      0
  LOP-P        0      0 OK
  AIS-P        0      0 OK
  RDI-P        0      0 OK
  UNEQ-P       0      0 OK
  PLM-P        0      0 OK
  ES-P         0
  SES-P        0
  UAS-P        0
  ES-PFE       0
  SES-PFE      0
  UAS-PFE      0

```

```

Received SONET overhead:
  F1      : 0x00, J0      : 0x00, K1      : 0x00, K2      : 0x00
  S1      : 0x00, C2      : 0xcf, C2(cmp) : 0xcf, F2      : 0x00
  Z3      : 0x00, Z4      : 0x00, S1(cmp) : 0x00, V5      : 0x00
  V5(cmp) : 0x00
Transmitted SONET overhead:
  F1      : 0x00, J0      : 0x01, K1      : 0x00, K2      : 0x00
  S1      : 0x00, C2      : 0xcf, F2      : 0x00, Z3      : 0x00
  Z4      : 0x00, V5      : 0x00
Received path trace: e so-0/0/1
  65 20 73 6f 2d 30 2f 30 2f 31 00 00 00 00 00 00 00 00  e so-0/0/1.....
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0d 0a  .....
Transmitted path trace: scu so-0/0/1
  67 68 62 20 73 6f 2d 30 2f 30 2f 31 00 00 00 00  scu so-0/0/1....
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
HDLC configuration:
  Policing bucket: Disabled
  Shaping bucket : Disabled
  Giant threshold: 4484, Runt threshold: 3
Packet Forwarding Engine configuration:
  Destination slot: 0, PLP byte: 1 (0x00)
  CoS transmit queue      Bandwidth      Buffer      Priority      Limit
                          %          bps      %          bytes
  0 best-effort            0          0  0          0          low  none
  1 expedited-forwarding  0          0  0          0          low  none
  2 assured-forwarding   0          0  0          0          low  none
  3 network-control       0          0  0          0          low  none

Logical interface so-0/0/1.0 (Index 4) (SNMP ifIndex 119) (Generation 3)
  Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
  Protocol inet, MTU: 4470
  Flags: SCU-in, SCU-out
  Generation: 6 Route table: 0
      Source class      Packets      Bytes
      scu-class-a      0            0
      scu-class-b      20000        1680000
  Filters: Input: icmp-so-0/0/1.0-i, Output: icmp-so-0/0/1.0-o
  Addresses, Flags: Is-Preferred Is-Primary
  Destination: 10.255.50/24, Local: 10.255.50.1, Broadcast: Unspecified,
  Generation: 8

```

Configuring SCU with Layer 3 VPNs

SCU can be implemented over regular interfaces; it is also used in combination with Layer 3 VPNs. When you view SCU traffic on an ingress PE router, use the standard procedure outlined in “Configuring SCU” on page 441. However, when you enable packet counting for Layer 3 VPNs at the egress point of the MPLS tunnel, you need to take some additional steps. To configure SCU on the egress router in a Layer 3 VPN, perform the following steps:

Configure Input SCU on the vt Interface of the Egress PE Router on page 452

Map the SCU-Enabled vt Interface to the VRF Instance on page 452

Configure SCU on the Output Interface on page 453

SCU over Layer 3 VPNs is not supported when the vrf table label is configured. Also, SCU is not supported over Layer 2 VPNs.

To view examples of SCU with Layer 3 VPNs, go to the following sections:

Example: SCU in a Layer 3 VPN Configuration on page 453

Check Your Work on page 459

Configure Input SCU on the vt Interface of the Egress PE Router

To enable SCU in a Layer 3 VPN, configure source class usage on the virtual loopback tunnel (vt) interface of the egress provider edge (PE) router equipped with a Tunnel PIC. The interface is equivalent to the inbound SCU interface, so use the input statement at the [edit interfaces vt-interface-number unit 0 family inet accounting source-class-usage] hierarchy level:

```
[edit]
interfaces
  vt-0/3/0 {
    unit 0 {
      family inet {
        accounting {
          source-class-usage {
            input;
          }
        }
      }
    }
  }
}
```

Map the SCU-Enabled vt Interface to the VRF Instance

Next, include the VPN loopback tunnel interface in the desired VRF instance at the [edit routing-instances routing-instance-name] hierarchy level:

```
[edit]
routing-instances {
  routing-instance-name {
    instance-type vrf;
    interface at-2/1/1.0;
    interface vt-0/3/0.0;
    route-distinguisher 10.250.14.225:100;
    vrf-import import-policy-name;
    vrf-export export-policy-name;
    protocols {
      bgp {
        group to-r4 {
          local-address 10.20.253.1;
          peer-as 400;
          neighbor 10.20.253.2;
        }
      }
    }
  }
}
```

Configure SCU on the Output Interface

Since VPN traffic enters the egress router through the VPN loopback tunnel interface, you still need to determine the exit interface for this traffic. To complete your SCU configuration, configure the output version of source class usage on the exit interface of your egress router:

```
[edit interfaces]
at-1/1/0 {
  unit 0 {
    family inet {
      accounting {
        source-class-usage {
          output;
        }
      }
    }
  }
}
```

Example: SCU in a Layer 3 VPN Configuration

Figure 47: SCU in a Layer 3 VPN Topology Diagram

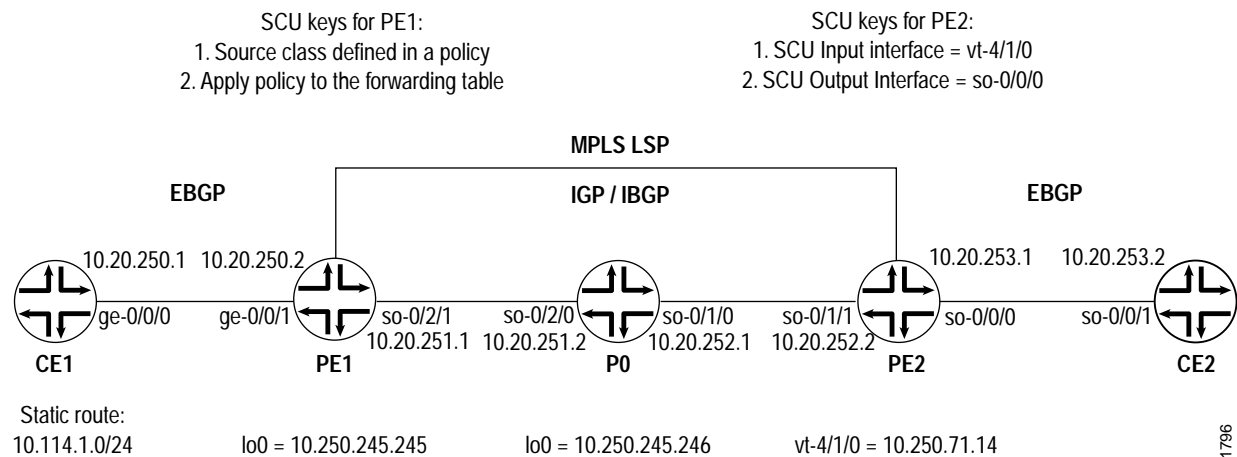


Figure 47 displays a Layer 3 VPN topology. CE1 and CE2 are customer edge (CE) routers connected by a VPN through provider routers PE1, P0, and PE2. EBGP is established between routers CE1 and PE1; IBGP connects routers PE1 and PE2 over an IS-IS/MPLS/LDP core; and a second EBGP connection flows between routers PE2 and CE2.

On router CE1, begin your VPN by setting up an EBGP connection to PE1. Install a static route of 10.114.1.0/24 and advertise this route to your EBGP neighbor.

```

CE1 [edit]
interfaces {
  ge-0/0/0 {
    unit 0 {
      family inet {
        address 10.20.250.1/30;
      }
    }
  }
}
routing-options {
  static {
    route 10.114.1.0/24 reject;
  }
  autonomous-system 100;
}
protocols {
  bgp {
    group to-pe1 {
      local-address 10.20.250.1;
      export inject-direct;
      peer-as 300;
      neighbor 10.20.250.2;
    }
  }
}
policy-options {
  policy-statement inject-direct {
    term 1 {
      from {
        protocol static;
        route-filter 10.114.1.0/24 exact;
      }
      then accept;
    }
    term 2 {
      from protocol direct;
      then accept;
    }
  }
}

```

On PE1, complete the EBGp connection to CE1 through a VRF routing instance. Set an export policy for your VRF instance that puts BGP traffic into a community, and an import policy that accepts like community traffic from your VPN neighbor. Lastly, configure an IBGP relationship to router PE2 that runs over an IS-IS, MPLS and LDP core.

```

PE1 [edit]
interfaces {
  ge-0/0/1 {
    unit 0 {
      family inet {
        address 10.20.250.2/30;
      }
    }
  }
}

```

```

so-0/2/1 {
  unit 0 {
    family inet {
      address 10.20.251.1/30;
    }
    family iso;
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 10.250.245.245/32;
    }
    family iso;
    family mpls;
  }
}
routing-options {
  autonomous-system 300;
}
protocols {
  mpls {
    interface so-0/2/1;
  }
  bgp {
    group ibgp {
      type internal;
      local-address 10.250.245.245;
      family inet-vpn {
        unicast;
      }
      neighbor 10.250.71.14;
    }
  }
  isis {
    interface so-0/2/1;
  }
  ldp {
    interface so-0/2/1;
  }
}
policy-options {
  policy-statement red-import {
    from {
      protocol bgp;
      community red-com;
    }
    then accept;
  }
  policy-statement red-export {
    from protocol bgp;
    then {
      community add red-com;
      accept;
    }
  }
  community red-com members target:20:20;
}

```

```

routing-instances {
  red {
    instance-type vrf;
    interface ge-0/0/1.0;
    route-distinguisher 10.250.245.245:100;
    vrf-import red-import;
    vrf-export red-export;
    protocols {
      bgp {
        group to-ce1 {
          local-address 10.20.250.2;
          peer-as 100;
          neighbor 10.20.250.1;
        }
      }
    }
  }
}

```

On P0, connect the IBGP neighbors located at PE1 and PE2. Remember to include VPN-related protocols (MPLS, LDP, and IGP) on all interfaces.

```

P0 [edit]
interfaces {
  so-0/1/0 {
    unit 0 {
      family inet {
        address 10.20.252.1/30;
      }
      family iso;
      family mpls;
    }
  }
  so-0/2/0 {
    unit 0 {
      family inet {
        address 10.20.251.2/30;
      }
      family iso;
      family mpls;
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 10.250.245.246/32;
      }
      family iso;
      family mpls;
    }
  }
}
routing-options {
  autonomous-system 300;
}
protocols {
  mpls {
    interface so-0/1/0;
    interface so-0/2/0;
  }
}

```

```

isis {
  interface all;
}
ldp {
  interface all;
}
}

```

On PE2, complete the IBGP relationship to router PE1. Establish an EBGp connection to CE2 through a VRF routing instance. Set an export policy for the VRF instance that places BGP traffic into a community, and an import policy that accepts like community traffic from the VPN neighbor. Next, establish a policy that adds the static route from CE1 to a source class called GOLD1. Also, export this SCU policy into the forwarding table. Finally, set your vt interface as the SCU input interface and establish the CE-facing interface so-0/0/0 as the SCU output interface.

```

PE2 [edit]
interfaces {
  so-0/1/1 {
    unit 0 {
      family inet {
        address 10.20.252.2/30;
      }
      family iso;
      family mpls;
    }
  }
  so-0/0/0 {
    unit 0 {
      family inet {
        accounting {
          source-class-usage {
            output;
          }
        }
        address 10.20.253.1/30;
      }
    }
  }
  vt-4/1/0 {
    unit 0 {
      family inet {
        accounting {
          source-class-usage {
            input;
          }
        }
        address 10.250.71.14/32;
      }
      family iso;
      family mpls;
    }
  }
}
routing-options {
  autonomous-system 300;
  forwarding-table {
    export inject-customer2-dest-class;
  }
}

```

```

}
protocols {
  mpls {
    interface so-0/1/1;
    interface vt-4/1/0;
  }
  bgp {
    group ibgp {
      type internal;
      local-address 10.250.71.14;
      family inet-vpn {
        unicast;
      }
      neighbor 10.250.245.245;
    }
  }
  isis {
    interface so-0/1/1;
  }
  ldp {
    interface so-0/1/1;
  }
}
routing-instances {
  red {
    instance-type vrf;
    interface so-0/0/0.0;
    interface vt-4/1/0.0;
    route-distinguisher 10.250.71.14:100;
    vrf-import red-import;
    vrf-export red-export;
    protocols {
      bgp {
        group to-ce2 {
          local-address 10.20.253.1;
          peer-as 400;
          neighbor 10.20.253.2;
        }
      }
    }
  }
}
policy-options {
  policy-statement red-import {
    from {
      protocol bgp;
      community red-com;
    }
    then accept;
  }
  policy-statement red-export {
    from protocol bgp;
    then {
      community add red-com;
      accept;
    }
  }
}

```

```

policy-statement inject-customer2-dest-class {
  term term-gold1-traffic {
    from {
      route-filter 10.114.1.0/24 exact;
    }
    then source-class GOLD1;
  }
}
community red-com members target:20:20;
}

```

On Router CE2, complete the VPN path by finishing the EBGp connection to PE2.

```

CE2 [edit]
interfaces {
  so-0/0/1 {
    unit 0 {
      family inet {
        address 10.20.253.2/30;
      }
    }
  }
}
routing-options {
  autonomous-system 400;
}
protocols {
  bgp {
    group to-pe2 {
      local-address 10.20.253.2;
      export inject-direct;
      peer-as 300;
      neighbor 10.20.253.1;
    }
  }
}
policy-options {
  policy-statement inject-direct {
    from {
      protocol direct;
    }
    then accept;
  }
}
}

```

Check Your Work

To verify that SCU is functioning properly in the Layer 3 VPN, use the following commands:

```

show interfaces interface-name statistics

show interfaces source-class source-class-name interface-name

show interfaces interface-name (extensive | detail)

show route (extensive | detail)

clear interface interface-name statistics

```

You should always verify SCU statistics at the outbound SCU interface on which you configured the output statement. To check SCU functionality, follow these steps:

1. Clear all counters on your SCU-enabled router and verify they are empty.
2. Send a ping from the ingress CE router to the second CE router to generate SCU traffic across the SCU-enabled VPN route.
3. Verify that the counters are incrementing correctly on the outbound interface.

The following section shows the output of these commands used with the configuration example.

**Step One:
Clear the Counters**

```
user@pe2> clear interfaces statistics all

user@pe2> show interfaces so-0/0/0.0 statistics
Logical interface so-0/0/0.0 (Index 6) (SNMP ifIndex 113)
Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
Protocol inet, MTU: 4470
Source class
          GOLD1
          Packets
          Bytes
          0
          0
Addresses, Flags: Is-Preferred Is-Primary

user@pe2> show interfaces source-class GOLD1 so-0/0/0.0
Protocol inet
Source class
          GOLD1
          Packets
          Bytes
          0
          0
```

**Step Two:
Ping from the Edge Router**

```
user@ce1> ping 10.20.253.2 source 10.114.1.1 rapid count 10000
```

**Step Three:
Verify the Counters**

```
user@scu> show interfaces source-class GOLD1 so-0/0/0.0
Protocol inet
Source class
          GOLD1
          Packets
          Bytes
          20000
          1680000

user@scu> show interfaces so-0/0/0.0 statistics
Logical interface so-0/0/0.0 (Index 6) (SNMP ifIndex 113)
Flags: Point-To-Point SNMP-Traps Encapsulation: PPP
Protocol inet, MTU: 4470
Source class
          GOLD1
          Packets
          Bytes
          20000
          1680000
Addresses, Flags: Is-Preferred Is-Primary
Destination: 10.20.253/24, Local: 10.20.253.1
```

Configuring Accounting Profiles with SCU

Perhaps the most useful application of SCU is when a source class is combined with an accounting profile. Instead of using show commands to observe SCU data, you can use an accounting profile to capture this information in a log file. This allows you to monitor and view SCU traffic statistics at your convenience.

A class usage profile collects destination or source traffic class statistics for interfaces that have the appropriate traffic class feature enabled. The profile logs these statistics by means of a file specified at the [edit accounting-options] hierarchy level. This accounting profile is a superset of the previously released destination class profile. In the JUNOS 5.4 release, source class profiles and destination class profiles are part of class usage profiles.

A class usage profile can collect statistics for either source or destination class usage. The destination-classes and source-classes fields are mutually exclusive and indicate the type of statistics that are collected.

To enable SCU accounting profiles, perform these steps:

Configure Standard SCU on page 461

Associate an Accounting Profile with SCU Classes on page 461

To view a sample accounting profile log, see “Check Your Work” on page 462

Configure Standard SCU

To view the full standard SCU configuration steps, see “Configuring SCU” on page 441.

Associate an Accounting Profile with SCU Classes

Once your source classes are defined, implemented on the inbound and outbound interfaces, and applied to the forwarding table, you are ready to associate the source class with an accounting profile. Configure the accounting profile at the [edit accounting-options class-usage-profile] hierarchy level. You can associate either an SCU source class or a DCU destination class with the accounting profile. You can also specify the file name for the data capture, a class usage profile name, and an interval (in minutes) indicating how often you want the SCU information to be saved to the file.

```
[edit]
accounting-options {
  file filename;
  class-usage-profile profile-name {
    file filename;
    interval minutes;
    source-classes {
      source-class-name;
    }
    destination-classes {
      destination-class-name;
    }
  }
}
```



Caution

SCU accounting occurs on the outbound interface before output filter processing. If an SCU-marked packet is discarded in the router, the SCU counters can indicate more traffic than actually exists. You must use filter counters or traceoptions logs to ensure that all packets dropped by the SCU filter are recorded. If logged, you can subtract the discarded packets from the SCU counter tallies and calculate the true traffic profile.

Because DCU accounting occurs after the filtering process, DCU is unaffected by this disclaimer.

Check Your Work

To view the results of the SCU accounting profile you created, navigate to the `/var/log` directory of your router; it should contain the designated class usage profile log. The layout of an SCU profile looks like this:

```
profile_name,epoch-timestamp,interface-name,source-class-name,packet-count,byte-count
```

An example of the actual output from a profile looks like this:

```
scu_profile,980313078,ge-1/0/0.0,gold,82,6888
scu_profile,980313078,ge-1/0/0.0,silver,164,13776
scu_profile,980313078,ge-1/0/0.0,bronze,0,0
scu_profile,980313678,ge-1/0/0.0,gold,82,6888
scu_profile,980313678,ge-1/0/0.0,silver,246,20664
scu_profile,980313678,ge-1/0/0.0,bronze,0,0
```

To view the parameters of your SCU accounting profile, you can use the `show accounting-options class-usage-profile scu-profile-name` command.

For More Information

For additional information about SCU, SCU in Layer 3 VPNs, and SCU accounting profiles, see the following resources:

JUNOS Internet Software Configuration Guide: Network Management

JUNOS Internet Software Configuration Guide: Policy Framework

JUNOS Internet Software Configuration Guide: Routing and Routing Protocols

JUNOS Internet Software Configuration Guide: VPNs

Revision History

30 June 2003—6.0R1 Release. Richard Hendricks.

2 April 2003—5.7R1 Release. Richard Hendricks.

27 December 2002—Added SCU as a firewall term, 5.6R1 Release. Richard Hendricks.

30 September 2002—5.5R1 Release. Richard Hendricks.

19 July 2002—5.4R1 Release. Richard Hendricks.

6 May 2002—Initial document written. Richard Hendricks.