



**JUNOS**

## **Source Class Usage Feature Guide**

*Release 9.6*

**Juniper Networks, Inc.**  
1194 North Mathilda Avenue  
Sunnyvale, California 94089  
USA  
408-745-2000  
**[www.juniper.net](http://www.juniper.net)**

Published: 2009-07-12

This product includes the Envoy SNMP Engine, developed by Epilogue Technology, an Integrated Systems Company. Copyright © 1986-1997, Epilogue Technology Corporation. All rights reserved. This program and its documentation were developed at private expense, and no part of them is in the public domain.

This product includes memory allocation software developed by Mark Moraes, copyright © 1988, 1989, 1993, University of Toronto.

This product includes FreeBSD software developed by the University of California, Berkeley, and its contributors. All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by the Regents of the University of California. Copyright © 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994. The Regents of the University of California. All rights reserved.

GateD software copyright © 1995, the Regents of the University. All rights reserved. Gate Daemon was originated and developed through release 3.0 by Cornell University and its collaborators. Gated is based on Kirton's EGP, UC Berkeley's routing daemon (routed), and DCN's HELLO routing protocol. Development of Gated has been supported in part by the National Science Foundation. Portions of the GateD software copyright © 1988, Regents of the University of California. All rights reserved. Portions of the GateD software copyright © 1991, D. L. S. Associates.

This product includes software developed by Maker Communications, Inc., copyright © 1996, 1997, Maker Communications, Inc.

Juniper Networks, the Juniper Networks logo, JUNOS, NetScreen, ScreenOS, and Steel-Belted Radius are registered trademarks of Juniper Networks, Inc. in the United States and other countries. JUNOSe is a trademark of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Products made or sold by Juniper Networks or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

*JUNOS Source Class Usage Feature Guide.*

Release 9.6

Copyright © 2009, Juniper Networks, Inc.

All rights reserved. Printed in USA.

Writing: Merisha Wazna, Fawn Damitio, and Richard Hendricks

Editing: Sonia Saruba

Illustration: Faith Bradford, Fawn Damitio, Nathaniel Woodward, and Richard Hendricks

Cover Design: Edmonds Design

Revision History

July 2009—R1 JUNOS 9.6

The information in this document is current as of the date listed in the revision history.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. The JUNOS Software has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

## END USER LICENSE AGREEMENT

**READ THIS END USER LICENSE AGREEMENT (“AGREEMENT”) BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE.** BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE OR OTHERWISE EXPRESSING YOUR AGREEMENT TO THE TERMS CONTAINED HEREIN, YOU (AS CUSTOMER OR IF YOU ARE NOT THE CUSTOMER, AS A REPRESENTATIVE/AGENT AUTHORIZED TO BIND THE CUSTOMER) CONSENT TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT OR CANNOT AGREE TO THE TERMS CONTAINED HEREIN, THEN (A) DO NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND (B) YOU MAY CONTACT JUNIPER NETWORKS REGARDING LICENSE TERMS.

1. **The Parties.** The parties to this Agreement are (i) Juniper Networks, Inc. (if the Customer’s principal office is located in the Americas) or Juniper Networks (Cayman) Limited (if the Customer’s principal office is located outside the Americas) (such applicable entity being referred to herein as “Juniper”), and (ii) the person or organization that originally purchased from Juniper or an authorized Juniper reseller the applicable license(s) for use of the Software (“Customer”) (collectively, the “Parties”).
2. **The Software.** In this Agreement, “Software” means the program modules and features of the Juniper or Juniper-supplied software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller, or which was embedded by Juniper in equipment which Customer purchased from Juniper or an authorized Juniper reseller. “Software” also includes updates, upgrades and new releases of such software. “Embedded Software” means Software which Juniper has embedded in or loaded onto the Juniper equipment and any updates, upgrades, additions or replacements which are subsequently embedded in or loaded onto the equipment.
3. **License Grant.** Subject to payment of the applicable fees and the limitations and restrictions set forth herein, Juniper grants to Customer a non-exclusive and non-transferable license, without right to sublicense, to use the Software, in executable form only, subject to the following use restrictions:
  - a. Customer shall use Embedded Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller.
  - b. Customer shall use the Software on a single hardware chassis having a single processing unit, or as many chassis or processing units for which Customer has paid the applicable license fees; provided, however, with respect to the Steel-Belted Radius or Odyssey Access Client software only, Customer shall use such Software on a single computer containing a single physical random access memory space and containing any number of processors. Use of the Steel-Belted Radius or IMS AAA software on multiple computers or virtual machines (e.g., Solaris zones) requires multiple licenses, regardless of whether such computers or virtualizations are physically contained on a single chassis.
  - c. Product purchase documents, paper or electronic user documentation, and/or the particular licenses purchased by Customer may specify limits to Customer’s use of the Software. Such limits may restrict use to a maximum number of seats, registered endpoints, concurrent users, sessions, calls, connections, subscribers, clusters, nodes, realms, devices, links, ports or transactions, or require the purchase of separate licenses to use particular features, functionalities, services, applications, operations, or capabilities, or provide throughput, performance, configuration, bandwidth, interface, processing, temporal, or geographical limits. In addition, such limits may restrict the use of the Software to managing certain kinds of networks or require the Software to be used only in conjunction with other specific Software. Customer’s use of the Software shall be subject to all such limitations and purchase of all applicable licenses.
  - d. For any trial copy of the Software, Customer’s right to use the Software expires 30 days after download, installation or use of the Software. Customer may operate the Software after the 30-day trial period only if Customer pays for a license to do so. Customer may not extend or create an additional trial period by re-installing the Software after the 30-day trial period.
  - e. The Global Enterprise Edition of the Steel-Belted Radius software may be used by Customer only to manage access to Customer’s enterprise network. Specifically, service provider customers are expressly prohibited from using the Global Enterprise Edition of the Steel-Belted Radius software to support any commercial network access services.

The foregoing license is not transferable or assignable by Customer. No license is granted herein to any user who did not originally purchase the applicable license(s) for the Software from Juniper or an authorized Juniper reseller.

4. **Use Prohibitions.** Notwithstanding the foregoing, the license provided herein does not permit the Customer to, and Customer agrees not to and shall not: (a) modify, unbundle, reverse engineer, or create derivative works based on the Software; (b) make unauthorized copies of the Software (except as necessary for backup purposes); (c) rent, sell, transfer, or grant any rights in and to any copy of the Software, in any form, to any third party; (d) remove any proprietary notices, labels, or marks on or in any copy of the Software or any product in which the Software is embedded; (e) distribute any copy of the Software to any third party, including as may be embedded in Juniper equipment sold in the secondhand market; (f) use any ‘locked’ or key-restricted feature, function, service, application, operation, or capability without first purchasing the applicable license(s) and obtaining a valid key from Juniper, even if such feature, function, service, application, operation, or capability is enabled without a key; (g) distribute any key for the Software provided by Juniper to any third party; (h) use the Software in any manner that extends or is broader than the uses purchased by Customer from Juniper or an authorized Juniper reseller; (i) use Embedded Software on non-Juniper equipment; (j) use Embedded Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; (k) disclose the results of testing or benchmarking of the Software to any third party without the prior written consent of Juniper; or (l) use the Software in any manner other than as expressly provided herein.
5. **Audit.** Customer shall maintain accurate records as necessary to verify compliance with this Agreement. Upon request by Juniper, Customer shall furnish such records to Juniper and certify its compliance with this Agreement.

6. **Confidentiality.** The Parties agree that aspects of the Software and associated documentation are the confidential property of Juniper. As such, Customer shall exercise all reasonable commercial efforts to maintain the Software and associated documentation in confidence, which at a minimum includes restricting access to the Software to Customer employees and contractors having a need to use the Software for Customer's internal business purposes.
7. **Ownership.** Juniper and Juniper's licensors, respectively, retain ownership of all right, title, and interest (including copyright) in and to the Software, associated documentation, and all copies of the Software. Nothing in this Agreement constitutes a transfer or conveyance of any right, title, or interest in the Software or associated documentation, or a sale of the Software, associated documentation, or copies of the Software.
8. **Warranty, Limitation of Liability, Disclaimer of Warranty.** The warranty applicable to the Software shall be as set forth in the warranty statement that accompanies the Software (the "Warranty Statement"). Nothing in this Agreement shall give rise to any obligation to support the Software. Support services may be purchased separately. Any such support shall be governed by a separate, written support services agreement. TO THE MAXIMUM EXTENT PERMITTED BY LAW, JUNIPER SHALL NOT BE LIABLE FOR ANY LOST PROFITS, LOSS OF DATA, OR COSTS OR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, THE SOFTWARE, OR ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. IN NO EVENT SHALL JUNIPER BE LIABLE FOR DAMAGES ARISING FROM UNAUTHORIZED OR IMPROPER USE OF ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. EXCEPT AS EXPRESSLY PROVIDED IN THE WARRANTY STATEMENT TO THE EXTENT PERMITTED BY LAW, JUNIPER DISCLAIMS ANY AND ALL WARRANTIES IN AND TO THE SOFTWARE (WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE), INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT DOES JUNIPER WARRANT THAT THE SOFTWARE, OR ANY EQUIPMENT OR NETWORK RUNNING THE SOFTWARE, WILL OPERATE WITHOUT ERROR OR INTERRUPTION, OR WILL BE FREE OF VULNERABILITY TO INTRUSION OR ATTACK. In no event shall Juniper's or its suppliers' or licensors' liability to Customer, whether in contract, tort (including negligence), breach of warranty, or otherwise, exceed the price paid by Customer for the Software that gave rise to the claim, or if the Software is embedded in another Juniper product, the price paid by Customer for such other product. Customer acknowledges and agrees that Juniper has set its prices and entered into this Agreement in reliance upon the disclaimers of warranty and the limitations of liability set forth herein, that the same reflect an allocation of risk between the Parties (including the risk that a contract remedy may fail of its essential purpose and cause consequential loss), and that the same form an essential basis of the bargain between the Parties.
9. **Termination.** Any breach of this Agreement or failure by Customer to pay any applicable fees due shall result in automatic termination of the license granted herein. Upon such termination, Customer shall destroy or return to Juniper all copies of the Software and related documentation in Customer's possession or control.
10. **Taxes.** All license fees payable under this agreement are exclusive of tax. Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software. If applicable, valid exemption documentation for each taxing jurisdiction shall be provided to Juniper prior to invoicing, and Customer shall promptly notify Juniper if their exemption is revoked or modified. All payments made by Customer shall be net of any applicable withholding tax. Customer will provide reasonable assistance to Juniper in connection with such withholding taxes by promptly: providing Juniper with valid tax receipts and other required documentation showing Customer's payment of any withholding taxes; completing appropriate applications that would reduce the amount of withholding tax to be paid; and notifying and assisting Juniper in any audit or tax proceeding related to transactions hereunder. Customer shall comply with all applicable tax laws and regulations, and Customer will promptly pay or reimburse Juniper for all costs and damages related to any liability incurred by Juniper as a result of Customer's non-compliance or delay with its responsibilities herein. Customer's obligations under this Section shall survive termination or expiration of this Agreement.
11. **Export.** Customer agrees to comply with all applicable export laws and restrictions and regulations of any United States and any applicable foreign agency or authority, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. Customer shall be liable for any such violations. The version of the Software supplied to Customer may contain encryption or other capabilities restricting Customer's ability to export the Software without an export license.
12. **Commercial Computer Software.** The Software is "commercial computer software" and is provided with restricted rights. Use, duplication, or disclosure by the United States government is subject to restrictions set forth in this Agreement and as provided in DFARS 227.7201 through 227.7202-4, FAR 12.212, FAR 27.405(b)(2), FAR 52.227-19, or FAR 52.227-14(ALT III) as applicable.
13. **Interface Information.** To the extent required by applicable law, and at Customer's written request, Juniper shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of applicable fee, if any. Customer shall observe strict obligations of confidentiality with respect to such information and shall use such information in compliance with any applicable terms and conditions upon which Juniper makes such information available.
14. **Third Party Software.** Any licensor of Juniper whose software is embedded in the Software and any supplier of Juniper whose products or technology are embedded in (or services are accessed by) the Software shall be a third party beneficiary with respect to this Agreement, and such licensor or vendor shall have the right to enforce this Agreement in its own name as if it were Juniper. In addition, certain third party software may be provided with the Software and is subject to the accompanying license(s), if any, of its respective owner(s). To the extent portions of the Software are distributed under and subject to open source licenses obligating Juniper to make the source code for such portions publicly available (such as the GNU General Public License ("GPL") or the GNU Library General Public License ("LGPL")), Juniper will make such source code portions (including Juniper modifications, as appropriate) available upon request for a period of up to three years from the date of distribution. Such request can be made in writing to Juniper Networks, Inc., 1194 N. Mathilda Ave., Sunnyvale, CA 94089, ATTN: General Counsel. You may obtain a copy of the GPL at <http://www.gnu.org/licenses/gpl.html>, and a copy of the LGPL at <http://www.gnu.org/licenses/lgpl.html>.
15. **Miscellaneous.** This Agreement shall be governed by the laws of the State of California without reference to its conflicts of laws principles. The provisions of the U.N. Convention for the International Sale of Goods shall not apply to this Agreement. For any disputes arising under this Agreement, the Parties hereby consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within Santa Clara County, California. This Agreement constitutes the entire and sole agreement between Juniper and the Customer with respect to the Software, and supersedes all prior and contemporaneous

agreements relating to the Software, whether oral or written (including any inconsistent terms contained in a purchase order), except that the terms of a separate written agreement executed by an authorized Juniper representative and Customer shall govern to the extent such terms are inconsistent or conflict with terms contained herein. No modification to this Agreement nor any waiver of any rights hereunder shall be effective unless expressly assented to in writing by the party to be charged. If any portion of this Agreement is held invalid, the Parties agree that such invalidity shall not affect the validity of the remainder of this Agreement. This Agreement and associated documentation has been written in the English language, and the Parties agree that the English version will govern. (For Canada: Les parties aux présentes confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattache, soient rédigés en langue anglaise. (Translation: The parties confirm that this Agreement and all related documentation is and will be in the English language)).



# Table of Contents

<b>Part 1</b>	<b>Source Class Usage</b>	
<b>Chapter 1</b>	<b>Source Class Usage Concepts and Reference Material</b>	<b>3</b>
	Overview of Source Class Usage .....	3
	Guidelines for Configuring SCU .....	4
	System Requirements for SCU .....	5
<b>Chapter 2</b>	<b>Configuring Source Class Usage</b>	<b>7</b>
	Roadmap for Configuring SCU .....	7
	Configuring Route Filters and Source Classes in a Routing Policy .....	8
	Applying the Policy to the Forwarding Table .....	8
	Enabling Accounting on Inbound and Outbound Interfaces .....	9
	Roadmap for Configuring SCU with Layer 3 VPNs .....	10
	Configuring Input SCU on the vt Interface of the Egress PE Router .....	10
	Mapping the SCU-Enabled vt Interface to the VRF Instance .....	11
	Configuring SCU on the Output Interface .....	12
	Associating an Accounting Profile with SCU Classes .....	12
	Verifying Your SCU Accounting Profile .....	13
<b>Chapter 3</b>	<b>Source Class Usage Configuration Examples</b>	<b>15</b>
	Merging Examples .....	15
	Example: SCU Configuration .....	16
	Example: SCU Configuration .....	16
	Verifying Your Work .....	19
	Example: SCU with Layer 3 VPNs Configuration .....	24
	Example: SCU in a Layer 3 VPN Configuration .....	25
	Verifying Your Work .....	31
<b>Part 2</b>	<b>Index</b>	
	Index .....	35



# List of Figures

## Part 1

### Source Class Usage

---

Chapter 1	<b>Source Class Usage Concepts and Reference Material</b>	<b>3</b>
	Figure 1: DCU/SCU Concept .....	3
Chapter 3	<b>Source Class Usage Configuration Examples</b>	<b>15</b>
	Figure 2: SCU Topology Diagram .....	16
	Figure 3: SCU in a Layer 3 VPN Topology Diagram .....	25



## **Part 1**

# **Source Class Usage**

- Source Class Usage Concepts and Reference Material on page 3
- Configuring Source Class Usage on page 7
- Source Class Usage Configuration Examples on page 15



## Chapter 1

# Source Class Usage Concepts and Reference Material

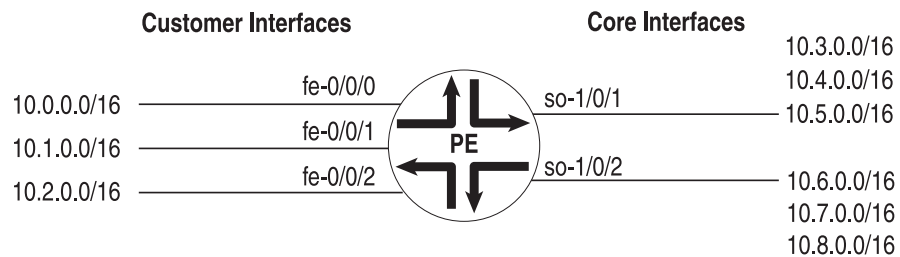
- Overview of Source Class Usage on page 3
- Guidelines for Configuring SCU on page 4
- System Requirements for SCU on page 5
- Terms and Acronyms for SCU on page 6

## Overview of Source Class Usage

---

Source class usage (SCU) is a logical extension of the destination class usage (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 1 on page 3 shows a service provider edge (PE) router diagram.

**Figure 1: DCU/SCU Concept**



9017168

The Fast Ethernet interfaces contain inbound traffic from customers, and the SONET/SDH 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 10.3.0.0/16 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.

- Related Topics**
- Source Class Usage Solutions Page
  - System Requirements for SCU on page 5
  - Roadmap for Configuring SCU on page 7
  - Roadmap for Configuring SCU with Layer 3 VPNs on page 10
  - Example: SCU Configuration on page 16
  - Example: SCU with Layer 3 VPNs Configuration on page 24

## Guidelines for Configuring SCU

---

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

- In JUNOS Release 5.6 and later for M Series routers only, 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 Policy Framework Configuration Guide*.
- 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 and is marked with the SCU class. 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. When the packet is ready to leave the router, the router detects that 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 because 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.

- Execute command line interface (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.
- On M320 and T Series routers, the source and destination classes are not carried across the platform fabric. For these routers, SCU and DCU accounting is performed before the packet enters the fabric and DCU is performed before output filters are evaluated.
- If an output filter drops traffic on M Series routers other than the M120 router and M320 router, the dropped packets are excluded from DCU statistics. If an output filter drops traffic on M320 and T Series routers, the dropped packets are included in DCU statistics.

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Roadmap for Configuring SCU on page 7
  - Example: SCU Configuration on page 16

## System Requirements for SCU

---

To implement SCU, your system must meet these requirements:

- JUNOS Release 8.2 or later for M120 and MX Series router support
- JUNOS Release 6.2 or later for IPv6 SCU
- JUNOS Release 5.6 or later to use a source class or a destination class as a match condition in a firewall filter
- JUNOS Release 5.4 or later for IPv4 SCU
- Three Juniper Networks M Series, MX Series, or T Series routers for basic SCU and five routers for SCU with Layer 3 VPNs. One router acts as a source class usage transit router, and the other routers are used to generate traffic or participate in the Layer 3 VPN.
- For M Series and T Series routers, a Tunnel Services PIC for SCU with Layer 3 VPNs

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - Roadmap for Configuring SCU on page 7
  - Roadmap for Configuring SCU with Layer 3 VPNs on page 10
  - Example: SCU Configuration on page 16
  - Example: SCU with Layer 3 VPNs Configuration on page 24

## Terms and Acronyms for SCU

---

### D

---

<b>destination address (DA)</b>	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.
<b>destination class usage (DCU)</b>	A method of grouping certain types of traffic and monitoring these groups through CLI <code>show</code> commands, accounting profiles, or SNMP. DCU uses a destination address lookup when determining group membership. For more information about DCU, see the <i>JUNOS Policy Framework Configuration Guide</i> .

### S

---

<b>source address (SA)</b>	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.
<b>source class usage (SCU)</b>	A method of grouping certain types of traffic and monitoring these groups through CLI <code>show</code> commands, accounting profiles, or SNMP. SCU uses a source address lookup when determining group membership. For more information about SCU, see the <i>JUNOS Policy Framework Configuration Guide</i> .

## Chapter 2

# Configuring Source Class Usage

- Roadmap for Configuring SCU on page 7
- Configuring Route Filters and Source Classes in a Routing Policy on page 8
- Applying the Policy to the Forwarding Table on page 8
- Enabling Accounting on Inbound and Outbound Interfaces on page 9
- Roadmap for Configuring SCU with Layer 3 VPNs on page 10
- Configuring Input SCU on the vt Interface of the Egress PE Router on page 10
- Mapping the SCU-Enabled vt Interface to the VRF Instance on page 11
- Configuring SCU on the Output Interface on page 12
- Associating an Accounting Profile with SCU Classes on page 12
- Verifying Your SCU Accounting Profile on page 13

### Roadmap for Configuring SCU

---

To configure source class usage (SCU), you must:

1. Create a routing policy that includes prefix route filters that indicate the IPv4 or IPv6 source addresses to monitor. See “Configuring Route Filters and Source Classes in a Routing Policy” on page 8.
2. Apply the filters to the forwarding table. See “Applying the Policy to the Forwarding Table” on page 8.
3. Enable accounting on the inbound and outbound interfaces. See “Enabling Accounting on Inbound and Outbound Interfaces” on page 9.

#### Related Topics

- Source Class Usage Solutions Page
- Overview of Source Class Usage on page 3
- System Requirements for SCU on page 5
- Example: SCU Configuration on page 16

## Configuring 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 IPv4 or IPv6 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, using the `forwarding-class` policy action, is even more flexible. It allows your IPv4 or IPv6 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 `then` 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 class-name;
}
```

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

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Roadmap for Configuring SCU on page 7
  - Example: SCU Configuration on page 16

## Applying 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;
}
```

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Roadmap for Configuring SCU on page 7
  - Example: SCU Configuration on page 16

## Enabling 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. SCU can be enabled in an IPv4 (family `inet`) or IPv6 (family `inet6`) network. To configure SCU accounting, include the `source-class-usage` statement at the [edit interfaces *interface-name* unit *logical-unit-number* family (inet | inet6) accounting] hierarchy level:

```
[edit]
interfaces {
  interface-name {
    unit unit-number {
      family (inet | inet6) {
        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.

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Roadmap for Configuring SCU on page 7
  - Example: SCU Configuration on page 16

## Roadmap for 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 provider edge (PE) router, use the standard procedure outlined in “Roadmap for Configuring SCU” on page 7. 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, as follows:

1. Configure SCU on the virtual loopback tunnel (vt) interface of the egress PE router. See “Configuring Input SCU on the vt Interface of the Egress PE Router” on page 10.
2. Map the SCU-enabled input interface of that router to the virtual routing and forwarding (VRF) instance. See “Mapping the SCU-Enabled vt Interface to the VRF Instance” on page 11.
3. Configure SCU on the output interface of the egress router. See “Configuring SCU on the Output Interface” on page 12.
4. Configure an accounting profile and associate the source class with that accounting profile. You can also specify the filename for the data capture, a class usage profile name, and an interval indicating how often you want the SCU information to be saved. See “Associating an Accounting Profile with SCU Classes” on page 12.



**NOTE:** SCU over Layer 3 VPNs is not supported when the VRF table label is configured. Also, SCU is not supported over Layer 2 VPNs.

---

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Example: SCU with Layer 3 VPNs Configuration on page 24

## Configuring 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 PE router that is either configured for or 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;
```



## Configuring 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;
        }
      }
    }
  }
}
```

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Roadmap for Configuring SCU with Layer 3 VPNs on page 10
  - Example: SCU with Layer 3 VPNs Configuration on page 24

## Associating 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 filename 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;
    }
  }
}
```

```
}
}
```



**NOTE:** 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.

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Roadmap for Configuring SCU with Layer 3 VPNs on page 10
  - Example: SCU with Layer 3 VPNs Configuration on page 24

## Verifying Your SCU Accounting Profile

**Purpose** To view the results of the SCU accounting profile you created.

**Action** 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.

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Associating an Accounting Profile with SCU Classes on page 12



## Chapter 3

# Source Class Usage Configuration Examples

- Merging Examples on page 15
- Example: SCU Configuration on page 16
- Example: SCU with Layer 3 VPNs Configuration on page 24

## Merging Examples

---

To merge a full example, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration example into a text file, save the file with a name, and copy the file to a directory on your router.

For example, copy the following configuration to a file and name the file `ex-script.conf`. Copy the `ex-script.conf` file to the `/var/tmp` directory on your router.

```
system {
  scripts {
    commit {
      file ex-script.xsl;
    }
  }
}
interfaces {
  fxp0 {
    disable;
    unit 0 {
      family inet {
        address 10.0.0.1/24;
      }
    }
  }
}
```

2. Merge the contents of the file into your router configuration by issuing the `load merge` configuration mode command:

```
[edit]
user@host# load merge /var/tmp/ex-script.conf
load complete
```

To merge a snippet, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration snippet into a text file, save the file with a name, and copy the file to a directory on your router.

For example, copy the following snippet to a file and name the file `ex-script-snippet.conf`. Copy the `ex-script-snippet.conf` file to the `/var/tmp` directory on your router.

```
commit {
  file ex-script-snippet.xml; }
```

2. Move to the hierarchy level that is relevant for this snippet by issuing the following configuration mode command:

```
[edit]
user@host# edit system scripts
[edit system scripts]
```

3. Merge the contents of the file into your router configuration by issuing the `load merge relative` configuration mode command:

```
[edit system scripts]
user@host# load merge relative /var/tmp/ex-script-snippet.conf
load complete
```

**Related Topics** For more information about the `load` command, see the *JUNOS CLI User Guide*.

## Example: SCU Configuration

- Example: SCU Configuration on page 16
- Verifying Your Work on page 19

### Example: SCU Configuration

**Figure 2: SCU Topology Diagram**

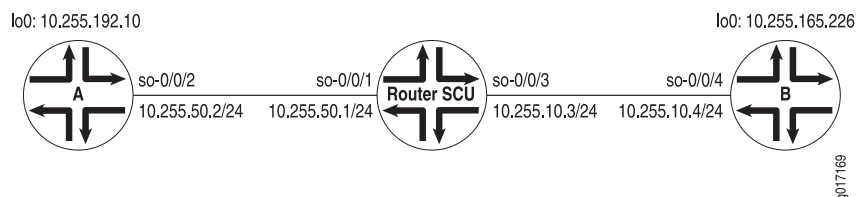


Figure 2 on page 16 shows a basic SCU configuration with three routers. Source routers A and B use loopback addresses as the prefixes to be monitored. Most of the configuration tasks and actual monitoring occurs on transit Router SCU.

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 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: [edit]
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;
    }
  }
}

```

## Verifying 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 perform the following 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.

```

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

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

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                               Packets          Bytes
          scu-class-b                       20000           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                               Packets          Bytes
          scu-class-a                       20000           1680000
          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 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
<b>scu-class-b</b>	<b>20000</b>	<b>1680000</b>

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, mp1s:
  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, mp1s:
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



## Example: SCU in a Layer 3 VPN Configuration

**Figure 3: SCU in a Layer 3 VPN Topology Diagram**

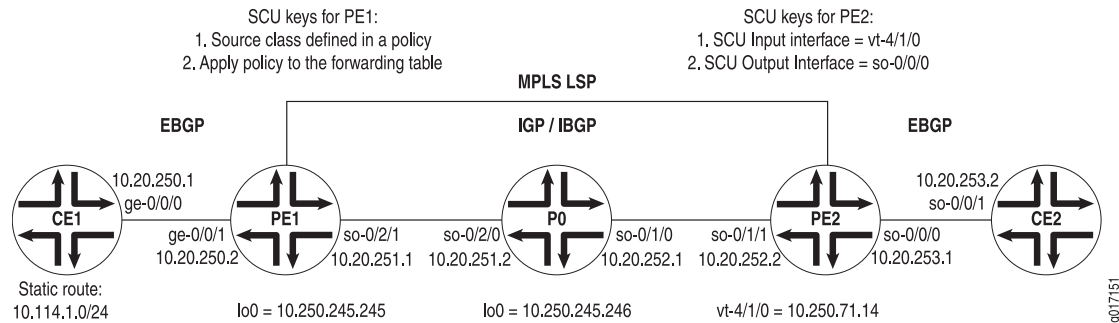


Figure 3 on page 25 displays a Layer 3 VPN topology. CE1 and CE2 are customer edge (CE) routers connected by a VPN through provider routers PE1, P0, and PE2. EBGP is established between routers CE1 and PE1, IBGP connects routers PE1 and PE2 over an IS-IS/MPLS/LDP core, and a second EBGP connection flows 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.

```

Router 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.

```

Router 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.

**Router 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.

```

Router 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.

```

Router 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;
  }
}

```

## Verifying 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.

```

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                                0                0                Bytes
Addresses, Flags: Is-Preferred Is-Primary

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

user@ce1> ping 10.20.253.2 source 10.114.1.1 rapid count 10000

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

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                                20000            1680000           Bytes
Addresses, Flags: Is-Preferred Is-Primary
Destination: 10.20.253/24, Local: 10.20.253.1

```

- Related Topics**
- Source Class Usage Solutions Page
  - Overview of Source Class Usage on page 3
  - System Requirements for SCU on page 5
  - Example: SCU with Layer 3 VPNs Configuration on page 24

## Part 2

# Index

- Index on page 35



# Index

## L

Layer 3 VPNs	
source class usage	
configuration procedure.....	10, 24
example configuration.....	25

## S

source class usage	
configuration procedure.....	7
example configuration.....	16
Layer 3 VPNs	
configuration procedure.....	10, 24
example configuration.....	25
operational mode commands.....	19, 31
overview.....	3
system requirements.....	5
system requirements	
source class usage.....	5

