Best Practices

SNMP-Based Network Management on Devices Running the Junos OS

Release

11.1

Published: 2011-01-19

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Best Practices SNMP-Based Network Management on Devices Running the Junos OS
Release 11.1

Revision History
January 2011—R1 Junos OS 11.1

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 OS 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 présentés confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattaché, soient redigés 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

Understanding SNMP Implementation in the Junos OS

A typical SNMP implementation includes three components:

- Managed device
- SNMP agent
- Network management system (NMS)

A managed device is any device on a network, also known as a network element, that is managed by the network management system. Routers and switches are common examples of managed devices. The SNMP agent is the SNMP process that resides on the managed device and communicates with the network management system. The NMS is a combination of hardware and software that is used to monitor and administer a network.

The SNMP data is stored in a highly-structured, hierarchical format known as a management information base (MIB). The MIB structure is based on a tree structure, which defines a grouping of objects into related sets. Each object in the MIB is associated with an object identifier (OID), which names the object. The “leaf” in the tree structure is the actual managed object instance, which represents a resource, event, or activity that occurs in your network device.

The SNMP agent exchanges network management information with SNMP manager software running on an NMS, or host. The agent responds to requests for information and actions from the manager. The agent also controls access to the agent’s MIB, the collection of objects that can be viewed or changed by the SNMP manager.

The SNMP manager collects information about network connectivity, activity, and events by polling managed devices.

Communication between the agent and the manager occurs in one of the following forms:

- **Get**, **GetBulk**, and **GetNext** requests—The manager requests information from the agent. The agent returns the information in a **Get** response message.
- **Set** requests—The manager changes the value of a MIB object controlled by the agent. The agent indicates status in a **Set** response message.
- **Traps** notification—The agent sends traps to notify the manager of significant events that occur on the network device.

The SNMP implementation in the Junos OS contains:

- A master SNMP agent (known as the SNMP process or snmpd) that resides on the managed device and is managed by the NMS or host.
- Various subagents that reside on different modules of the Junos OS, such as the Routing Engine, and are managed by the master SNMP agent (snmpd).

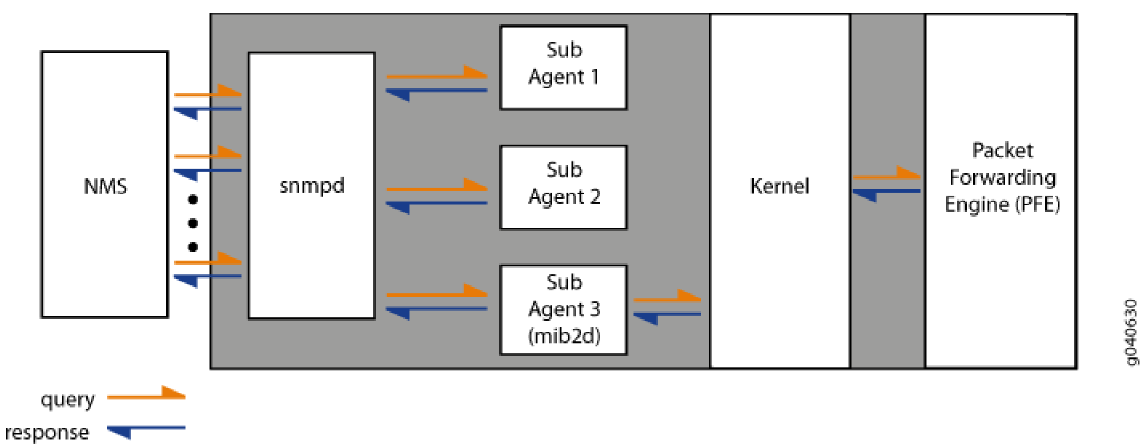
NOTE: By default, SNMP is not enabled on devices running the Junos OS. For information about enabling SNMP on a device running the Junos OS, see “Configuring SNMP on Devices Running the Junos OS” on page 5.

The SNMP implementation in the Junos OS uses both standard (developed by IETF and documented in RFCs) and enterprise-specific (developed and supported by specific vendors) MIBs.

In the Junos OS, the management data is maintained by the snmpd at one level (for example, snmpVacmMIB and snmpUsmMIB), and the subagents at the next level (for example, routing MIBs and RMON MIBs). However, there is another level of data that is maintained neither by the master agent nor by the subagents. In such cases, the data is maintained by Junos OS processes that share the data with the subagents when polled for SNMP data. Interface-related MIBs and Firewall MIBs are good examples of data maintained by Junos OS processes.

When a network mangement system polls the master agent for data, the master agent immediately shares the data with the network mangement system if the requested data is available with the master agent or one of the subagents. However, if the requested data does not belong to those categories that are maintained by the master agent or the subagents, the subagent polls the Junos OS kernel or the process that maintains that data. On receiving the required data, the subagent passes the response back to the master agent, which in turn passes it to the NMS.

The following illustration shows the communication flow among the NMS, SNMP process (snmpd), SNMP subagents, and Junos OS processes.



When a significant event, most often an error or a failure, occurs on a network device, the SNMP agent sends notifications to the SNMP manager. The SNMP implementation in the Junos OS supports two types of notifications: traps and informs. *Traps* are unconfirmed notifications, whereas *informs* are confirmed notifications. Informs are supported only on devices that support SNMP version 3 (SNMPv3) configuration.

The Junos OS supports trap queuing to ensure that traps are not lost because of temporary unavailability of routes. Two types of queues, *destination queues* and a *throttle queue*, are formed to ensure delivery of traps and to control the trap traffic.

The Junos OS forms a destination queue when a trap to a particular destination is returned because the host is not reachable, and adds the subsequent traps to the same destination to the queue. The Junos OS checks for availability of routes every 30 seconds and sends the traps from the destination queue in a round-robin fashion.

If the trap delivery fails, the trap is added back to the queue, and the delivery attempt counter and the next delivery attempt timer for the queue are reset. Subsequent attempts occur at progressive intervals of 1 minute, 2 minutes, 4 minutes, and 8 minutes. The maximum delay between the attempts is 8 minutes, and the maximum number of attempts is 10. After 10 unsuccessful attempts, the destination queue and all the traps in the queue are deleted.

The Junos OS also has a throttle mechanism to control the number of traps (throttle threshold; default value of 500 traps) sent during a particular time period (throttle interval; default of 5 seconds) and to ensure consistency in trap traffic, especially when large number of traps are generated because of interface status changes. The throttle interval period begins when the first trap arrives at the throttle. All traps within the trap threshold are processed, and the traps beyond the threshold limit are queued.

The maximum size of trap queues—that is, throttle queue and destination queue put together—is 40,000. However, on EX Series Ethernet Switches, the maximum size of trap queue is 1,000. The maximum size of any one queue is 20,000 for devices other than EX Series Switches. On EX Series Switches, the maximum size of one queue is 500. When a trap is added to the throttle queue, or if the throttle queue has exceeded the maximum size, the trap is added back on top of the destination queue, and all subsequent attempts from the destination queue are stopped for a 30-second period, after which the destination queue restarts sending the traps.

Related Documentation

- Configuring SNMP on Devices Running the Junos OS on page 5
- Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS on page 9
- Optimizing the Network Management System Configuration for the Best Results on page 15
- Configuring Options on Managed Devices for Better SNMP Response Time on page 17
- Managing Traps and Informs on page 19
- Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage on page 23

Configuring SNMP on Devices Running the Junos OS

The following sections contain information about basic SNMP configuration and a few examples of configuring the basic SNMP operations on devices running the Junos OS:

Configuring Basic Settings for SNMPv1 and SNMPv2

By default, SNMP is not enabled on devices running the Junos OS. To enable SNMP on devices running the Junos OS, include the **community public** statement at the **[edit snmp]** hierarchy level.

Enabling SNMPv1 and SNMPv2 Get and GetNext Operations

```
[edit]
snmp {
  community public;
}
```

A community that is defined as public grants access to all MIB data to any client.

To enable SNMPv1 and SNMPv2 **Set** operations on the device, you must include the following statements at the **[edit snmp]** hierarchy level:

Enabling SNMPv1 and SNMPv2 Set Operations

```
[edit snmp]
view all {
  oid .1;
}
community private {
  view all;
  authorization read-write;
}
```

The following example shows the basic minimum configuration for SNMPv1 and SNMPv2 traps on a device:

Configuring SNMPv1 and SNMPv2 Traps

```
[edit snmp]
trap-group jnpr {
  targets {
    192.168.69.179;
  }
}
```

Configuring Basic Settings for SNMPv3

The following example shows the minimum SNMPv3 configuration for enabling **Get**, **GetNext**, and **Set** operations on a device (note that the configuration has authentication set to **md5** and privacy to **none**):

Enabling SNMPv3 Get, GetNext, and Set Operations

```
[edit snmp]
v3 {
  usm {
    local-engine {
```



```
            user jnpruser {
                authentication-md5 {
                    authentication-key "$9$guaDiQFnAuOQzevMWx7ikqP"; ## SECRET-DATA
                }
                privacy-none;
            }
        }
    }
    vacm {
        security-to-group {
            security-model usm {
                security-name jnpruser {
                    group grpnm;
                }
            }
        }
        access {
            group grpnm {
                default-context-prefix {
                    security-model any {
                        security-level authentication {
                            read-view all;
                            write-view all;
                        }
                    }
                }
            }
        }
    }
}
view all {
    oid .1;
}
```

The following example shows the basic configuration for SNMPv3 informs on a device (the configuration has authentication and privacy set to **none**):

Configuring SNMPv3 Informs

```
[edit snmp]
v3 {
    usm {
        remote-engine 00000063000100a2c0a845b3 {
            user RU2_v3_sha_none {
                authentication-none;
                privacy-none;
            }
        }
    }
    vacm {
        security-to-group {
            security-model usm {
                security-name RU2_v3_sha_none {
                    group g1_usm_auth;
                }
            }
        }
        access {
```



```
            group g1_usm_auth {
                default-context-prefix {
                    security-model usm {
                        security-level authentication {
                            read-view all;
                            write-view all;
                            notify-view all;
                        }
                    }
                }
            }
        }
    }
    target-address TA2_v3_sha_none {
        address 192.168.69.179;
        tag-list tl1;
        address-mask 255.255.252.0;
        target-parameters TP2_v3_sha_none;
    }
    target-parameters TP2_v3_sha_none {
        parameters {
            message-processing-model v3;
            security-model usm;
            security-level none;
            security-name RU2_v3_sha_none;
        }
        notify-filter nf1;
    }
    notify N1_all_tl1_informs {
        type inform; # Replace inform with trap to convert informs to traps.
        tag tl1;
    }
    notify-filter nf1 {
        oid .1 include;
    }
}
view all {
    oid .1 include;
}
```

You can convert the SNMPv3 informs to traps by setting the value of the **type** statement at the **[edit snmp v3 notify N1_all_tl1_informs]** hierarchy level to **trap** as shown in the following example:

Converting Informs to Traps

```
user@host# set snmp v3 notify N1_all_tl1_informs type trap
```

Configuring System Name, Location, Description, and Contact Information

The Junos OS enables you to include the name and location of the system, administrative contact information, and a brief description of the system in the SNMP configuration.

NOTE: Always keep the name, location, contact, and description information configured and updated for all your devices that are managed by SNMP.

The following example shows a typical configuration.

TIP: Use quotation marks to enclose the system name, contact, location, and description information that contain spaces.

```
[edit]
  snmp {
    name “snmp 001”; # Overrides the system name.
    contact “Juniper Berry, (650) 555 1234”; # Specifies the name and phone number of
      the administrator.
    location “row 11, rack C”; # Specifies the location of the device.
    description “M40 router with 8 FPCs” # Configures a description for the device.
  }
```

Related Documentation

- Understanding SNMP Implementation in the Junos OS on page 1
- Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS on page 9
- Optimizing the Network Management System Configuration for the Best Results on page 15
- Configuring Options on Managed Devices for Better SNMP Response Time on page 17
- Managing Traps and Informs on page 19
- Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage on page 23

Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS

The following sections contain information about monitoring the SNMP activity on devices running the Junos OS and identifying problems that might impact the SNMP performance on devices running the Junos OS:

Checking for MIB Objects Registered with the snmpd

For the SNMP process to be able to access data related to a MIB object, the MIB object must be registered with the snmpd. When an SNMP subagent comes online, it tries to register the associated MIB objects with the snmpd. The snmpd maintains a mapping of the objects and the subagents with which the objects are associated. However, the registration attempt fails occasionally, and the objects remain unregistered with the snmpd until the next time the subagent restarts and successfully registers the objects.

When a network management system polls for data related to objects that are not registered with the snmpd, the snmpd returns either a **noSuchName** error (for SNMPv1 objects) or a **noSuchObject** error (for SNMPv2 objects).

You can use the following commands to check for MIB objects that are registered with the snmpd:

- **show snmp registered-objects**—Creates a **/var/log/snmp_reg_objs** file that contains the list of registered objects and their mapping to various subagents.
- **file show /var/log/snmp_reg_objs**—Displays the contents of the **/var/log/snmp_reg_objs** file.

The following example shows the steps for creating and displaying the **/var/log/snmp_reg_objs** file:

```
user@host> show snmp registered-objects
user@host> file show /var/log/snmp_reg_objs
--------------------------------------------------------------
 Registered MIB Objects
 root_name =
--------------------------------------------------------------
.1.2.840.10006.300.43.1.1.1.1.2 (dot3adAggMACAddress) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.3 (dot3adAggActorSystemPriority) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.4 (dot3adAggActorSystemID) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.5 (dot3adAggAggregateOrIndividual)
(/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.6 (dot3adAggActorAdminKey) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.7 (dot3adAggActorOperKey) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.8 (dot3adAggPartnerSystemID) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.9 (dot3adAggPartnerSystemPriority)
```

```
(/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.10 (dot3adAggPartnerOperKey) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.1.1.11 (dot3adAggCollectorMaxDelay) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.1.2.1.1 (dot3adAggPortListPorts) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.2 (dot3adAggPortActorSystemPriority)
(/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.3 (dot3adAggPortActorSystemID) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.4 (dot3adAggPortActorAdminKey) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.5 (dot3adAggPortActorOperKey) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.6 (dot3adAggPortPartnerAdminSystemPriority)
(/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.7 (dot3adAggPortPartnerOperSystemPriority)
(/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.8 (dot3adAggPortPartnerAdminSystemID)
(/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.9 (dot3adAggPortPartnerOperSystemID)
(/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.10 (dot3adAggPortPartnerAdminKey) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.11 (dot3adAggPortPartnerOperKey) (/var/run/mib2d-11)
.1.2.840.10006.300.43.1.2.1.1.12 (dot3adAggPortSelectedAggID) (/var/run/mib2d-11)
---(more)---
```

NOTE: The **/var/log/snmp_reg_objs** file contains only those objects that are associated with Junos OS processes that are up and running and registered with the snmpd, at the time of executing the **show snmp registered-objects** command. If a MIB object related to a Junos OS process that is up and running is not shown in the list of registered objects, you might want to restart the software process to retry object registration with the snmpd.

Tracking SNMP Activity

SNMP tracing operations track activity of SNMP agents and record the information in log files. The logged event descriptions provide detailed information to help you solve problems faster. By default, the Junos OS does not trace any SNMP activity. To enable tracking of SNMP activities on a device running the Junos OS, include the **traceoptions** statement at the **[edit snmp]** hierarchy level.

A sample **traceoptions** configuration might look like:

```
[edit snmp]
set traceoptions flag all;
```

When the **traceoptions flag all** statement is included at the **[edit snmp]** hierarchy level, the following log files are created:

- snmpd
- mib2d
- rmopd

You can use the **show log *log-filename*** operational mode command to view the contents of the log file. In the snmpd log file (see the following example), a sequence of **>>>** represents an incoming packet, whereas a sequence of **<<<** represents an outgoing packet. Note that the request response pair might not follow any sequence if there are multiple network management systems polling the device at the same time. You can use the source and request ID combinations to match requests and responses. However, note that no response log is created in the log file if the SNMP master agent or the SNMP subagent has not responded to a request.

A careful analysis of the request-response time can help you identify and understand delayed responses.

Reviewing a Log File

The following example shows the output for the **show log snmpd** command:

```
user@host> show log snmpd
Apr 12 06:40:03 snmpd[7ee783df] >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Apr 12 06:40:03 snmpd[7ee783df] >>> Get-Bulk-Request
Apr 12 06:40:03 snmpd[7ee783df] >>>  Source:      10.209.63.42
Apr 12 06:40:03 snmpd[7ee783df] >>>  Destination: 10.209.2.242
Apr 12 06:40:03 snmpd[7ee783df] >>>  Version:     SNMPv2
Apr 12 06:40:03 snmpd[7ee783df] >>>  Request_id:  0x7ee783df
Apr 12 06:40:03 snmpd[7ee783df] >>>  Community:   public
Apr 12 06:40:03 snmpd[7ee783df] >>>  Non-repeaters:   0
Apr 12 06:40:03 snmpd[7ee783df] >>>  Max-repetitions: 10
Apr 12 06:40:03 snmpd[7ee783df] >>>   OID  : jnxContentsType.6.1.2.0
Apr 12 06:40:03 snmpd[7ee783df] >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Apr 12 06:40:03 snmpd[7ee783df]  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Apr 12 06:40:03 snmpd[7ee783df]  <<< Get-Response
Apr 12 06:40:03 snmpd[7ee783df]  <<<  Source:      10.209.63.42
Apr 12 06:40:03 snmpd[7ee783df]  <<<  Destination: 10.209.2.242
Apr 12 06:40:03 snmpd[7ee783df]  <<<  Version:     SNMPv2
Apr 12 06:40:03 snmpd[7ee783df]  <<<  Request_id:  0x7ee783df
Apr 12 06:40:03 snmpd[7ee783df]  <<<  Community:   public
Apr 12 06:40:03 snmpd[7ee783df]  <<<  Error:       status=0 / vb_index=0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.7.1.0.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxM10iFPC.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.7.1.1.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxChassisTempSensor.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.7.2.0.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxM10iFPC.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.7.2.1.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxChassisTempSensor.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.9.1.0.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxM10iRE.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.9.1.1.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxPCMCIACard.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.9.2.0.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxM10iRE.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
```

```
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.9.2.1.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxPCMCIACard.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.12.1.0.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxM10iHCM.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<
Apr 12 06:40:03 snmpd[7ee783df]  <<<   OID  : jnxContentsType.12.2.0.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<   type : Object
Apr 12 06:40:03 snmpd[7ee783df]  <<<   value: jnxM10iHCM.0
Apr 12 06:40:03 snmpd[7ee783df]  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Apr 12 06:40:03 snmpd[7ee783e0] >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Apr 12 06:40:03 snmpd[7ee783e0] >>> Get-Bulk-Request
Apr 12 06:40:03 snmpd[7ee783e0] >>>  Source:      10.209.63.42
Apr 12 06:40:03 snmpd[7ee783e0] >>>  Destination: 10.209.2.242
Apr 12 06:40:03 snmpd[7ee783e0] >>>  Version:     SNMPv2
Apr 12 06:40:03 snmpd[7ee783e0] >>>  Request_id:  0x7ee783e0
Apr 12 06:40:03 snmpd[7ee783e0] >>>  Community:   public
Apr 12 06:40:03 snmpd[7ee783e0] >>>  Non-repeaters:   0
Apr 12 06:40:03 snmpd[7ee783e0] >>>  Max-repetitions: 10
Apr 12 06:40:03 snmpd[7ee783e0] >>>   OID  : jnxContentsType.12.2.0.0
Apr 12 06:40:03 snmpd[7ee783e0] >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
……
……
```

Monitoring SNMP Statistics

The **show snmp statistics extensive** operational mode command provides you with an option to review SNMP traffic, including traps, on a device. Output for the **show snmp statistics extensive** command shows real-time values and can be used to monitor values such as throttle drops, currently active, max active, not found, time out, max latency, current queued, total queued, and overflows. You can identify slowness in SNMP responses by monitoring the currently active count, because a constant increase in the currently active count is directly linked to slow or no response to SNMP requests.

Sample Output for the show snmp statistics extensive Command

```
user@host> show snmp statistics extensive
  SNMP statistics:
   Input:
     Packets: 226656, Bad versions: 0, Bad community names: 0,
     Bad community uses: 0, ASN parse errors: 0,
     Too bigs: 0, No such names: 0, Bad values: 0,
     Read onlys: 0, General errors: 0,
     Total request varbinds: 1967606, Total set varbinds: 0,
     Get requests: 18478, Get nexts: 75794, Set requests: 0,
     Get responses: 0, Traps: 0,
     Silent drops: 0, Proxy drops: 0, Commit pending drops: 0,
     Throttle drops: 27084, Duplicate request drops: 0
   V3 Input:
     Unknown security models: 0, Invalid messages: 0
     Unknown pdu handlers: 0, Unavailable contexts: 0
     Unknown contexts: 0, Unsupported security levels: 0
     Not in time windows: 0, Unknown user names: 0
     Unknown engine ids: 0, Wrong digests: 0, Decryption errors: 0
   Output:
     Packets: 226537, Too bigs: 0, No such names: 0,
     Bad values: 0, General errors: 0,
     Get requests: 0, Get nexts: 0, Set requests: 0,
     Get responses: 226155, Traps: 382
```



```
SA Control Blocks:
  Total: 222984, Currently Active: 501, Max Active: 501,
  Not found: 0, Timed Out: 0, Max Latency: 25
SA Registration:
  Registers: 0, Deregisters: 0, Removes: 0
Trap Queue Stats:
  Current queued: 0, Total queued: 0, Discards: 0, Overflows: 0
Trap Throttle Stats:
  Current throttled: 0, Throttles needed: 0
Snmp Set Stats:
  Commit pending failures: 0, Config lock failures: 0
  Rpc failures: 0, Journal write failures: 0
  Mgd connect failures: 0, General commit failures: 0
```

Checking CPU Utilization

High CPU usage of the software processes that are being queried, such as snmpd or mib2d, is another factor that can lead to slow response or no response. You can use the **show system processes extensive** operational mode command to check the CPU usage levels of the Junos OS processes.

Sample Output of show system processes extensive Command

```
user@host> show system processes extensive
last pid:  1415;  load averages:  0.00,  0.00,  0.00  up 0+02:20:54    10:26:25
117 processes: 2 running, 98 sleeping, 17 waiting

Mem: 180M Active, 54M Inact, 39M Wired, 195M Cache, 69M Buf, 272M Free
Swap: 1536M Total, 1536M Free


  PID USERNAME   THR PRI NICE   SIZE    RES STATE    TIME   WCPU COMMAND
   11 root         1 171   52     0K    12K RUN    132:09 95.21% idle
 1184 root         1  97    0 35580K  9324K select   4:16  1.61% chassisd
  177 root         1  -8    0     0K    12K mdwait   0:51  0.00% md7
  119 root         1  -8    0     0K    12K mdwait   0:20  0.00% md4
   13 root         1 -20 -139     0K    12K WAIT     0:16  0.00% swi7: clock sio
 1373 root         1  96    0 15008K 12712K select   0:09  0.00% snmpd
 1371 root         1  96    0  9520K  5032K select   0:08  0.00% jdiameterd
   12 root         1 -40 -159     0K    12K WAIT     0:07  0.00% swi2: net
 1375 root         2  96    0 15016K  5812K select   0:06  0.00% pfed
   49 root         1  -8    0     0K    12K mdwait   0:05  0.00% md0
 1345 root         1  96    0 10088K  4480K select   0:05  0.00% l2ald
 1181 root         1  96    0  1608K   908K select   0:05  0.00% bslockd
   23 root         1 -68 -187     0K    12K WAIT     0:04  0.00% irq10: fxp1
   30 root         1 171   52     0K    12K pgzero   0:04  0.00% pagezero
 1344 root         1   4    0 39704K 11444K kqread   0:03  0.00% rpd
1205 root        1  96    0  3152K   912K select   0:03  0.00% license-check
 1372 root         1  96    0 28364K  6696K select   0:03  0.00% dcd
 1374 root         1  96    0 11764K  7632K select   0:02  0.00% mib2d
 1405 regress      1  96    0 15892K 11132K select   0:02  0.00% cli
  139 root         1  -8    0     0K    12K mdwait   0:02  0.00% md5
   22 root         1 -80 -199     0K    12K WAIT     0:02  0.00% irq9: cbb1 fxp0
 1185 root         1  96    0  4472K  2036K select   0:02  0.00% alarmd
    4 root         1  -8    0     0K    12K -        0:02  0.00% g_down
    3 root         1  -8    0     0K    12K -        0:02  0.00% g_up
   43 root         1 -16    0     0K    12K psleep   0:02  0.00% vmkmemdaemon
 1377 root         1  96    0  3776K  2256K select   0:01  0.00% irsd
   48 root         1 -16    0     0K    12K -        0:01  0.00% schedcpu
   99 root         1  -8    0     0K    12K mdwait   0:01  0.00% md3
  953 root         1  96    0  4168K  2428K select   0:01  0.00% eventd
 1364 root         1  96    0  4872K  2808K select   0:01  0.00% cfmd
```

```
  15 root        1 -16    0     0K    12K -        0:01  0.00% yarrow
1350 root        1  96    0 31580K  7248K select   0:01  0.00% cosd
1378 root        1  96    0 19776K  6292K select   0:01  0.00% lpdfd

...
```

Checking Kernel and Packet Forwarding Engine Response

As mentioned in “Understanding SNMP Implementation in the Junos OS” on page 1, some SNMP MIB data are maintained by the kernel or Packet Forwarding Engine. For such data to be available for the network management system, the kernel has to provide the required information to the SNMP subagent in mib2d. A slow response from the kernel can cause a delay in mib2d returning the data to the network management system. The Junos OS adds an entry in the mib2d log file every time that an interface takes more than 10,000 microseconds to respond to a request for interface statistics. You can use the **show log *log-filename* | grep “kernel response time”** command to find out the response time taken by the kernel.

Checking the Kernel Response Time

```
user@host> show log mib2d | grep “kernel response time”
 Aug 17 22:39:37  == kernel response time for
 COS_IPVPN_DEFAULT_OUTPUT-t1-7/3/0:10:27.0-o: 9.126471 sec, range
 (0.000007, 11.000806)

 Aug 17 22:39:53  == kernel response time for
 COS_IPVPN_DEFAULT_INPUT-t1-7/2/0:5:15.0-i: 5.387321 sec, range
 (0.000007, 11.000806)

 Aug 17 22:39:53  == kernel response time for ct1-6/1/0:9:15: 0.695406
 sec, range (0.000007, 11.000806)

 Aug 17 22:40:04  == kernel response time for t1-6/3/0:6:19: 1.878542
 sec, range (0.000007, 11.000806)

 Aug 17 22:40:22  == kernel response time for lsq-7/0/0: 2.556592 sec,
 range (0.000007, 11.000806)
```

Related Documentation

- Understanding SNMP Implementation in the Junos OS on page 1
- Configuring SNMP on Devices Running the Junos OS on page 5
- Optimizing the Network Management System Configuration for the Best Results on page 15
- Configuring Options on Managed Devices for Better SNMP Response Time on page 17
- Managing Traps and Informs on page 19
- Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage on page 23

Optimizing the Network Management System Configuration for the Best Results

You can modify your network management system configuration to optimize the response time for SNMP queries. The following sections contain a few tips on how you can configure the network management system:

Changing the Polling Method from Column-by-Column to Row-by-Row

You can configure the network management system to use the row-by-row method for SNMP data polling. It has been proven that the row-by-row and multiple row-by-multiple-row polling methods are more efficient than column-by-column polling. By configuring the network management system to use the row-by-row data polling method, you can ensure that data for only one interface is polled in a request instead of a single request polling data for multiple interfaces, as is the case with column-by-column polling. Row-by-row polling also reduces the risk of requests timing out.

Reducing the Number of Variable Bindings per PDU

By reducing the number of variable bindings per protocol data unit (PDU), you can improve the response time for SNMP requests. A request that polls for data related to multiple objects, which are mapped to different index entries, translates into multiple requests at the device-end because the subagent might have to poll different modules to obtain data that are linked to different index entries. The recommended method is to ensure that a request has only objects that are linked to one index entry instead of multiple objects linked to different index entries.

NOTE: If responses from a device are slow, avoid using the **GetBulk** option for the device, because a **GetBulk** request might contain objects that are linked to various index entries and might further increase the response time.

Increasing Timeout Values in Polling and Discovery Intervals

By increasing the timeout values for polling and discovery intervals, you can increase the queuing time at the device end and reduce the number of throttle drops that occur because of the request timing out.

Reducing Incoming Packet Rate at the snmpd

By reducing the frequency of sending SNMP requests to a device, you can reduce the risk of SNMP requests piling up at any particular device. Apart from reducing the frequency of sending SNMP requests to a device, you can also increase the polling interval, control the use of **GetNext** requests, and reduce the number of polling stations per device.

Related Documentation

- Understanding SNMP Implementation in the Junos OS on page 1
- Configuring SNMP on Devices Running the Junos OS on page 5
- Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS on page 9
- Configuring Options on Managed Devices for Better SNMP Response Time on page 17
- Managing Traps and Informs on page 19
- Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage on page 23

Configuring Options on Managed Devices for Better SNMP Response Time

The following sections contain information about configuration options on the managed devices that can enhance SNMP performance:

Enabling the stats-cache-lifetime Option

The Junos OS provides you with an option to configure the length of time an SNMP request stays active and queued so as to reduce the possibility of request drops during slow response times. You can use the **stats-cache-lifetime *seconds*** option at the **[edit snmp]** hierarchy level to specify the length of time that an SNMP request remains queued. The recommended value for the **stats-cache-lifetime** option is in the range of 30 to 60 seconds.

NOTE: The **set snmp stats-cache-lifetime *seconds*** command is a hidden command and is supported only on devices running Junos OS Release 9.3 and later.

Filtering Out Duplicate SNMP Requests

If a network management station retransmits a **Get**, **GetNext**, or **GetBulk** SNMP request too frequently to a device, that request might interfere with the processing of previous requests and slow down the response time of the agent. Filtering these duplicate requests improves the response time of the SNMP agent. The Junos OS enables you to filter out duplicate **Get**, **GetNext**, and **GetBulk** SNMP requests. The Junos OS uses the following information to determine if an SNMP request is a duplicate:

- Source IP address of the SNMP request
- Source UDP port of the SNMP request
- Request ID of the SNMP request

NOTE: By default, filtering of duplicate SNMP requests is disabled on devices running the Junos OS.

To enable filtering of duplicate SNMP requests on devices running the Junos OS, include the **filter-duplicates** statement at the **[edit snmp]** hierarchy level:

```
[edit snmp]
filter-duplicates;
```

Excluding Interfaces That Are Slow in Responding to SNMP Queries

An interface that is slow in responding to SNMP requests for interface statistics can delay kernel responses to SNMP requests. You can review the mib2d log file to find out how long the kernel takes to respond to various SNMP requests. For more information about reviewing the log file for kernel response data, see "Checking Kernel and Packet Forwarding Engine Response" under "Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS" on page 9. If you notice that a particular interface is slow in responding, and think that it is slowing down the kernel from responding to SNMP requests, exclude that interface from the SNMP queries to the device. You can exclude an interface from the SNMP queries either by configuring the **filter-interface** statement or by modifying the SNMP view settings.

The following example shows a sample configuration for excluding interfaces from the SNMP **Get**, **GetNext**, and **Set** operations:

```
[edit]
  snmp {
  filter-interfaces {
    interfaces { # exclude the specified interfaces
      interface1;
      interface2;
    }
    all-internal-interfaces; # exclude all internal interfaces
  }
}
```

The following example shows the SNMP view configuration for excluding the interface with an interface index (ifIndex) value of 312 from a request for information related to the ifTable and ifXtable objects:

```
[edit snmp]
  view test {
    oid .1 include;
    oid ifTable.1.*.312 exclude;
    oid ifXTable.1.*.312 exclude
  }
```

Alternatively, you can take the interface that is slow in responding offline.

Related Documentation

- Understanding SNMP Implementation in the Junos OS on page 1
- Configuring SNMP on Devices Running the Junos OS on page 5
- Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS on page 9
- Optimizing the Network Management System Configuration for the Best Results on page 15
- Managing Traps and Informs on page 19
- Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage on page 23

Managing Traps and Informs

The following sections contain a few tips on managing SNMP notifications:

Generating Traps Based on SysLog Events

Event policies can include an action that raises traps for events based on system log messages. This feature enables notification of an SNMP trap-based application when an important system log message occurs. You can convert any system log message,for which there is no corresponding trap, into a trap. If you are using network management system traps rather than system log messages to monitor your network, you can use this feature to ensure that you are notified of all the major events.

To configure a policy that raises a trap on receipt of an event, include the following statements at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name]
events [ events ];
then {
  raise-trap;
}
```

The following example shows the sample configuration for raising a trap for the event **ui_mgd_terminate**:

Generating Traps Based on SysLog Events

```
[edit event-options policy p1]
events ui_mgd_terminate;
then {
  raise-trap;
}
```

Filtering Traps Based on the Trap Category

SNMP traps are categorized into many categories. The Junos OS provides a configuration option, **categories** at the **[edit snmp trap-group *trap-group*]** hierarchy level, that enables you to specify categories of traps that you want to receive on a particular host. You can use this option when you want to monitor only specific modules of the Junos OS.

The following example shows a sample configuration for receiving only **link**, **vrrp-events**, **services**, and **otn-alarms** traps:

```
[edit snmp]
trap-group jnpr {
  categories {
    link;
    vrrp-events;
    services;
    otn-alarms;
  }
```



```
    targets {
      192.168.69.179;
    }
  }
```

Filtering Traps Based on the Object Identifier

The Junos OS also provides a more advanced filter option that enables you to filter out specific traps based on their object identifiers. You can use the **notify-filter** option to filter out a specific trap or a group of traps.

The following example shows the sample configuration for excluding Juniper Networks enterprise-specific configuration management traps (note that the SNMPv3 configuration also supports filtering of SNMPv1 and SNMPv2 traps as is shown in the following example):

```
[edit snmp]
v3 {
  vacm {
    security-to-group {
      security-model v2c {
        security-name sn_v2c_trap {
          group gr_v2c_trap;
        }
      }
    }
    access {
      group gr_v2c_trap {
        default-context-prefix {
          security-model v2c {
            security-level none {
              read-view all;
              notify-view all;
            }
          }
        }
      }
    }
  }
  target-address TA_v2c_trap {
    address 10.209.196.166;
    port 9001;
    tag-list tg1;
    target-parameters TP_v2c_trap;
  }
  target-parameters TP_v2c_trap {
    parameters {
      message-processing-model v2c;
      security-model v2c;
      security-level none;
      security-name sn_v2c_trap;
    }
    notify-filter nf1;
  }
  notify v2c_notify {
    type trap;
    tag tg1;
```



```
    }
    notify-filter nf1 {
        oid .1.3.6.1.4.1.2636.4.5 exclude;
        oid .1 include;
    }
    snmp-community index1 {
        community-name "$9$tDLl01h7Nbw2axN"; ## SECRET-DATA
        security-name sn_v2c_trap;
        tag tg1;
    }
    view all {
        oid .1 include;
    }
}
```

Related Documentation

- Understanding SNMP Implementation in the Junos OS on page 1
- Configuring SNMP on Devices Running the Junos OS on page 5
- Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS on page 9
- Optimizing the Network Management System Configuration for the Best Results on page 15
- Configuring Options on Managed Devices for Better SNMP Response Time on page 17
- Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage on page 23

Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage

Even though the Junos OS has built-in performance metrics and monitoring options, you might need to have customized performance metrics. To make it easier for you to monitor such customized data through a standard monitoring system, the Junos OS provides you with an enterprise-specific Utility MIB that can store such data and thus extend SNMP support for managing and monitoring the data of your choice.

The enterprise-specific Utility MIB provides you with container objects of the following types: **32-bit counters**, **64-bit counters**, **signed integers**, **unsigned integers**, and **octet strings**. You can use these container MIB objects to store the data that are otherwise not supported for SNMP operations. You can populate data for these objects either by using CLI commands or with the help of Op scripts and an RPC API that can invoke the CLI commands.

The following CLI commands enable you to set and clear Utility MIB object values:

- **request snmp utility-mib set instance *name* object-type <counter | counter 64 | integer | string | unsigned integer> object-value *value***
- **request snmp utility-mib clear instance *name* object-type <counter | counter 64 | integer | string | unsigned integer>**

The **instance *name*** option of the **request snmp utility-mib <set | clear>** command specifies the name of the data instance and is the main identifier of the data. The **object-type <counter | counter 64 | integer | string | unsigned integer>** option enables you specify the object type, and the **object-value *value*** option enables you to set the value of the object.

To automate the process of populating Utility MIB data, you can use a combination of an event policy and event script. The following examples show the configuration for an event policy to run **show system buffers** every hour and to store the **show system buffers** data in Utility MIB objects by running an event script (**check-mbufs.slax**).

Event Policy Configuration

To configure an event policy that runs the **show system buffers** command every hour and invokes **check-mbufs.slax** to store the **show system buffers** data into Utility MIB objects, include the following statements at the [**edit**] hierarchy level:

```
event-options {
  generate-event {
    1-HOUR time-interval 600;
  }
  policy MBUFS {
    events 1-HOUR;
    then {
      event-script check-mbufs.slax; # script stored at /var/db/scripts/event/
    }
  }
  event-script {
    file check-mbufs.slax;
  }
}
```

check-mbufs.slax Script

The following example shows the **check-mbufs.slax** script that is stored under **/var/db/scripts/event/**:

```
------ script START ------
version 1.0;

ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";
ns ext = "http://xmlsoft.org/XSLT/namespace";

match / {
    <op-script-results>{
 var $cmd = <command> "show system buffers";
        var $out = jcs:invoke($cmd);

     var $lines = jcs:break_lines($out);
     for-each ($lines) {
         if (contains(., "current/peak/max")) {
             var $pattern = "([0-9]+)/([0-9]+)/([0-9]+) mbufs";
             var $split = jcs:regex($pattern, .);
             var $result = $split[2];

             var $rpc = <request-snmp-utility-mib-set> {
                 <object-type> "integer";
                 <instance> "current-mbufs";
                 <object-value> $result;
             }
             var $res = jcs:invoke($rpc);
                }
        }
    }
}
------ script END ------
```

You can run the following command to check the data stored in the Utility MIB as a result of the event policy and script shown in the preceding examples:

```
user@host> show snmp mib walk jnxUtilData ascii jnxUtilIntegerValue."current-mbufs"
= 0 jnxUtilIntegerTime."current-mbufs" = 07 da 05 0c 03 14 2c 00 2d 07 00
regress@caramels>
```

Related Documentation

- Understanding SNMP Implementation in the Junos OS on page 1
- Configuring SNMP on Devices Running the Junos OS on page 5
- Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running the Junos OS on page 9
- Optimizing the Network Management System Configuration for the Best Results on page 15
- Configuring Options on Managed Devices for Better SNMP Response Time on page 17
- Managing Traps and Informs on page 19

