SNMP-Based Network Management on Devices Running the Junos OS
Table of Contents

About the Documentation | v
Documentation and Release Notes | v
Documentation Conventions | v
Documentation Feedback | viii
Requesting Technical Support | viii
  Self-Help Online Tools and Resources | ix
  Creating a Service Request with JTAC | ix

Overview

Introduction to Understanding SNMP Implementation | 2
Understanding SNMP Implementation in Junos OS | 2
  SNMP Architecture | 2
    SNMP MIBs | 3
    SNMP Manager and Agent Authentication and Communication | 3
    SNMP Traps and Informs | 4
SNMP on Junos OS | 4
  Junos OS Support of SNMP Versions | 5
  System Logging Severity Levels for SNMP Traps | 6
  SNMP Communication Flow | 6
  Trap Queuing | 6

Configuration

Configuring SNMP on Junos Devices | 9
Best Practices for Configuring SNMP | 9
  Configuring Basic Settings for SNMPv1 and SNMPv2 | 9
  Configuring Basic Settings for SNMPv3 | 10
  Configuring System Name, Location, Description, and Contact Information | 13
Configuring Options on Managed Devices for Better SNMP Response Time | 14
  Enabling the stats-cache-lifetime Option | 15
  Filtering Out Duplicate SNMP Requests | 15
Network Configuration Examples, Release 13.1

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at https://www.juniper.net/documentation/.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at https://www.juniper.net/books.

Documentation Conventions

Table 1 on page vi defines notice icons used in this guide.
### Table 1: Notice Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Info" /></td>
<td>Informational note</td>
<td>Indicates important features or instructions.</td>
</tr>
<tr>
<td><img src="image" alt="Caution" /></td>
<td>Caution</td>
<td>Indicates a situation that might result in loss of data or hardware damage.</td>
</tr>
<tr>
<td><img src="image" alt="Warning" /></td>
<td>Warning</td>
<td>Alerts you to the risk of personal injury or death.</td>
</tr>
<tr>
<td><img src="image" alt="Laser" /></td>
<td>Laser warning</td>
<td>Alerts you to the risk of personal injury from a laser.</td>
</tr>
<tr>
<td><img src="image" alt="Tip" /></td>
<td>Tip</td>
<td>Indicates helpful information.</td>
</tr>
<tr>
<td><img src="image" alt="Best Practice" /></td>
<td>Best practice</td>
<td>Alerts you to a recommended use or implementation.</td>
</tr>
</tbody>
</table>

### Table 2: Text and Syntax Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| **Bold text like this** | Represents text that you type. | To enter configuration mode, type the `configure` command:  
user@host> `configure` |
| **Fixed-width text like this** | Represents output that appears on the terminal screen. | user@host> `show chassis alarms`  
No alarms currently active |
| **Italic text like this** | • Introduces or emphasizes important new terms.  
• Identifies guide names.  
• Identifies RFC and Internet draft titles. | • A policy term is a named structure that defines match conditions and actions.  
• *Junos OS CLI User Guide*  
• RFC 1997, *BGP Communities Attribute* |
<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Italic text like this</em></td>
<td>Represents variables (options for which you substitute a value) in commands or configuration statements.</td>
<td>Configure the machine’s domain name:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[edit]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>root@# set system domain-name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>domain-name</td>
</tr>
<tr>
<td><em>Text like this</em></td>
<td>Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.</td>
<td>• To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The console port is labeled CONSOLE.</td>
</tr>
<tr>
<td>&lt; &gt; (angle brackets)</td>
<td>Encloses optional keywords or variables.</td>
<td>stub &lt;default-metric metric&gt;</td>
</tr>
<tr>
<td></td>
<td>(pipe symbol)</td>
<td>Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.</td>
</tr>
<tr>
<td></td>
<td>(string1</td>
<td>string2</td>
</tr>
<tr>
<td></td>
<td>(pound sign)</td>
<td>Indicates a comment specified on the same line as the configuration statement to which it applies.</td>
</tr>
<tr>
<td>[] (square brackets)</td>
<td>Encloses a variable for which you can substitute one or more values.</td>
<td>community name members [ community-ids ]</td>
</tr>
<tr>
<td>Indention and braces ([ ] )</td>
<td>Identifies a level in the configuration hierarchy.</td>
<td>[edit] routing-options { static { route default { nexthop address; retain; } } }</td>
</tr>
<tr>
<td>; (semicolon)</td>
<td>Identifies a leaf statement at a configuration hierarchy level.</td>
<td></td>
</tr>
</tbody>
</table>

**GUI Conventions**
Table 2: Text and Syntax Conventions (continued)

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| **Bold text like this** | Represents graphical user interface (GUI) items you click or select. | • In the Logical Interfaces box, select **All Interfaces**.  
• To cancel the configuration, click **Cancel**. |
| > (bold right angle bracket) | Separates levels in a hierarchy of menu selections. | In the configuration editor hierarchy, select **Protocols>Osfp**. |

**Documentation Feedback**

We encourage you to provide feedback so that we can improve our documentation. You can use either of the following methods:

- Online feedback system—Click TechLibrary Feedback, on the lower right of any page on the Juniper Networks TechLibrary site, and do one of the following:
  - Click the thumbs-up icon if the information on the page was helpful to you.
  - Click the thumbs-down icon if the information on the page was not helpful to you or if you have suggestions for improvement, and use the pop-up form to provide feedback.
  
- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

**Requesting Technical Support**

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active Juniper Care or Partner Support Services support contract, or are
covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- Product warranties—For product warranty information, visit https://www.juniper.net/support/warranty/.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: https://www.juniper.net/customers/support/
- Search for known bugs: https://prsearch.juniper.net/
- Find product documentation: https://www.juniper.net/documentation/
- Find solutions and answer questions using our Knowledge Base: https://kb.juniper.net/
- Download the latest versions of software and review release notes: https://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: https://kb.juniper.net/InfoCenter/
- Join and participate in the Juniper Networks Community Forum: https://www.juniper.net/company/communities/
- Create a service request online: https://myjuniper.juniper.net

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: https://entitlementsearch.juniper.net/entitlementsearch/

Creating a Service Request with JTAC

You can create a service request with JTAC on the Web or by telephone.

- Visit https://myjuniper.juniper.net.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see https://support.juniper.net/support/requesting-support/.
Overview

Introduction to Understanding SNMP Implementation | 2
A typical SNMP implementation includes three components:

- Network management system (NMS)—A combination of hardware (devices) and software (the SNMP manager) that is used to monitor and administer a network. The manager polls the devices on your network how ever often you specify for information about network connectivity, activity, and events.

- Managed device—A managed device (also called a network element) is any device on a network that is managed by the NMS. Routers and switches are common examples of managed devices.
SNMP agent—The SNMP agent is the SNMP process that resides on the managed device and
communicates with the NMS. The SNMP agent exchanges network management information with the
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.

This topic contains the following sections:

SNMP MIBs

SNMP data is stored in a highly structured, hierarchical format known as a management information base
(MIB). A MIB defines managed objects in a network device.

The MIB structure is based on a tree structure and 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.

MIBs are either standard or enterprise-specific. Standard MIBs are created by the Internet Engineering
Task Force (IETF) and documented in various RFCs. Depending on the vendor, many standard MIBs are
delivered with the NMS software. You can also download the standard MIBs from the IETF website,
www.ietf.org, and compile them into your NMS, if necessary.

For a list of standard supported MIBs, see Standard SNMP MIBs Supported by Junos OS.

Enterprise-specific MIBs are developed and supported by a specific equipment manufacturer. If your
network contains devices that have enterprise-specific MIBs, you must obtain them from the manufacturer
and compile them into your network management software.

For a list of Juniper Networks enterprise-specific supported MIBs, see Enterprise-Specific SNMP MIBs
Supported by Junos OS.

SNMP Manager and Agent Authentication and Communication

SNMP uses a very basic form of authentication called community strings to control access between a
manager and remote agents. Community strings are administrative names used to group collections of
devices (and the agents running on them) into common management domains. If a manager and an agent
share the same community, they can talk to one another. Many people associate SNMP community strings
with passwords and keys because the jobs they do are similar. As a result, SNMP communities are
traditionally referred to as strings.

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.

**SNMP Traps and Informs**

Routers can send notifications to SNMP managers when significant events occur on a network device, most often errors or failures. SNMP notifications can be sent as traps or inform requests. SNMP traps are unconfirmed notifications. SNMP informs are confirmed notifications.

SNMP traps are defined in either standard or enterprise-specific MIBs. Standard traps are created by the IETF and documented in various RFCs. The standard traps are compiled into the network management software. You can also download the standard traps from the IETF website, [www.ietf.org](http://www.ietf.org).

For more information about standard traps supported by the Junos OS, see [Standard SNMP Traps Supported on Devices Running Junos OS](#).

Enterprise-specific traps are developed and supported by a specific equipment manufacturer. If your network contains devices that have enterprise-specific traps, you must obtain them from the manufacturer and compile them into your network management software.

For more information about enterprise-specific traps supported by the Junos OS, see [Enterprise-Specific SNMP Traps Supported by Junos OS](#). For information about system logging severity levels for SNMP traps, see “[System Logging Severity Levels for SNMP Traps](#)” on page 6.

With traps, the receiver does not send any acknowledgment when it receives a trap, and the sender cannot determine if the trap was received. To increase reliability, SNMP informs are supported in SNMPv3. An SNMP manager that receives an inform acknowledges the message with a response. For information about SNMP informs, see [Configuring SNMP Informs](#).

**SNMP on Junos OS**

---

### IN THIS SECTION

- Junos OS Support of SNMP Versions | 5
- System Logging Severity Levels for SNMP Traps | 6
- SNMP Communication Flow | 6
- Trap Queuing | 6

---

On Junos OS, SNMP uses both standard (developed by IETF and documented in RFCs) and Juniper Networks enterprise-specific MIBs.
NOTE: By default, SNMP is not enabled on devices running Junos OS.

In Junos OS, the processes that maintain the SNMP management data include the following:

- A master SNMP agent which resides on the managed device and is managed by the NMS, or host.
  The Junos OS SNMP agent software consists of an SNMP primary agent (known as the SNMP process, or snmpd). It resides on the managed device and is managed by the NMS, or host.
- Various subagents that reside on different modules of Junos OS, such as the Routing Engine. The master SNMP agent delegates all SNMP requests to the subagents. Each subagent is responsible for the support of a specific set of MIBs.
- Junos OS processes that share data with the subagents when polled for SNMP data (for example, interface-related MIBs).

The community string is the first level of management authentication implemented by the SNMP agent in Junos OS.

See the following sections for more information.

**Junos OS Support of SNMP Versions**

The Junos OS supports the following versions of SNMP:

- SNMPv1—The initial implementation of SNMP that defines the architecture and framework for SNMP.
- SNMPv2c—The revised protocol, with improvements to performance and manager-to-manager communications. Specifically, SNMPv2c implements community strings, which act as passwords when determining who, what, and how the SNMP clients can access the data in the SNMP agent. The community string is contained in SNMP Get, GetBulk, GetNext, and Set requests. The agent might require a different community string for Get, GetBulk, and GetNext requests (read-only access) than it does for Set requests (read-write access).
- SNMPv3—The most up-to-date protocol focuses on security. SNMPv3 defines a security model, user-based security model (USM), and a view-based access control model (VACM). SNMPv3 USM provides data integrity, data origin authentication, message replay protection, and protection against disclosure of the message payload. SNMPv3 VACM provides access control to determine whether a specific type of access (read or write) to the management information is allowed.

In addition, the Junos OS SNMP agent software accepts IPv4 and IPv6 addresses for transport over IPv4 and IPv6. For IPv6, the Junos OS supports the following features:

- SNMP data over IPv6 networks
- IPv6-specific MIB data
- SNMP agents for IPv6
**System Logging Severity Levels for SNMP Traps**

For some traps, when a trap condition occurs, regardless of whether the SNMP agent sends a trap to an NMS, the trap is logged if the system logging is configured to log an event with that system logging severity level.

For more information about system logging severity levels for standard traps, see *Standard SNMP Traps Supported by Junos OS*. For more information about system logging severity levels for enterprise-specific traps, see *Enterprise-Specific SNMP Traps Supported by Junos OS*.

**SNMP Communication Flow**

When a NMS polls the primary agent for data, the primary agent immediately shares the data with the NMS if the requested data is available from the primary agent or one of the subagents. However, if the requested data does not belong to those categories that are maintained by the primary 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 primary agent, which in turn passes it to the NMS.

*Figure 1 on page 6* shows the communication flow among the NMS, SNMP primary agent (snmpd), SNMP subagents, Junos OS kernel, and the Packet Forwarding Engine.

---

**Figure 1: SNMP Communication Flow**

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 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.

**Trap Queuing**

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.
NOTE: You cannot configure trap queueing in Junos OS. You cannot view information about trap queues except for what is provided in the system logs.

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. 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.

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 the 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.

NOTE: Users cannot configure the Junos OS for trap queueing. Users cannot view any information about trap queues except what is available in the logged information.
Configuration

Configuring SNMP on Junos Devices | 9
Monitoring SNMP Activity and Traps | 21
Optimizing the Network Management System | 33
Configuring SNMP on Junos Devices

### Best Practices for Configuring SNMP

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

#### Configuring Basic Settings for SNMPv1 and SNMPv2

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

**Enabling SNMPv1 and SNMPv2 Get and GetNext Operations**

```
[edit]
snmp {
```
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**
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**

```text
[edit snmp]
v3 {
  usm {
    local-engine {
      user jnpruser {
        authentication-md5 {
          authentication-key "$9$guADiQFNaOQzevMWx7ikqP'; ## 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;
    }
  }
}
```
[edit snmp]
v3 {
  usm {
    remote-engine 00000063200133a2c0a845c3 {
      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;
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**

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.

```
[edit]
  snmp {
    name "snmp001"; # 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 Junos OS**
- **Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running Junos OS**
- **Optimizing the Network Management System Configuration for the Best Results**
- **Configuring Options on Managed Devices for Better SNMP Response Time**
- **Managing Traps and Informs**

---

**Configuring Options on Managed Devices for Better SNMP Response Time**

---

**IN THIS SECTION**

- Enabling the stats-cache-lifetime Option | 15
- Filtering Out Duplicate SNMP Requests | 15
- Excluding Interfaces That Are Slow in Responding to SNMP Queries | 16

---

The following sections contain information about configuration options on the managed devices that can enhance SNMP performance:
Enabling the stats-cache-lifetime Option

Junos OS provides you with an option to configure the length of time (in seconds) that the interface stats is cached. If the NMS queries for the same interface again within the cache time, the same data is returned. If the NMS queries after the cache time, the cache is no more valid and fresh data is fetched from the lower layers and the cache timestamp is updated. The default value for stats-cache-lifetime option is 5 seconds. This can be tuned as per the polling frequency.

NOTE: Reducing the value of the stats-cache-lifetime option results in more queries and can impact performance. To get the live uncached statistics, set the value of the stats-cache-lifetime option to 0. However, this is not recommended since it completely disables the caching feature and impacts performance.

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 Junos OS. 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 Junos OS
- Best Practices for Configuring SNMP
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 | counter64 | integer | string | unsigned integer> object-value value
- request snmp utility-mib clear instance name object-type <counter | counter64 | 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 | counter64 | 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:

```xml
event-options {
    generate-event {
        1-HOUR time-interval 3600;
    }
    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/`:

```perl
------- 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 $split = jcs:regex($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
user@caramels>

NOTE: The `show snmp mib walk` command is not available on the QFabric system, but you can use external SNMP client applications to perform this operation.

RELATED DOCUMENTATION

- **Understanding SNMP Implementation in Junos OS**
- **Best Practices for Configuring SNMP**
- **Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running Junos OS**
- **Optimizing the Network Management System Configuration for the Best Results**
- **Configuring Options on Managed Devices for Better SNMP Response Time**
- **Managing Traps and Informs**
Understanding the Implementation of SNMP on the QFabric System
Monitoring SNMP Activity and Traps

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

Checking for MIB Objects Registered with the snmpd

Tracking SNMP Activity

Monitoring SNMP Statistics

Checking CPU Utilization

Checking Kernel and Packet Forwarding Engine Response

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 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

```

```
RSakMIBSRakMIB

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)
```
NOTE: The /var/log/snmp_reg_objs file contains only those objects that are associated with the 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, Junos OS does not trace any SNMP activity. To enable tracking of SNMP activities on a device running 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] >>> OID : jnxContentsType.7.1.0.0
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] <<< 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] <<< OID : jnxContentsType.7.2.0.0
Apr 12 06:40:03 snmpd[7ee783df] <<< type : Object
Apr 12 06:40:03 snmpd[7ee783df] <<< value: jnxM10iRE.0
```

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

<table>
<thead>
<tr>
<th>PID</th>
<th>USERNAME</th>
<th>THR</th>
<th>PRI</th>
<th>NICE</th>
<th>SIZE</th>
<th>RES</th>
<th>STATE</th>
<th>TIME</th>
<th>WCPU</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>root</td>
<td>1</td>
<td>171</td>
<td>52</td>
<td>0K</td>
<td>12K</td>
<td>RUN</td>
<td>132:09: 95.21% idle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1184</td>
<td>root</td>
<td>1</td>
<td>97</td>
<td>0</td>
<td>35580K</td>
<td>9324K</td>
<td>select</td>
<td>4:16</td>
<td>1.61%</td>
<td>chassisd</td>
</tr>
<tr>
<td>177</td>
<td>root</td>
<td>1</td>
<td>-8</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>mdwait</td>
<td>0:51</td>
<td>0.00%</td>
<td>md7</td>
</tr>
<tr>
<td>119</td>
<td>root</td>
<td>1</td>
<td>-8</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>mdwait</td>
<td>0:20</td>
<td>0.00%</td>
<td>md4</td>
</tr>
<tr>
<td>13</td>
<td>root</td>
<td>1</td>
<td>-20</td>
<td>-139</td>
<td>0K</td>
<td>12K</td>
<td>WAIT</td>
<td>0:16</td>
<td>0.00%</td>
<td>swi7: clock sio</td>
</tr>
<tr>
<td>1373</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>15008K</td>
<td>12712K</td>
<td>select</td>
<td>0:09</td>
<td>0.00%</td>
<td>snmpd</td>
</tr>
<tr>
<td>1371</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>9520K</td>
<td>5032K</td>
<td>select</td>
<td>0:08</td>
<td>0.00%</td>
<td>jdiameeterd</td>
</tr>
<tr>
<td>12</td>
<td>root</td>
<td>1</td>
<td>-40</td>
<td>-159</td>
<td>0K</td>
<td>12K</td>
<td>WAIT</td>
<td>0:07</td>
<td>0.00%</td>
<td>swi2: net</td>
</tr>
<tr>
<td>1375</td>
<td>root</td>
<td>2</td>
<td>96</td>
<td>0</td>
<td>15016K</td>
<td>5812K</td>
<td>select</td>
<td>0:06</td>
<td>0.00%</td>
<td>pfed</td>
</tr>
<tr>
<td>49</td>
<td>root</td>
<td>1</td>
<td>-8</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>mdwait</td>
<td>0:05</td>
<td>0.00%</td>
<td>md0</td>
</tr>
<tr>
<td>1345</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>10088K</td>
<td>4480K</td>
<td>select</td>
<td>0:05</td>
<td>0.00%</td>
<td>12ald</td>
</tr>
<tr>
<td>1181</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>1608K</td>
<td>908K</td>
<td>select</td>
<td>0:05</td>
<td>0.00%</td>
<td>bslockd</td>
</tr>
<tr>
<td>23</td>
<td>root</td>
<td>1</td>
<td>-68</td>
<td>-187</td>
<td>0K</td>
<td>12K</td>
<td>WAIT</td>
<td>0:04</td>
<td>0.00%</td>
<td>irq10: fxpl</td>
</tr>
<tr>
<td>30</td>
<td>root</td>
<td>1</td>
<td>171</td>
<td>52</td>
<td>0K</td>
<td>12K</td>
<td>pgzero</td>
<td>0:04</td>
<td>0.00%</td>
<td>pagezero</td>
</tr>
<tr>
<td>1344</td>
<td>root</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>39704K</td>
<td>11444K</td>
<td>kqread</td>
<td>0:03</td>
<td>0.00%</td>
<td>rpdp</td>
</tr>
<tr>
<td>1205</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>3152K</td>
<td>912K</td>
<td>select</td>
<td>0:03</td>
<td>0.00%</td>
<td>license-check</td>
</tr>
<tr>
<td>1372</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>28364K</td>
<td>6696K</td>
<td>select</td>
<td>0:03</td>
<td>0.00%</td>
<td>dcd</td>
</tr>
<tr>
<td>1374</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>11764K</td>
<td>7632K</td>
<td>select</td>
<td>0:02</td>
<td>0.00%</td>
<td>mib2d</td>
</tr>
<tr>
<td>1405</td>
<td>user</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>15892K</td>
<td>11132K</td>
<td>select</td>
<td>0:02</td>
<td>0.00%</td>
<td>cli</td>
</tr>
<tr>
<td>139</td>
<td>root</td>
<td>1</td>
<td>-8</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>mdwait</td>
<td>0:02</td>
<td>0.00%</td>
<td>md5</td>
</tr>
<tr>
<td>22</td>
<td>root</td>
<td>1</td>
<td>-80</td>
<td>-199</td>
<td>0K</td>
<td>12K</td>
<td>WAIT</td>
<td>0:02</td>
<td>0.00%</td>
<td>irq9: cbbl fxp0</td>
</tr>
<tr>
<td>1185</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>4472K</td>
<td>2036K</td>
<td>select</td>
<td>0:02</td>
<td>0.00%</td>
<td>alarmd</td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>1</td>
<td>-8</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>-</td>
<td>0:02</td>
<td>0.00%</td>
<td>g_down</td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>1</td>
<td>-8</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>-</td>
<td>0:02</td>
<td>0.00%</td>
<td>g_up</td>
</tr>
<tr>
<td>43</td>
<td>root</td>
<td>1</td>
<td>-16</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>psleep</td>
<td>0:02</td>
<td>0.00%</td>
<td>vmkmemdaemon</td>
</tr>
<tr>
<td>1377</td>
<td>root</td>
<td>1</td>
<td>96</td>
<td>0</td>
<td>3776K</td>
<td>2256K</td>
<td>select</td>
<td>0:01</td>
<td>0.00%</td>
<td>irsd</td>
</tr>
<tr>
<td>48</td>
<td>root</td>
<td>1</td>
<td>-16</td>
<td>0</td>
<td>0K</td>
<td>12K</td>
<td>-</td>
<td>0:01</td>
<td>0.00%</td>
<td>schedcpu</td>
</tr>
</tbody>
</table>
Checking Kernel and Packet Forwarding Engine Response

As mentioned in *Understanding SNMP Implementation in Junos OS*, 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. 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)

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)
```
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:

```plaintext
[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**
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:

```yaml
[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):

```yaml
[edit event-options policy p1]
events ui_mgd_terminate;
then {
  raise-trap;
}
```
[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 "$9$table1h7Nbw2axN"; ## SECRET-DATA
    security-name sn_v2c_trap;
    tag tg1;
}
view all {
    oid .1 include;
}

RELATED DOCUMENTATION

- *Best Practices for Configuring SNMP*
- *Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running Junos OS*
- *Optimizing the Network Management System Configuration for the Best Results*
- *Configuring Options on Managed Devices for Better SNMP Response Time*
- *Using the Enterprise-Specific Utility MIB to Enhance SNMP Coverage*
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.

snmp bulk-get recommended number of OIDs and max-repetitions

Generally an SNMP bulk-get request responds with a total of (max-repetitions * number-of-OIDs) variable bindings. When interface statistics objects (such as ifInOctets, ifOutOctets, etc) are present in a query, the requests are sent to lower layers. Hence these responses are impacted by an increase in the 'max-repetitions' that you send in a bulk-get request. For bulk-get queries, for interface stats objects it is recommended to use 'max-repetitions' value of 10 and maximum number of OIDs per request is 10.

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 Junos OS
Monitoring SNMP Activity and Tracking Problems That Affect SNMP Performance on a Device Running Junos OS
Managing Traps and Informs