

## Chapter 5

# Configure SNMP

To configure the Simple Network Management Protocol (SNMP), include the following statements at the [edit snmp] hierarchy level of the configuration.

```
snmp {
  community community-name {
    authorization authorization;
    clients {
      address restrict;
    }
    view view-name;
  }
  contact contact;
  description description;
  engine-id {
    (local engine-id | use-fxp0-mac-address | use-default-ip-address);
  }
  interface [ interface-names ];
  location location;
  name name;
  nonvolatile {
    commit-delay seconds;
  }
  rmon {
    alarm index {
      description text-description;
      falling-event-index index;
      falling-threshold integer;
      interval seconds;
      rising-event-index index;
      falling-threshold integer;
      sample-type type;
      startup-alarm alarm;
      variable oid-variable;
    }
    event index {
      community community-name;
      description text-description;
      sample-type type;
    }
  }
}
```

```

traceoptions {
  file size size files number;
  flag flag;
}
trap-group group-name {
  categories [ categories ];
  destination-port <port-number>;
  targets {
    address;
  }
  version (all | v1 | v2);
}
trap-options {
  agent-address outgoing-interface;
  source-address address;
}
view view-name; {
  oid object-identifier (include | exclude);
)
}

```

For information about configuring Remote Monitoring (RMON) alarms and events, see “Configure RMON Alarms and Events” on page 179 and “Summary of RMON Alarm and Event Configuration Statements” on page 199.

By default, SNMP is disabled.

This chapter describes the minimum required configuration and discusses the following tasks for configuring SNMP:

- Minimum SNMP Configuration on page 29
- Configure the System Contact on page 29
- Configure the System Location on page 29
- Configure the System Description on page 30
- Configure the Commit Delay Timer on page 31
- Configure the System Name on page 31
- Configure the SNMP Community String on page 32
- Configure SNMP Trap Options and Groups on page 34
- Configure the Interfaces on Which SNMP Requests Can Be Accepted on page 41
- Configure MIB Views on page 42
- Trace SNMP Activity on page 43
- Configure the Local Engine ID on page 44

## Minimum SNMP Configuration

---

To configure the minimum requirements for SNMP, include statements at the [edit snmp] hierarchy level of the configuration:

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

The community defined here as public grants read access to all MIB data to any client.

## Configure the System Contact

---

You can specify an administrative contact for each system being managed by SNMP. This name is placed into the MIB II sysContact object. To configure a contact name, include the contact statement at the [edit snmp] hierarchy level:

```
[edit snmp]
contact contact;
```

If the name contains spaces, enclose it in quotation marks (" ").

### **Example: Configure the System Contact**

Define the system contact:

```
[edit]
snmp {
  contact "Junipero Berry, (650) 555-1234";
}
```

## Configure the System Location

---

You can specify the location of each system being managed by SNMP. This string is placed into the MIB II sysLocation object. To configure a system location, include the location statement at the [edit snmp] hierarchy level:

```
[edit snmp]
location location;
```

If the location contains spaces, enclose it in quotation marks (" ").

### **Example: Configure the System Location**

Specify where the system is located:

```
[edit]
snmp {
  location "Row 11, Rack C";
}
```

## Configure the System Description

---

You can specify a description for each system being managed by SNMP. This string is placed into the MIB II sysDescription object. To configure a description, include the description statement at the [edit snmp] hierarchy level:

```
[edit snmp]
description description;
```

If the description contains spaces, enclose it in quotation marks (" ").

### ***Example: Configure the System Description***

Specify the system description:

```
[edit]
snmp {
  description "M40 router with 8 FPCs";
}
```

## Configure the Commit Delay Timer

---

When the router first receives an SNMP nonvolatile set request, a JUNOScript session opens and prevents other users or applications from changing the candidate configuration (equivalent to the command-line interface [CLI] `configure exclusive` command). If the router does not receive new SNMP set requests within 5 seconds (the default value), the candidate configuration is committed and the JUNOScript session closes (the configuration lock is released). If the router receives new SNMP set requests while the candidate configuration is being committed, the SNMP set request is rejected and an error is generated. If the router receives new SNMP set requests before 5 seconds have elapsed, the commit-delay timer (the length of time between when the last SNMP request is received and the commit is requested) resets to 5 seconds.

By default, the timer is set to 5 seconds. To configure the timer for the SNMP set reply and start of the commit, include the `commit-delay` statement at the `[edit snmp nonvolatile]` hierarchy level:

```
[edit snmp nonvolatile]
  commit-delay seconds;
```

*seconds* is the length of the time between when the SNMP request is received and the commit is requested for the candidate configuration. For more information about the `configure exclusive` command and locking the configuration, see the *JUNOS Internet Software System Basics Configuration Guide* and the *JUNOScript API Reference*.

## Configure the System Name

---

To specify the system name override, include the `name` statement at the `[edit snmp]` hierarchy level:

```
[edit snmp]
  name name;
```

If the name contains spaces, enclose it in quotation marks (" ").

### **Example: Configure the System Name**

Specify the system name override:

```
[edit]
snmp {
  name "snmp 1";
}
```

## Configure the SNMP Community String

---

The SNMP community string defines the relationship between an SNMP server system and the client systems. This string acts like a password to control the clients' access to the server. To configure a community string, include the community statement at the [edit snmp] hierarchy level:

```
[edit snmp]
community name {
  authorization authorization;
  clients {
    default restrict;
    address restrict;
  }
  view view-name;
}
```

If the community name contains spaces, enclose it in quotation marks (" ").

The default authorization level for a community is read-only. To allow Set requests within a community, you need to define that community as authorization read-write. For Set requests, you also need to include the specific MIB objects that are accessible with read-write privileges using the view statement. The default view includes all supported MIB objects that are accessible with read-only privileges; no MIB objects are accessible with read-write privileges. For more information on the view statement, see “view” on page 152.

The clients statement lists the IP addresses of the clients (community members) that are allowed to use this community. If no clients statement is present, all clients are allowed. For *address*, you must specify an IPv4 or IPv6 address, not a hostname. Include the default restrict option to deny access to all SNMP clients for which access is not explicitly granted. We recommend that you always include the default restrict option to limit SNMP client access to the local router.



**NOTE:** Community names must be unique. You cannot configure the same community name at the [edit snmp community] and [edit snmp v3 snmp-community *community-index*] hierarchy levels.

---

### Examples: Configure the SNMP Community String

Grant read-only access to all clients. With the following configuration, the system responds to SNMP Get, GetNext, and GetBulk requests that contain the community string public:

```
[edit]
snmp {
  community public {
    authorization read-only;
  }
}
```

Grant all clients read-write access to ping MIB and jnxPingMIB. With the following configuration, the system responds to SNMP Get, GetNext, GetBulk, and Set requests that contain the community string private and specify an OID contained in the ping MIB or jnxPingMIB hierarchy:

```
[edit]
snmp {
  view ping-mib-view {
    oid pingMIB include;
    oid jnxPingMIB include;
  }
  community private {
    authorization read-write;
    view ping-mib-view;
  }
}
```

The following configuration allows read-only access to clients with IP addresses in the range 1.2.3.4/24, and denies access to systems in the range fe80::1:2:3:4/64:

```
[edit]
snmp {
  community field-service {
    authorization read-only;
    clients {
      default restrict; # Restrict access to all SNMP clients not explicitly
                        # listed on the following lines.
      1.2.3.4/24;      # Allow access by all clients in 1.2.3.4/24; except
      fe80::1:2:3:4/64 restrict; # fe80::1:2:3:4/64
    }
  }
}
```

## Configure SNMP Trap Options and Groups

---

Some carriers have more than one trap receiver that forwards traps to a central NMS. This allows for more than one path for SNMP traps from a router to the central NMS through different trap receivers. A router can be configured to send the same copy of each SNMP trap to every trap receiver configured in the trap group.

The source address in the IP header of each SNMP trap packet is set to the address of the outgoing interface by default. When a trap receiver forwards the packet to the central NMS, the source address is preserved. The central NMS, looking only at the source address of each SNMP trap packet, assumes that each SNMP trap came from a different source.

In reality, the SNMP traps came from the same router, but each left the router through a different outgoing interface.

The statements discussed in the following sections are provided to allow the NMS to recognize the duplicate traps and to distinguish SNMPv1 traps based on the outgoing interface.

To configure SNMP trap options and trap groups, include the trap-options and trap-group statements at the [edit snmp] hierarchy level:

```
[edit snmp]
trap-options {
  agent-address outgoing-interface;
  source-address address;
}
trap-group group-name {
  categories [ categories ];
  destination-port <port-number>;
  targets {
    address;
  }
  version (all | v1 | v2);
}
```

This section includes the following topics:

Configure SNMP Trap Options on page 35

Configure SNMP Trap Groups on page 38

## Configure SNMP Trap Options

Using SNMP trap options, you can set the source address of every SNMP trap packet sent by the router to a single address regardless of the outgoing interface. In addition, you can set the agent address of the SNMPv1 traps. For more information on the contents of SNMPv1 traps, see RFC 1157.



**NOTE:** SNMP cannot be associated with any routing instances other than the master routing instance.

To configure SNMP trap options, include the trap-options statement at the [edit snmp] hierarchy level:

```
[edit snmp]
trap-options {
  agent-address outgoing-interface;
  source-address address;
}
```

You must also configure a trap group for the trap options to take effect. For information about trap groups, see “Configure SNMP Trap Groups” on page 38.

This section contains the following topics:

Configure the Source Address for SNMP Traps on page 35

Configure the Agent Address for SNMP Traps on page 37

### Configure the Source Address for SNMP Traps

You can configure the source address of trap packets in two ways: lo0 or a valid IPv4 address configured on one of the router interfaces. The value lo0 indicates that the source address of the SNMP trap packets will be set to the lowest loopback address configured on the interface lo0.

To specify a valid interface address as the source address for SNMP traps on one of the router interfaces, include the source-address statement at the [edit snmp trap-options] hierarchy level:

```
[edit snmp trap-options]
source-address address;
```

*address* is a valid IPv4 address configured on one of the router interfaces.

To specify the source address of the SNMP traps so that they will be sent to the lowest loopback address configured on the interface lo0, include the source-address statement at the [edit snmp trap-options] hierarchy level:

```
[edit snmp]
trap-options {
  source-address lo0;
}
```

To enable and configure the loopback address, include the address statement at the [edit interfaces lo0 unit 0 family inet] hierarchy level:

```
[edit interfaces]
lo0 {
  unit 0 {
    family inet {
      address ip-address;
    }
  }
}
```

**Example: Configure the Loopback Address as the Source Address of Trap Packets**

To configure the loopback address and source address trap option:

```
[edit snmp]
trap-options {
  source-address lo0;
}
trap-group "urgent-dispatcher" {
  version v2;
  categories link startup;
  targets {
    192.168.10.22;
    172.17.1.2;
  }
}

[edit interfaces]
lo0 {
  unit 0 {
    family inet {
      address 10.0.0.1/32;
      address 127.0.0.1/32;
    }
  }
}
```

In this example, the IP address 10.0.0.1 is the source address of every trap sent from this router.

## Configure the Agent Address for SNMP Traps

The agent address is only available in the SNMPv1 trap packets (see RFC 1157). By default, the router's default local address is used in the agent address field of the SNMPv1 trap. To configure the agent address, include the `agent-address` statement at the `[edit snmp trap-options]` hierarchy level. Currently, the agent address can only be the address of the outgoing interface:

```
[edit snmp]
trap-options {
    agent-address outgoing-interface;
}
```

### *Example: Configure the Outgoing Interface as the Agent Address*

Configure the outgoing interface as the agent address:

```
[edit snmp]
trap-options {
    agent-address outgoing-interface;
}
trap-group "urgent-dispatcher" {
    version v1;
    categories link startup;
    targets {
        192.168.10.22;
        172.17.1.2;
    }
}
```

In this example, each SNMPv1 trap packet sent has its agent address value set to the IP address of the outgoing interface.

## Configure SNMP Trap Groups

You can create and name a group of one or more types of SNMP traps and then define which systems receive the group of SNMP traps. The trap group must be configured for SNMP traps to be sent. To create an SNMP trap group, include the trap-group statement at the [edit snmp] hierarchy level:

```
[edit snmp]
trap-group group-name {
  categories [ categories ];
  destination-port <port-number>;
  targets {
    address;
  }
  version (all | v1 | v2);
}
```

The trap group name can be any string and is embedded in the community name field of the trap. To configure your own trap group port, include the destination-port statement. The default destination port is port 162.

Each trap group you define must have a name and one or more targets, which are the systems that receive the SNMP traps. Specify the targets by IPv4 or IPv6 address, not by hostname.

Specify the types of traps the trap group can receive in the categories statement. For information about which category traps belong to, see “Standard SNMP Traps” on page 119 and “Juniper Networks Enterprise-Specific SNMP Traps” on page 113.

A trap group can receive the following categories:

authentication—Authentication failures

chassis—Chassis/environment notifications

configuration—Configuration notifications

link—Link-related notifications (up-down transitions, DS-3 and DS-1 line status change, IPv6 interface state change, and Passive Monitoring PIC overload



**NOTE:** To send Passive Monitoring PIC overload interface traps, select the link trap category.

---

remote-operations—Remote operation notifications

rmon-alarm—Alarm for RMON events

routing—Routing protocol notifications

sonet-alarms—SONET/SDH alarms



**NOTE:** If you omit the SONET/SDH subcategories, all SONET/SDH trap alarm types are included in trap notifications.

If you include SONET/SDH subcategories, only those SONET/SDH trap alarm types are include in trap notifications.

---

loss-of-light—Loss of light alarm notification

pll-lock—PLL lock alarm notification

loss-of-frame—Loss of frame alarm notification

loss-of-signal—Loss of signal alarm notification

severely-errored-frame—Severely errored frame alarm notification

line-ais—Line AIS alarm notification

path-ais—Path AIS alarm notification

loss-of-pointer—Loss of pointer alarm notification

ber-defect—SONET/SDH bit error rate alarm defect notification

ber-fault—SONET/SDH error rate alarm fault notification

line-remote-defect-indication—Line remote defect indication alarm notification

path-remote-defect-indication—Path remote defect indication alarm notification

remote-error-indication—Remote error indication alarm notification

unequipped—Unequipped alarm notification

path-mismatch—Path mismatch alarm notification

loss-of-cell—Loss of cell delineation alarm notification

vt-ais—VT AIS alarm notification

vt-loss-of-pointer—VT loss of pointer alarm notification

vt-remote-defect-indication—VT remote defect indication alarm notification

vt-unequipped—VT unequipped alarm notification

vt-label-mismatch—VT label mismatch error notification

vt-loss-of-cell—VT loss of cell delineation notification

startup—System warm and cold starts

vrrp-events—VRRP events such as new-master or authentication failures

The version statement allows you to specify the SNMP version of the traps sent to targets of the trap group. If you specify v1 only, SNMPv1 traps are sent. If you specify v2 only, SNMPv2 traps are sent. If you specify all, both an SNMPv1 and an SNMPv2 trap are sent for every trap condition. For more information on the version statement, see version on page 151.

### Example: Configure SNMP Trap Groups

Set up a trap notification list named urgent-dispatcher for link and startup traps. This list is used to identify the network management hosts (1.2.3.4 and fe80::1:2:3:4) to which traps generated by the local router should be sent. The name specified for a trap group is used as the SNMP community string when the agent sends traps to the listed targets.

```
[edit]
snmp {
  trap-group "urgent-dispatcher" {
    version v2;
    categories link startup;
    targets {
      1.2.3.4;
      fe80::1:2:3:4;
    }
  }
}
```

## Configure the Interfaces on Which SNMP Requests Can Be Accepted

---

By default, all router interfaces have SNMP access privileges. To limit the access through certain interfaces only, include the interface statement at the [edit snmp] hierarchy level:

```
[edit snmp]
interface [ interface-names ];
```

Specify the names of any logical or physical interfaces that should have SNMP access privileges. Any SNMP requests entering the router from interfaces not listed are discarded.

### **Example: Configure Secured Access List Checking**

Grant SNMP access privileges only to devices on interfaces so-0/0/0 and at-1/0/1. The following example does this by configuring a list of logical interfaces:

```
[edit]
snmp {
  interface [ so-0/0/0.0 so-0/0/0.1 at-1/0/1.0 at-1/0/1.1 ];
}
```

The following example grants the same access by configuring a list of physical interfaces:

```
[edit]
snmp {
  interface [ so-0/0/0 at-1/0/1 ];
}
```

## Configure MIB Views

---

By default, an SNMP community grants read access and denies write access to all supported MIB objects (even communities configured as authorization read-write). To restrict or grant read or write access to a set of MIB objects, you must configure a MIB view and associate the view with a community.

To configure MIB views, include the view statement at the [edit snmp] hierarchy level:

```
[edit snmp]
view view-name {
    oid object-identifier (include | exclude);
}
```

The view statement defines a MIB view and identifies a group of MIB objects. Each MIB object of a view has a common OID prefix. Each object identifier represents a subtree of the MIB object hierarchy. The subtree can be represented either by a sequence of dotted integers (such as 1.3.6.1.2.1.2) or by its subtree name (such as interfaces). A configuration statement uses a view to specify a group of MIB objects on which to define access. To enable a view, you must associate the view with a community.



**NOTE:** To remove an OID completely, use the delete view all oid *oid-number* command but omit the include parameter.

---

To associate MIB views with a community, include the view statement at the [edit snmp community-name] hierarchy level:

```
[edit snmp community community-name]
view view-name;
```

### Example: Ping Proxy MIB

Restrict the ping-mib community to read and write access of the ping MIB and jnxpingMIB only. Read or write access to any other MIB using this community is not allowed.

```
[edit snmp]
view ping-mib-view {
    oid 1.3.6.1.2.1.80 include;      #pingMIB
    oid jnxPingMIB include;        #jnxPingMIB
}
community ping-mib {
    authorization read-write;
    view ping-mib-view;
}
```

For more information on the ping MIB, see RFC 2925 and “Juniper Networks Enterprise-Specific MIBs” on page 109.

## Trace SNMP Activity

---

To trace SNMP activity, include the `traceoptions` statement at the `[edit snmp]` hierarchy level:

```
[edit snmp]
traceoptions {
    file size size files number;
    flag flag;
}
```

The output of the tracing operations is placed into log files in the `/var/log` directory. Each log file is named after the SNMP agent that generates it. Currently, the following logs are created in the `/var/log` directory when the `traceoptions` statement is used:

```
chassisd
craftd
ilmid
mib2d
rmopd
serviced
snmpd
```

You can use the `file` statement to control log file generation. The `size` statement limits the size (in kilobytes) of each log file before it is closed and compressed and a new file opened in its place. The `file` statement limits the total number of log files archived for each SNMP agent.

You can specify one or more of the following values for the `flag` option:

```
all—Trace all SNMP events
general—Trace general events
interface-stats—Trace physical and logical interface statistics
nonvolatile—Trace nonvolatile SNMP set request handling
pdu—Trace SNMP request and response packets
protocol-timeouts—Trace SNMP response timeouts
routing-socket—Trace routing socket calls
subagent—Trace subagent restarts
timer—Trace internal timer events
varbind-error—Trace variable binding errors
```

### Example: Trace SNMP Activity

Trace information on SNMP packets:

```
[edit]
snmp {
  traceoptions {
    file size 10k files 5;
    flag pdu;
    flag protocol-timeouts;
    flag varbind-error;
  }
}
```

### Configure the Local Engine ID

---

By default, the local engine ID uses the default IP address of the router. The local engine ID is the administratively unique identifier for the SNMPv3 engine. This statement is optional. To configure the local engine ID, include the `engine-id` statement at the `[edit snmp]` hierarchy level:

```
[edit snmp]
message-processing-model {
  (local engine-id-suffix | use-fxp0-mac-address | use-default-ip-address);
}
```

`local engine-id-suffix`—The engine ID suffix is explicitly configured.

`use-fxp0-mac-address`—The engine ID suffix is generated from the MAC address of `fxp0`.

`use-default-ip-address`—The engine ID suffix is generated from the default IP address.

The local engine ID is defined as the administratively unique identifier of an SNMPv3 engine, and is used for identification, not for addressing. There are two parts of an engine ID: prefix and suffix. The prefix is formatted according to the specifications defined in RFC 3411, *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*. You can configure the suffix here. For information about how to configure SNMPv3, see “SNMPv3 Overview” on page 45 and “Configure SNMPv3” on page 47 .



**NOTE:** SNMPv3 authentication and encryption keys are generated based on the associated passwords and the engine ID. If you configure or change the engine ID, you must commit the new engine ID before you configure SNMPv3 users. Otherwise the keys generated from the configured passwords will be based on the previous engine ID.

For the engine ID, we recommend using the MAC address of `fxp0`.

---