JUNIPER NETWORKS

Technology Overview

Frequently Asked Questions: SNMP on Junos OS

Release
12.3

Published: 2012-11-14

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

Technology Overview Frequently Asked Questions: SNMP on Junos OS
Release 12.3

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. 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

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

Introduction

This document presents the most frequently asked questions about the features and technologies used to implement SNMP services on Juniper Networks devices using the Juniper Networks® Junos® operating system (Junos OS).

Junos OS SNMP FAQs Overview

SNMP enables users to monitor network devices from a central location. Many network management systems (NMS) are based on SNMP, and support for this protocol is a key feature of most network devices.

Juniper Networks provides many different platforms that support SNMP on Junos OS. Junos OS includes an onboard SNMP agent that provides remote management applications with access to detailed information about the devices on the network.

A typical SNMP implementation contains three components:

- Managed devices – Such as routers and switches.
- SNMP agent – Process that resides on a managed device and communicates with the NMS.
- NMS – A combination of hardware and software used to monitor and administer the network; network device that runs SNMP manager software. Also referred to as an SNMP manager.

The SNMP agent exchanges network management information with the SNMP manager (NMS). The agent responds to requests for information and actions from the manager. The SNMP manager collects information about network connectivity, activity, and events by polling managed devices.

SNMP implementation in Junos OS uses a master SNMP agent (known as an SNMP process or snmpd) that resides on the managed device. Various subagents reside on different modules of Junos OS as well (such as the Routing Engine), and these subagents are managed by the snmpd.

Related Documentation

- Junos OS SNMP FAQs on page 1

Junos OS SNMP FAQs

This Frequently Asked Questions technology overview covers these SNMP-related areas:

Junos OS SNMP Support FAQs

This section presents frequently asked questions and answers related to SNMP support on Junos OS.

Which SNMP versions does Junos OS support?

Junos OS supports SNMP version 1 (SNMPv1), version 2 (SNMPv2c), and version 3 (SNMPv3). By default, SNMP is disabled on a Juniper Networks device. To enable SNMP, see the instructions in Configuring SNMP on Devices Running Junos OS in the *Best Practices SNMP-Based Network Management on Devices Running Junos OS* document.

Which ports (sockets) does SNMP use?

The default port for SNMP queries is port 161. The default port for SNMP traps and informs is port 162. The ports used by SNMP are configurable, and you can configure your system to use ports other than the defaults.

Is SNMP support different among the Junos OS platforms?

No, SNMP support is not different among the Junos OS platforms. SNMP configuration, interaction, and behavior are the same on any Junos OS device. The only difference that might occur across platforms is MIB support. See "Junos OS MIBs FAQs" on page 3 for more information about MIB support on the Junos OS platforms.

See also the *Junos OS Network Management Configuration Guide* for a list of MIBs that are supported across the Junos OS platforms.

Does Junos OS support the user-based security model (USM)?

Yes, Junos OS supports USM as part of its support for SNMPv3. SNMPv3 contains more security measures than previous versions of SNMP, including providing a defined USM. SNMPv3 USM provides message security through data integrity, data origin authentication, message replay protection, and protection against disclosure of the message payload.

Does Junos OS support the view-based access control model (VACM)?

Yes, Junos OS supports VACM as part of its support for SNMPv3. SNMPv3 contains more security measures than previous versions of SNMP, including providing a defined VACM. SNMPv3 VACM determines whether a specific type of access (read or write) to the management information is allowed.

Does Junos OS support SNMP informs?

Yes, Junos OS supports SNMP informs as part of its support for SNMPv3. SNMP informs are confirmed notifications sent from SNMP agents to SNMP managers when significant

events occur on a network device. When an SNMP manager receives an inform, it sends a response to the sender to verify receipt of the inform.

Can I provision or configure a device using SNMP on Junos OS?

No, provisioning or configuring a device using SNMP is not allowed on Junos OS.

Related Documentation

- Junos OS MIBs FAQs on page 3
- Junos OS SNMP Configuration FAQs on page 11
- SNMPv3 FAQs on page 15
- SNMP Interaction with Juniper Networks Devices FAQs on page 17
- SNMP Traps and Informs FAQs on page 19
- SNMP Support for Routing Instances FAQs on page 25
- SNMP Counters FAQs on page 27

Junos OS MIBs FAQs

This section presents frequently asked questions and answers related to Junos OS MIBs.

What is a MIB?

A management information base (MIB) is a table of definitions for managed objects in a network device. MIBs are used by SNMP to maintain standard definitions of all of the components and their operating conditions within a network device. Each object in the MIB has an identifying code called an object identifier (OID).

MIBs are either standard or enterprise-specific. Standard MIBs are created by the Internet Engineering Task Force (IETF) and documented in various RFCs. Enterprise-specific MIBs are developed and supported by a specific equipment manufacturer.

For a list of supported standard MIBs, see Standard SNMP MIBs Supported by Junos OS in the *Junos OS SNMP MIBs and Traps Reference* document.

For a list of Juniper Networks enterprise-specific MIBs, see Juniper Networks Enterprise-Specific MIBs in the *Junos OS SNMP MIBs and Traps Reference* document.

Do MIB files reside on the Junos OS devices?

No, MIB files do not reside on the Junos OS devices. You must download the MIB files from the Juniper Networks Technical Publications page for the required Junos OS release: http://www.juniper.net/techpubs/en_US/release-independent/junos/mibs/mibs.html .

How do I compile and load the Junos OS MIBs onto an SNMP manager or NMS?

For your network management systems (NMSs) to identify and understand the MIB objects used by Junos OS, you must first load the MIB files to your NMS using a MIB compiler. A MIB compiler is a utility that parses the MIB information, such as the MIB object names, IDs, and data types for the NMS.

You can download the Junos OS MIB package from the Enterprise-Specific MIBs and Traps section at http://www.juniper.net/techpubs/en_US/release-independent/junos/mibs/mibs.html or http://www.juniper.net/techpubs/software/junos/index.html .

The Junos OS MIB package has two folders: **StandardMibs**, containing standard MIBs supported on Juniper Networks devices, and **JuniperMibs**, containing Juniper Networks enterprise-specific MIBs. You *must* have the required standard MIBs downloaded and decompressed before downloading any enterprise-specific MIBs. There might be dependencies that require a particular standard MIB to be present on the compiler before loading a particular enterprise-specific MIB.

The Junos OS MIB package is available in **.zip** and **.tar** formats. Download the format appropriate for your requirements.

Use the following steps to load MIB files for devices running Junos OS:

1. Navigate to the appropriate Juniper Networks software download page and locate the **Enterprise MIBs** link under the **Enterprise-Specific MIBs and Traps** section.

   

   NOTE: **Although the link is titled Enterprise MIBs, both standard MIBs and enterprise-specific MIBs are available for download from this location.**

2. Click the **TAR** or **ZIP** link to download the Junos OS MIB package.
3. Decompress the file (**.tar** or **.zip**) using an appropriate utility.

   

   NOTE: **Some commonly used MIB compilers are preloaded with standard MIBs. You can skip Step 4 and Step 5 and proceed to Step 6 if you already have the standard MIBs loaded on your system.**

4. Load the standard MIB files from the **StandardMibs** folder.

   Load the files in the following order:

   a. mib-SNMPv2-SMI.txt
   b. mib-SNMPv2-TC.txt
   c. mib-IANAifType-MIB.txt
   d. mib-IANA-RTPROTO-MIB.txt
   e. mib-rfc1907.txt
   f. mib-rfc2011a.txt
   g. mib-rfc2012a.txt
   h. mib-rfc2013a.txt
   i. mib-rfc2863a.txt

5. Load any remaining standard MIB files.

NOTE: You must follow the order specified in this procedure, and ensure that all standard MIBs are loaded before you load the enterprise-specific MIBs. There might be dependencies that require a particular standard MIB to be present on the compiler before loading a particular enterprise-specific MIB. Dependencies are listed in the IMPORT section of the MIB file.

6. After loading the standard MIBs, load the Juniper Networks enterprise-specific SMI MIB, **mib-jnx-smi.txt**, and the following optional SMI MIBs based on your requirements:
   - mib-jnx-exp.txt—(Recommended) for Juniper Networks experimental MIB objects
   - mib-jnx-js-smi.txt—(Optional) for Juniper Security MIB tree objects
   - mib-jnx-ex-smi.txt—(Optional) for EX Series Ethernet Switches
7. Load any remaining desired enterprise-specific MIBs from the **JuniperMibs** folder.

TIP: While loading a MIB file, if the compiler returns an error message indicating that any of the objects are undefined, open the MIB file using a text editor and ensure that all the MIB files listed in the IMPORT section are loaded on the compiler. If any of the MIB files listed in the IMPORT section are not loaded on the compiler, load the missing file or files first, then try to load the MIB file that failed.

The system might return an error if files are not loaded in a particular order.

What is SMI?

Structure of Management Information Version (SMI) is a subset of Abstract Syntax Notation One (ASN.1), which describes the structure of objects. SMI is the notation syntax, or “grammar”, that is the standard for writing MIBs.

Which versions of SMI does Junos OS support?

The Junos OS supports SMIv1 for SNMPv1 MIBs, and SMIv2 for SNMPv2c and enterprise MIBs.

Does Junos OS support MIB II?

Yes, Junos OS supports MIB II, the second version of the MIB standard.

The features of MIB II include:

- Additions that reflect new operational requirements.
- Backward compatibility with the original MIBs and SNMP.

- Improved support for multiprotocol entities.
- Improved readability.

Refer to the relevant release documentation for a list of MIBs that are supported. Go to http://www.juniper.net/techpubs/software/junos/index.html .

Are the same MIBs supported across all Juniper Networks devices?

There are some common MIBs supported by all the Junos OS devices, such as the Interface MIB (ifTable), System MIB, and Chassis MIB. Some MIBs are supported only by functionalities on specific platforms. For example, the Bridge MIB is supported on the EX Series Ethernet Switches and the SRX Series Services Gateways for the branch.

What is the system object identifier (SYSOID) of a device? How do I determine the SYSOID of my device?

The jnx-chas-defines (Chassis Definitions for Router Model) MIB has a **jnxProductName** branch for every Junos OS device. The system object ID of a device is identical to the object ID of the **jnxProductName** for the platform. For example, for an M7i Multiservice Edge Router, the jnxProductNameM7i is .1.3.6.1.4.1.2636.1.1.1.2.10 in the jnxProductName branch, which is identical to the SYSOID of the M7i (.1.3.6.1.4.1.2636.1.1.1.2.10).

How can I determine if a MIB is supported on a platform? How can I determine which MIBs are supported by a device?

MIBs device and platform support is listed on the Junos OS Technical Documentation index page. Go to http://www.juniper.net/techpubs/software/junos/ and select your version or release of Junos OS. Navigate to the *SNMP MIBs and Traps Reference*. The *SNMP MIBs and Traps Reference* specifies which MIBs are supported on the different platforms.

What can I do if the MIB OID query is not responding?

There can be various reasons why the MIB OID query stops responding. One reason could be that the MIB itself is unresponsive. To verify that the MIB responds, use the **show snmp mib walk | get *MIB name* | *MIB OID*** command:

- If the MIB responds, the communication issue exists between the SNMP master and SNMP agent. Possible reasons for this issue include network issues, an incorrect community configuration, an incorrect SNMP configuration, and so on.
- If the MIB does not respond, enable SNMP **traceoptions** to log PDUs and errors. All incoming and outgoing SNMP PDUs are logged. Check the **traceoptions** output to see if there are any errors.

If you continue to have problems with the MIB OID query, technical product support is available through the Juniper Networks Technical Assistance Center (JTAC).

What is the enterprise branch number for Junos OS?

The enterprise branch number for Junos OS is 2636. Enterprise branch numbers are used in SNMP MIB configurations, and they are also known as SMI network management private enterprise codes.

Which MIB displays the hardware and chassis details on a Juniper Networks device?

The Chassis MIB (jnxchassis.mib) displays the hardware and chassis details for each Juniper Networks device. It provides information about the router and its components. The Chassis MIB objects represent each component and its status.

For more information about enterprise-specific Chassis MIBs, see Chassis MIBs in the *Junos OS SNMP MIBs and Traps Reference* document.

Does Junos OS support the Entity MIB?

No, Junos OS does not support the Entity MIB, which is designed to identify physical and logical elements of a managed device. Instead, Junos OS supports the enterprise-specific Chassis MIB to identify the chassis components on the device.

Which MIB objects can I query to determine the CPU and memory utilization of the Routing Engine, Flexible PIC Concentrator (FPC), and PIC components on a device?

Query the Chassis MIB objects **jnxOperatingMemory**, **jnxOperatingtBuffer**, and **jnxOperatingCPU** to find out the CPU and memory utilization of the hardware components of a device.

Is the interface index (ifIndex) persistent?

For the Junos OS Release 10.0 and earlier, the ifIndex is persistent when reboots occur if the Junos OS version remains the same, meaning the values assigned to the interfaces in the ifIndex do not change. When there is a software upgrade, the device tries to keep the ifIndex persistent on a best effort basis.

For the Junos OS Release 10.1 and later, the ifIndex is persistent on all platforms, except for the EX4200 virtual chassis because it can have over 500 interfaces.

Is it possible to set the ifAdminStatus?

SNMP is not allowed to set the ifAdminStatus.

Which MIB objects support SNMP set operations?

The Junos OS SNMP set operations are supported in the following MIB tables and variables:

- snmpCommunityTable
- eventTable
- alarmTable
- snmpTargetAddrExtTable
- jnxPingCtlTable
- pingCtlTable

- traceRouteCtlTable
- jnxTraceRouteCtlTable
- sysContact.0
- sysName.0
- sysLocation.0
- pingMaxConcurrentRequests.0
- traceRouteMaxConcurrentRequests.0
- usmUserSpinLock
- usmUserOwnAuthKeyChange
- usmUserPublic
- vacmSecurityToGroupTable (vacmGroupName, vacmSecurityToGroupStorageType, and vacmSecurityToGroupStatus)
- vacmAccessTable (vacmAccessContextMatch, vacmAccessReadViewName, vacmAccessWriteViewName, vacmAccessNotifyViewName, vacmAccessStorageType, and vacmAccessStatus)
- vacmViewSpinLock
- vacmViewTreeFamilyTable (vacmViewTreeFamilyMask, vacmViewTreeFamilyType, vacmViewTreeFamilyStorageType, and vacmViewTreeFamilyStatus)

Does Junos OS support remote monitoring (RMON)?

Yes, Junos OS supports RMON as defined in RFC 2819, *Remote Network Monitoring Management Information Base*. However, remote monitoring version 2 (RMON 2) is not supported.

Can I use SNMP to determine the health of the processes running on the Routing Engine?

Yes, you can use SNMP to determine the health of the Routing Engine processes by configuring the health monitoring feature. On Juniper Networks devices, RMON alarms and events provide much of the infrastructure needed to reduce the polling overhead from the NMS. However, you must set up the NMS to configure specific MIB objects into RMON alarms. This often requires device-specific expertise and customizing the monitoring application. Additionally, some MIB object instances that need monitoring are set only at initialization, or they change at runtime and cannot be configured in advance.

To address these issues, the health monitor extends the RMON alarm infrastructure to provide predefined monitoring for a selected set of object instances, such as file system usage, CPU usage, and memory usage, and includes support for unknown or dynamic object instances, such as Junos OS software processes.

To display the health monitoring configuration, use the **show snmp health-monitor** command:

```
user@host> show snmp health-monitor
interval 300;
rising-threshold 90;
falling-threshold 80;
```

When you configure the health monitor, monitoring information for certain object instances is available, as shown in Table 1 on page 9.

Table 1: Monitored Object Instances

Object	Description
jnxHrStoragePercentUsed.1	Monitors the following file system on the router or switch: /dev/ad0s1a: This is the root file system mounted on **/**.
jnxHrStoragePercentUsed.2	Monitors the following file system on the router or switch: /dev/ad0s1e: This is the configuration file system mounted on **/config**.
jnxOperatingCPU (RE0) jnxOperatingCPU (RE1)	Monitor CPU usage for Routing Engines RE0 and RE1. The index values assigned to the Routing Engines depend on whether the Chassis MIB uses a zero-based or a ones-based indexing scheme. Because the indexing scheme is configurable, the correct index is determined whenever the router is initialized and when there is a configuration change. If the router or switch has only one Routing Engine, the alarm entry monitoring RE1 is removed after five failed attempts to obtain the CPU value.
jnxOperatingBuffer (RE0) jnxOperatingBuffer (RE1)	Monitor the amount of memory available on Routing Engines RE0 and RE1. Because the indexing of this object is identical to that used for jnxOperatingCPU, index values are adjusted depending on the indexing scheme used in the Chassis MIB. As with jnxOperatingCPU, the alarm entry monitoring RE1 is removed if the router or switch has only one Routing Engine.
sysApplElmtRunCPU	Monitors the CPU usage for each Junos OS software process. Multiple instances of the same process are monitored and indexed separately.
sysApplElmtRunMemory	Monitors the memory usage for each Junos OS software process. Multiple instances of the same process are monitored and indexed separately.

The system log entries generated for any health monitor events, such as thresholds crossed and errors, have a corresponding **HEALTHMONITOR** tag rather than a generic **SNMPD_RMON_EVENTLOG** tag. However, the health monitor sends generic RMON **risingThreshold** and **fallingThreshold** traps.

Are the Ping MIBs returned in decimal notation and ASCII?

Yes, both decimal notation and ASCII are supported, which is the standard implementation in SNMP. All strings are ASCII encoded.

The following example displays the Ping MIB in hexadecimal notation:

```
pingCtlTargetAddress.2.69.72.9.116.99.112.115.97.109.112.108.101 = 0a fa 01 02
```

This translates to ASCII:

```
pingCtlTargetAddress."EH"."tcpsample" = 0a fa 01 02
2= length of the string
69=E
72=H
9=length of second string
116=t
99 =c
112=p
115=s
97=a
109=m
112 =p
108 =l
101 =e
```

As of Junos OS Release 9.6 and later, the Junos OS CLI returns ASCII values using the command **show snmp mib *get* | *get-next* | *walk* ascii**.

The following example shows the output with the ASCII option:

```
user@host> show snmp mib walk pingCtlTargetAddress ascii
pingCtlTargetAddress."EH"."httpgetsample" = http://www.yahoo.com
pingCtlTargetAddress."p1"."t2" = 74 c5 b3 06
pingCtlTargetAddress."p1"."t3" = 74 c5 b2 0c
```

The following example shows the output without the ASCII option:

```
user@host> show snmp mib walk pingCtlTargetAddress
pingCtlTargetAddress.2.69.72.13.104.116.116.112.103.101.116.115.97.109.112.108.101
 = http://www.yahoo.com
pingCtlTargetAddress.2.112.49.2.116.50 = 74 c5 b3 06
pingCtlTargetAddress.2.112.49.2.116.51 = 74 c5 b2 0c
```

You can convert decimal and ASCII values using a decimal ASCII chart like the one at http://www.asciichart.com .

Is IPv6 supported by the Ping MIB for remote operations?

No, IPv6 is not supported.

Is there an SNMP MIB to show Address Resolution Protocol (ARP) table information? Are both IP and MAC addresses displayed in the same table?

Yes, the Junos OS supports the standard MIB **ipNetToMediaTable**, which is described in RFC 2011, *SNMPv2 Management Information Base for the Internet Protocol using SMIv2*. This table is used for mapping IP addresses to their corresponding MAC addresses.

Related Documentation

- Junos OS SNMP Support FAQs on page 2
- Junos OS SNMP Configuration FAQs on page 11
- SNMPv3 FAQs on page 15
- SNMP Interaction with Juniper Networks Devices FAQs on page 17
- SNMP Traps and Informs FAQs on page 19

Junos OS SNMP Configuration FAQs

This section presents frequently asked questions and answers related to Junos OS SNMP configuration.

Can the Junos OS be configured for SNMPv1 and SNMPv3 simultaneously?

Yes, SNMP has backward compatibility, meaning that all three versions can be enabled simultaneously.

Can I filter specific SNMP queries on a device?

Yes, you can filter specific SNMP queries on a device using **exclude** and **include** statements.

The following example shows a configuration that blocks read-write operation on all OIDs under .1.3.6.1.2.1.1 for the community **test**:

```
user@host# show snmp
view system-exclude {
   oid .1.3.6.1.2.1.1 exclude;
   oid .1 include;
}
   community test {
      view system-exclude;
      authorization read-write;
   }
```

Can I change the SNMP agent engine ID?

Yes, the SNMP agent engine ID can be changed to the MAC address of the device, the IP address of the device, or any other desired value. Several examples are included here.

The following example shows how to use the MAC address of a device as the SNMP agent engine ID:

```
user@host# show snmp
engine-id {
   use-mac-address;
}
```

The following example shows how to use the IP address of a device as the SNMP agent engine ID:

```
user@host# show snmp
engine-id {
   use-default-ip-address;
}
```

The following example shows the use of a selected value, **AA** in this case, as the SNMP agent engine ID of a device:

```
user@host# show snmp
engine-id {
```

```
    local AA;
}
```

How can I configure a device with dual Routing Engines or a chassis cluster (for SRX Series Services Gateways or J Series Service Routers) for continued communication during a switchover?

When configuring for continued communication, the SNMP configuration should be identical between the Routing Engines. However, it is best to have separate Routing Engine IDs configured for each Routing Engine, especially when using SNMPv3.

The following example shows the configuration of the Routing Engines in a dual Routing Engine device. Notice that the Routing Engine IDs are set to the MAC addresses for each Routing Engine:

```
user@host# show groups
re0 {
    system {
        host-name PE3-re0;
    }
    interfaces {
        fxp0 {
            unit 0 {
                family inet {
                    address 116.197.178.14/27;
                    address 116.197.178.29/27 {
                        master-only;
                    }
                }
            }
        }
    }
    snmp {
        engine-id {
            use-mac-address;
        }
    }
}
re1 {
    system {
        host-name PE3-re1;
    }
    interfaces {
        fxp0 {
            unit 0 {
                family inet {
                    address 116.197.178.11/27;
                    address 116.197.178.29/27 {
                        master-only;
                    }
                }
            }
        }
    }
    snmp {
        engine-id {
```

```
        use-mac-address;
      }
    }
  }
```

The following is an example of an SNMPv3 configuration on a dual Routing Engine device:

```
user@host> show snmp name host1
v3 {
  vacm {
    security-to-group {
      security-model usm {
        security-name test123 {
          group test1;
        }
        security-name juniper {
          group test1;
        }
      }
    }
    access {
      group test1 {
        default-context-prefix {
          security-model any {
            security-level authentication {
              read-view all;
            }
          }
        }
        context-prefix MGMT_10 {
          security-model any {
            security-level authentication {
              read-view all;
            }
          }
        }
      }
    }
  }
  target-address server1 {
    address 116.197.178.20;
    tag-list router1;
    routing-instance MGMT_10;
    target-parameters test;
  }
  target-parameters test {
    parameters {
      message-processing-model v3;
      security-model usm;
      security-level authentication;
      security-name juniper;
    }
    notify-filter filter1;
  }
  notify server {
    type trap;
```

```
        tag router1;
    }
    notify-filter filter1 {
        oid .1 include;
    }
    view all {
        oid .1 include;
    }
    community public {
        view all;
    }
    community comm1;
    community comm2;
    community comm3 {
        view all;
        authorization read-only;
        logical-system LDP-VPLS {
            routing-instance vpls-server1;
        }
    }
    trap-group server1 {
        targets {
            116.197.179.22;
        }
    }
    routing-instance-access;
    traceoptions {
        flag all;
    }
}
```

How can I track SNMP activities?

SNMP trace operations track activity of SNMP agents and record the information in log files.

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

```
[edit snmp]
user@host# 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

Related Documentation

- Junos OS SNMP Support FAQs on page 2
- Junos OS MIBs FAQs on page 3
- SNMPv3 FAQs on page 15
- SNMP Interaction with Juniper Networks Devices FAQs on page 17

SNMPv3 FAQs

This section presents frequently asked questions and answers related to SNMPv3.

Why is SNMPv3 important?

SNMP v3 provides enhanced security compared to the other versions of SNMP. It provides authentication and encryption of data. Enhanced security is important for managing devices at remote sites from the management stations.

In my system, the MIB object snmpEngineBoots is not in sync between two Routing Engines in a dual Routing Engine device. Is this normal behavior?

Yes, this is the expected behavior. Each Routing Engine runs its own SNMP process (snmpd), allowing each Routing Engine to maintain its own engine boots. However, if both routing engines have the same engine ID and the routing engine with lesser **snmpEngineBoots** value is selected as the master routing engine during the switchover process, the **snmpEngineBoots** value of the master routing engine is synchronized with the **snmpEngineBoots** value of the other routing engine.

Do I need the SNMP manager engine object identifier (OID) for informs?

Yes, the engine OID of the SNMP manager is required for authentication, and informs do not work without it.

I see the configuration of informs under the [edit snmp v3] hierarchy. Does this mean I cannot use informs with SNMPv2c?

Informs can be used with SNMPv2c. The following example shows the basic configuration for SNMPv3 informs on a device (note that the authentication and privacy is set to none):

```
[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:

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

Related Documentation

- Junos OS SNMP Support FAQs on page 2
- Junos OS MIBs FAQs on page 3
- Junos OS SNMP Configuration FAQs on page 11
- SNMP Interaction with Juniper Networks Devices FAQs on page 17
- SNMP Traps and Informs FAQs on page 19

SNMP Interaction with Juniper Networks Devices FAQs

This section presents frequently asked questions and answers related to how SNMP interacts with Juniper Networks devices.

How frequently should a device be polled? What is a good polling rate?

It is difficult to give an absolute number for the rate of SNMP polls per second since the rate depends on the following two factors:

- The number of variable bindings in a protocol data unit (PDU)
- The response time for an interface from the Packet Forwarding Engine

In a normal scenario where no delay is being introduced by the Packet Forwarding Engine and there is one variable per PDU (a Get request), the response time is 130+ responses per second. However, with multiple variables in an SNMP request PDU (30 to 40 for GetBulk requests), the number of responses per second is much less. Because the Packet Forwarding Engine load can vary for each system, there is greater variation in how frequently a device should be polled.

Frequent polling of a large number of counters, especially statistics, can impact the device. We recommend the following optimization on the SNMP managers:

- Use the row-by-row polling method, not the column-by-column method.
- Reduce the number of variable bindings per PDU.
- Increase timeout values in polling and discovery intervals.
- Reduce the incoming packet rate at the SNMP process (snmpd).

For better SNMP response on the device, the Junos OS does the following:

- Filters out duplicate SNMP requests.
- Excludes interfaces that are slow in response from SNMP queries.

One way to determine a rate limit is to note an increase in the **Currently Active** count from the **show snmp statistics extensive** command.

The following is a sample output of 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
```

Does SNMP open dynamic UDP ports? Why?

The SNMP process opens two additional ports (sockets): one for IPv4 and one for IPv6. This enables the SNMP process to send traps.

I am unable to perform a MIB walk on the ifIndex. Why is this?

Any variable bindings or values with an access level of **not-accessible** cannot be queried directly because they are part of other variable bindings in the SNMP MIB table. The ifIndex has an access level of **not-accessible**. Therefore, it cannot be accessed directly because it is part of the variable bindings. However, the ifIndex can be accessed indirectly through the variable bindings.

I see SNMP_IPC_READ_ERROR messages when the SNMP process restarts on my system and also during Routing Engine switchover. Is this acceptable?

Yes, it is acceptable to see **SNMP_IPC_READ_ERROR** messages when the SNMP process is restarted, the system reboots, or during a Routing Engine switchover. If all the processes come up successfully and the SNMP operations are working properly, then these messages can be ignored.

What is the source IP address used in the response PDUs for SNMP requests? Can this be configured?

The source IP address used in the response PDUs for SNMP requests is the IP address of the outgoing interface to reach the destination. The source IP address cannot be configured for responses. It can only be configured for traps.

Related Documentation

- Junos OS SNMP Support FAQs on page 2
- Junos OS MIBs FAQs on page 3
- Junos OS SNMP Configuration FAQs on page 11
- SNMPv3 FAQs on page 15
- SNMP Traps and Informs FAQs on page 19
- SNMP Support for Routing Instances FAQs on page 25
- SNMP Counters FAQs on page 27

SNMP Traps and Informs FAQs

This section presents frequently asked questions and answers related to SNMP traps and informs.

Does the Junos OS impose any rate limiting on SNMP trap generation?

The Junos OS implements a trap-queuing mechanism to limit the number of traps that are generated and sent.

If a 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, 2, 4, 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 traps in the queue are deleted.

Junos OS also has a throttle threshold mechanism to control the number of traps sent (default 500 traps) during a particular throttle interval (default 5 seconds). This helps ensure consistency in trap traffic, especially when a large number of traps are generated due to interface status changes.

The throttle interval begins when the first trap arrives at the throttle. All traps within the throttle threshold value are processed, and traps exceeding the threshold value are queued. The maximum size of all trap queues (the throttle queue and the destination queue) is 40,000 traps. The maximum size of any one queue is 20,000 traps. When a trap is added to the throttle queue, or if the throttle queue has exceeded the maximum size, the trap is moved to the top of the destination queue. Further attempts to send the trap from the destination queue are stopped for a 30-second period, after which the destination queue restarts sending the traps.

NOTE: For the Juniper Networks EX Series Ethernet Switch, the maximum size of all trap queues (the throttle queue and the destination queue) is 1,000 traps. The maximum size for any one queue on the EX Series is 500 traps.

I did not see a trap when I had a syslog entry with a critical severity. Is this normal? Can it be changed?

Not every syslog entry with critical severity is a trap. However, you can convert any syslog entry to a trap using the **event-options** statement.

The following example shows how to configure a **jnxSyslogTrap** whenever an **rpd_ldp_nbrdown** syslog entry message error occurs.

```
user@host> show event-options
policy snmptrap {
   events rpd_ldp_nbrdown;
   then {
      raise-trap;
   }
}
```

Are SNMP traps compliant with the Alarm Reporting Function (X.733) on the Junos OS?

No, SNMP traps on the Junos OS are not X.733 compliant.

Can I set up filters for traps or informs?

Traps and informs can be filtered based on the trap category and the object identifier. You can specify categories of traps to receive per host by using the **categories** statement at the **[edit snmp trap-group *trap-group*]** hierarchy level. 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;
   }
}
```

The Junos OS also has a more advanced filter option (**notify-filter**) for filtering specific traps or a group of traps based on their object identifiers.

The SNMPv3 configuration also supports filtering of SNMPv1 and SNMPv2 traps and excluding Juniper Networks enterprise-specific configuration management traps, as shown in the following configuration 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;
}
}
```

Can I simulate traps on a device?

Yes, you can use the **request snmp spoof-trap** ***trap name*** command for simulating a trap to the NMS that normally receives your device's traps. You can also add required values using the **variable-bindings** parameter.

The following example shows how to simulate a trap to the local NMS using variable bindings:

```
user@host> request snmp spoof-trap linkDown variable-bindings "ifIndex[116]=116,
  ifAdminStatus[116]=1 ,ifOperStatus[116]=2 , ifName[116]=ge-1/0/1"
```

How do I generate a warm start SNMPv1 trap?

When the SNMP process is restarted under normal conditions, a warm start trap is generated if the system up time is more than 5 minutes. If the system up time is less than 5 minutes, a cold start trap is generated.

The NMS sees only the MIB OIDs and numbers, but not the names of the SNMP traps. Why?

Before the NMS can recognize the SNMP trap details, such as the names of the traps, it must first compile and understand the MIBs and then parse the MIB OIDs.

In the Junos OS, how can I determine to which category a trap belongs?

For a list of common traps and their categories, see Juniper Networks Enterprise-Specific SNMP Version 1 Traps and Juniper Networks Enterprise-Specific SNMP Version 2 Traps in the *Junos OS SNMP MIBs and Traps Reference* document.

Can I configure a trap to include the source IP address?

Yes, you can configure the **source-address**, **routing-instance**, or **logical-instance** name for the source IP address using the **trap-options** command:

```
user@host> show snmp trap-options
source-address 10.1.1.1;
```

Can I create a custom trap?

Yes, you can use the **jnxEventTrap** event script to create customized traps as needed.

In the following example, a Junos OS operations (op) script is triggered when a **UI_COMMIT_NOT_CONFIRMED** event is received. The Junos OS op script matches the complete message of the event and generates an SNMP trap.

Example: Junos OS Op Script

```
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";

param $event;
param $message;

match / {

    /*
     * trapm utilty wants the following characters in the value to be escaped
     * '[', ']', ' ', '=', and ','
     */
    var $event-escaped = {
        call escape-string($text = $event, $vec = '[] =,');
    }
```

```
    var $message-escaped = {
        call escape-string($text = $message, $vec = '[] =,');
    }

    <op-script-results> {
 var $rpc = <request-snmp-generate-trap> {
     <trap> "jnxEventTrap";
     <variable-bindings> "jnxEventTrapDescr[0]='Event-Trap' , "
  _ "jnxEventAvAttribute[1]='event' , "
  _ "jnxEventAvValue[1]='" _ $event-escaped _ "' , "
  _ "jnxEventAvAttribute[2]='message' , "
  _ "jnxEventAvValue[1]='" _ $message-escaped _ "'";
 }

 var $res = jcs:invoke($rpc);
    }
}

template escape-string ($text, $vec) {

    if (jcs:empty($vec)) {
        expr $text;

    } else {
        var $index = 1;
        var $from = substring($vec, $index, 1);
        var $changed-value = {
            call replace-string($text, $from) {
                with $to = {
                    expr "\\";
                    expr $from;
                 }
            }
        }

        call escape-string($text = $changed-value, $vec = substring($vec, $index

 + 1));
    }
}

template replace-string ($text, $from, $to) {

    if (contains($text, $from)) {
        var $before = substring-before($text, $from);
        var $after = substring-after($text, $from);
        var $prefix = $before _  $to;

        expr $before;
        expr $to;
        call replace-string($text = $after, $from, $to);

    } else {
        expr $text;
    }
}
```

After creating your customized trap, you must configure a policy on your device to tell the device what actions to take after it receives the trap.

Here is an example of a configured policy under the **[edit event-options]** hierarchy:

```
[edit event-options]
user@host> show
policy trap-on-event {
   events UI_COMMIT_NOT_CONFIRMED;
   attributes-match {
      UI_COMMIT_NOT_CONFIRMED.message matches complete;
   }
   then {
      event-script ev-syslog-trap.junos-op {
         arguments {
            event UI_COMMIT_NOT_CONFIRMED;
            message "{$$.message}";
         }
      }
   }
}
```

Can I disable link up and link down traps on interfaces?

Yes, link up and link down traps can be disabled in the interface configuration. To disable the traps, use the **no-traps** statement at the **[edit interfaces** *interface-name* **unit** *logical-unit-number***]** and **[edit logical-systems** *logical-system-name* **interfaces** *interface-name* **unit** *logical-unit-number***]** hierarchies for physical and logical interfaces.

```
(traps | no-traps);
```

I see the link up traps on logical interfaces, but I do not see the link down traps. Is this normal behavior?

The Junos OS does not send link down traps for logical interfaces if the physical interface is down to prevent flooding alarms for the same root cause. However, when the physical interface and logical interfaces come back up, traps are sent indicating link up. This is because the physical interface coming up does not necessarily mean that the logical interfaces are also coming up.

Related Documentation

- Junos OS SNMP Support FAQs on page 2
- Junos OS MIBs FAQs on page 3
- Junos OS SNMP Configuration FAQs on page 11
- SNMPv3 FAQs on page 15
- SNMP Interaction with Juniper Networks Devices FAQs on page 17
- SNMP Support for Routing Instances FAQs on page 25
- SNMP Counters FAQs on page 27

Junos OS Dual Routing Engine Configuration FAQs

This section presents frequently asked questions and answers related to the configuration of dual Routing Engines.

The SNMP configuration should be identical between the Routing Engines when configuring for continued communication. However, we recommend having separate Routing Engine IDs configured for each Routing Engine, when using SNMPv3.

In my system, the MIB object snmpEngineBoots is not in sync between two Routing Engines in a dual Routing Engine device. Is this normal behavior?

Yes. This is the normal behavior. Each Routing Engine runs its own SNMP process (snmpd) agent, allowing each Routing Engine to maintain its own engine boots.

Is there a way to identify that an address belongs to RE0, RE1, or the master Routing Engine management interface (fxp0) by looking at an SNMP walk?

No. When you do an SNMP walk on the device, it only displays the master Routing Engine management interface address.

What is the best way to tell if the current IP address belongs to fxp0 or a Routing Engine, from a CLI session?

Routing Engines are mapped with the **fxp0** interface. This means that when you query RE0, the ifTable reports the **fxp0** interface address of RE0 only. Similarly, if you query RE1, the ifTable reports the **fxp0** interface address of RE1 only.

When there is a failover, the master hostname is changed since the hostname belongs to the Routing Engine. Is this correct?

Yes. You can configure the same hostname or different hostnames. Either would work.

If only the master IP address is configured (for example, 192.168.2.5), and the **sysDescr.0** object has the same the string configured on both of the Routing Engines, then even after a switchover, the **sysDescr.0** object returns the same value. The following sample shows the results you get by using the **snmpget** command:

```
bng-junos-pool02: /c/svivek/PR_BRANCH/src> snmpget -c jnpr -v2c 192.168.2.5
sysDescr.0 system.sysDescr.0 = foo
```

SNMP Support for Routing Instances FAQs

This section presents frequently asked questions and answers related to how SNMP supports routing instances.

Can the SNMP manager access data for routing instances?

Yes, the Junos OS enables SNMP managers for all routing instances to request and manage SNMP data related to the corresponding routing instances and logical system networks.

Two different routing instance behaviors can occur, depending on where the clients originate:

- Clients from routing instances other than the default can access MIB objects and perform SNMP operations only on the logical system networks to which they belong.

- Clients from the default routing instance can access information related to all routing instances and logical system networks.

Routing instances are identified by either the context field in SNMPv3 requests or encoded in the community string in SNMPv1 or SNMPv2c requests.

When encoded in a community string, the routing instance name appears first and is separated from the actual community string by the @ character.

To avoid conflicts with valid community strings that contain the @ character, the community is parsed only if typical community string processing fails. For example, if a routing instance named **RI** is configured, an SNMP request with **RI@public** is processed within the context of the **RI** routing instance. Access control (including views, source address restrictions, and access privileges) is applied according to the actual community string (the set of data after the @ character—in this case **public**). However, if the community string **RI@public** is configured, the PDU is processed according to that community, and the embedded routing instance name is ignored.

Logical systems perform a subset of the actions of a physical router and have their own unique routing tables, interfaces, policies, and routing instances. When a routing instance is defined within a logical system, the logical system name must be encoded along with the routing instance using a slash (/) to separate the two. For example, if the routing instance **RI** is configured within the logical system **LS**, that routing instance must be encoded within a community string as **LS/RI@public**. When a routing instance is configured outside a logical system (within the default logical system), no logical system name, or **/** character, is needed.

Additionally, when a logical system is created, a default routing instance named **default** is always created within the logical system. This name should be used when querying data for that routing instance, for example **LS/default@public**. For SNMPv3 requests, the name ***logical system/routing instance*** should be identified directly in the context field.

Can I access a list of all routing instances on a device?

Yes, you can access a list of all the routing instances on a device using the vacmContextName object in the SNMP-VIEW-BASED-ACM MIB. In SNMP, each routing instance becomes a VACM context; this is why the routing instances appear in the vacmContextName object.

Can I access a default routing instance from a client in another logical router or routing instance?

No, the SNMP agent can only access data of the logical router to which it is connected.

Related Documentation

- Junos OS SNMP Support FAQs on page 2
- Junos OS MIBs FAQs on page 3
- Junos OS SNMP Configuration FAQs on page 11
- SNMPv3 FAQs on page 15
- SNMP Interaction with Juniper Networks Devices FAQs on page 17

- SNMP Traps and Informs FAQs on page 19
- SNMP Counters FAQs on page 27

SNMP Counters FAQs

This section presents frequently asked questions and answers related to SNMP counters.

Which MIB should I use for interface counters?

Interface management over SNMP is based on two tables: the **ifTable** and its extension the **ifXTable**. Both are described in RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II* and RFC 2233, *The Interfaces Group MIB using SMIv2*.

Interfaces can have several layers, depending on the media, and each sublayer is represented by a separate row in the table. The relationship between the higher layer and lower layers is described in the **ifStackTable**.

The **ifTable** defines 32-bit counters for inbound and outbound octets (ifInOctets/ifOutOctets), packets (ifInUcastPkts/ifOutUcastPkts, ifInNUcastPkts /ifOutNUcastPkts), errors, and discards.

The **ifXTable** provides similar 64-bit counters, also called high capacity (HC) counters, for inbound and outbound octets (ifHCInOctets/ifHCOutOctets) and inbound packets (ifHCInUcastPkts).

When should 64-bit counters be used?

It is always good to use 64-bit counters because they contain statistics for both low and high capacity components.

Are the SNMP counters ifInOctets and ifOutOctets the same as the command reference show interfaces statistics in and out counters?

Yes, these are the same, but only if SNMP is enabled when the router boots up. If you power on a Juniper Networks device and then enable SNMP, the SNMP counters start from 0. SNMP counters do not automatically receive their statistics from the **show** command output. Similarly, using the **clear statistics** command does not clear the statistics that the SNMP counters collected, which can cause a discrepancy in the data that is seen by both processes.

Do the SNMP counters ifInOctets and ifOutOctets include the framing overhead for Point-to-Point Protocol (PPP) and High-Level Data Link Control (HDLC)?

Yes.

Related Documentation

- Junos OS SNMP FAQs Overview on page 1
- Junos OS MIBs FAQs on page 3
- Junos OS SNMP Support FAQs on page 2
- SNMPv3 FAQs on page 15