JUNIPER NETWORKS

Network Configuration Example

Logging Network Statistics Using Accounting Profiles

Release
11.4

Published: 2012-01-30

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

Network Configuration Example Logging Network Statistics Using Accounting Profiles
Release 11.4

Revision History
October 2011—R1 Junos OS 11.4

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

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. 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 describes the accounting profiles feature and how it can be used as an alternative to SNMP polling. The document also provides multiple configuration examples for accounting profiles.

Accounting Profiles: An Alternative to SNMP Statistics

SNMP enables users to monitor network devices from a central location. Juniper Networks provides many different platforms that support SNMP on the Junos operating system (Junos OS). Junos OS includes an onboard SNMP agent that provides remote management applications with access to detailed information about the devices in the network. The SNMP agent exchanges network management information with the SNMP network management system (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.

When network problems occur, SNMP requests might not reach the device, or the SNMP replies might not reach the server. After the problem is resolved, the SNMP monitoring application gets the sum of the minutes and can only average it over the missing time spans. This means that the time you most need to know what is going on in your network is the time you are unable to get the statistics.

Delays and time-outs (where the response does not arrive before the SNMP monitor application times out) provide an equally poor view of the network. If the request is delayed, the time period between requests is not consistent, leading to poor data. If a time-out results in discarded packets, the device is likely wasting resources generating statistics that are being discarded.

Juniper Networks devices can collect various types of data about the traffic passing through the device using a feature called *accounting profiles*. You can set up accounting profiles to avoid the limitations of using SNMP polling. You can set up one or more accounting profiles that specify some common characteristics of the required data. Setting up accounting profiles provides you with persistent data even during network outages. This data can be transferred when the network is restored and can be used for troubleshooting and monitoring.

Understanding Accounting Profiles

Accounting profiles are implemented in the Packet Forwarding Engine process (pfed). The pfed process routinely requests interface statistics from the Packet Forwarding Engine to ensure that accurate values are cached in the Routing Engine kernel. Accounting profiles enable you to store these statistics in a file on the hard disk.

You can configure multiple accounting profiles, as described in Table 1 on page 2.

Table 1: Types of Accounting Profiles

Type of Profile	Description
Interface profile	Collects the specified error and statistic information
Filter profile	Collects the byte and packet counts for the counter names specified in the filter profile
MIB profile	Collects selected MIB statistics
Routing Engine profile	Collects selected Routing Engine statistics
Class usage profile	Collects class usage statistics

Each of these profile types have specific fields that can be recorded as shown in Table 2 on page 2.

Table 2: Profile Types and Corresponding Fields

Profile Type	Fields
interface-profile	**input-bytes**–Input bytes
	input-errors–Generic input error packets
	input-multicast–Input packets arriving by multicast
	input-packets–Input packets
	input-unicast–Input unicast packets
	output-bytes–Output bytes
	output-errors–Generic output error packets
	output-multicast–Output packets sent by multicast
	output-packets–Output packets
	output-unicast–Output unicast packets
	rpf-check-bytes–Bytes failing the IPv4 reverse-path-forwarding check
	rpf-check-packets–Packets failing the IPv4 reverse-path-forwarding check
	rpf-check6-bytes–Bytes failing the IPv6 reverse-path-forwarding check
	rpf-check6-packets–Packets failing the IPv6 reverse-path-forwarding check
	unsupported-protocol–Packets for an unsupported protocol

Table 2: Profile Types and Corresponding Fields *(continued)*

Profile Type	Fields
filter-profile	**counter**–Counter name configured at the **[edit firewall filter *filter-name* term *rule-name* then count *counter-name*]** hierarchy level
mib-profile	**object-names**–Set of SNMP object identifiers (OIDs)
routing-engine-profile	**cpu-load-1**–Average system load (number of processes in the run queue) over the last 1 minute
	cpu-load-15–Average system load (number of processes in the run queue) over the last 15 minutes
	cpu-load-5–Average system load (number of processes in the run queue) over the last 5 minutes
	date–Date in YYYYMMDD format
	host-name–Router hostname
	memory-usage–Instantaneous active memory usage
	time-of-day–Time of day in HHMMSS format
	total-cpu-usage–Total CPU usage percentage
	uptime–Time (in seconds) since last reboot
class-usage-profile	**source-class**–A set of named source classes

By saving these statistics locally, data values are recorded regardless of network conditions. Because the requests are generated locally, they are sent regardless of network conditions. By driving both sides locally, data is consistent and of higher quality. Handling data locally avoids encoding and decoding issues, providing better performance.

Related Documentation

- Configuring the Interface Profile
- Configuring the Filter Profile
- Understanding Source Class Usage and Destination Class Usage Options
- Configuring SCU or DCU
- Configuring SCU on a Virtual Loopback Tunnel Interface
- Configuring Class Usage Profiles
- Configuring the MIB Profile
- Configuring the Routing Engine Profile

Example: Configuring an Accounting Profile for Statistics Collection

This example shows how to configure an accounting profile for statistics collection.

Requirements

Accounting profiles are supported on M Series Multiservice Edge Routers, MX Series 3D Universal Edge Routers, T Series Core Routers, SRX Series Services Gateways, and EX Series Ethernet Switches with Junos OS Release 7.4 and later. This example uses the following hardware and software components:

- Junos OS Release 11.2
- M7i Multiservice Edge router

Overview

The procedures in this example describe four tasks for creating a sample **ifstats** interface profile. The interface profile **ifstats** specifies that statistics are collected every 5 minutes, and the statistics are written to the file **/var/log/ifstats.log.**

Configuration

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see Using the CLI Editor in Configuration Mode in the *Junos OS CLI User Guide*.

The accounting profile configuration consists of:

- Configuring the fields that need to be recorded
- Configuring when to record the fields
- Configuring the instances that need accounting
- Saving the recorded data

Configuring Fields to Be Recorded

The fields to record are configured by choosing a profile type, specifying a name for the profile, and then listing the field names to be recorded. For example, to configure the **ifstats** interface profile:

1. Configure the desired fields for the **ifstats** interface profile at the **[edit accounting-options interface-profile ifstats fields]** hierarchy level.

```
[edit accounting-options interface-profile ifstats fields]
user@host#  set input-bytes output-bytes input-errors output-errors
  unsupported-protocol rpf-check-bytes
```

2. Verify the configuration by using the **show** command at the **[edit accounting-options]** hierarchy level.

```
[edit accounting-options]
user@host# show
interface-profile ifstats {
    fields {
        input-bytes;
        output-bytes;
        input-errors;
        output-errors;
        unsupported-protocol;
        rpf-check-bytes;
    }
}
```

Configuring When to Record Fields

To capture the fields at regular intervals:

1. Configure the time interval for the **ifstats** interface profile.

   ```
   [edit accounting-options]
   user@host#  set interface-profile ifstats interval 5
   ```

2. Verify the configuration by using the **show** command at the **[edit accounting-options]** hierarchy level.

```
[edit accounting-options]
user@host# show
interface-profile ifstats {
    interval 5;
    fields {
        input-bytes;
        output-bytes;
        input-errors;
        output-errors;
        unsupported-protocol;
        rpf-check-bytes;
    }
}
```

Configuring Instances That Need Accounting

To configure the interfaces on which the accounting needs to be performed, apply the interface profile to a physical interface or a logical interface by including the **accounting-profile** statement at either the **[edit interfaces *interface-name*]** or the **[edit interfaces *interface-name* unit *logical-unit-number*]** hierarchy level. For example, perform the following steps to apply the **ifstats** accounting profile to the **fe-0/2/3** interface:

1. Apply the **ifstats** accounting profile to the **fe-0/2/3** interface.

   ```
   [edit interfaces]
   user@host# set fe-0/2/3 unit 0 accounting-profile ifstats
   ```

2. Verify the configuration by using the **show** command at the **[edit interfaces]** hierarchy level.

```
[edit interfaces]
user@host# show
fe-0/2/3 {
    unit 0 {
        accounting-profile ifstats;
    }
}
```

Saving the Recorded Data

To configure the router to save the recorded data to an accounting-data log file:

1. Configure the accounting-data log file name.

   ```
   [edit accounting-options]
   user@host# edit file ifstats.log
   ```

2. Configure the size of the accounting-data log file.

   ```
   [edit accounting-options file ifstats.log]
   user@host# set size 262144
   ```

3. Verify the configuration by using the **show** command at the **[edit accounting-options]** hierarchy level.

   ```
   [edit accounting-options]
   user@host# show
   file ifstats.log {
       size 256k;
   }
   interface-profile ifstats {
      interval 5;
       fields {
           input-bytes;
           output-bytes;
           input-errors;
           output-errors;
           unsupported-protocol;
           rpf-check-bytes;
       }
   }
   ```

Related Documentation

- Configuring Accounting-Data Log Files
- Example: Configuring Statistics Collection for a Standard Firewall Filter

Using the Recorded Data

After you have set up an accounting profile, you can use the recorded data for monitoring and troubleshooting purposes.

Understanding and Viewing the Data

This section covers the following topics:

Understanding the Accounting-Data Log File Format

In the accounting-data log file, the data is saved in a modified comma separated value (CSV) format. In this format, the CSV lines are combined with lines that define the layout of the fields used by the CSV data. Each CSV line begins with two standard fields, a layout name, and a timestamp. The file also contains a format definition line that maps the layout name to the fields that appear in the lines tagged with that layout name. This format enables multiple layouts of data inside a single file, and also allows the definition of the layout to change over time without requiring a new file.

For example, consider the following accounting-data log file of an interface profile that saves input bytes:

```
#profile-layout ifstats,epoch-timestamp,interface-name,input-bytes
ifstats,1266020093,fe-0/2/3.0,1592386
ifstats,1266020123,fe-0/2/3.0,1762869
```

This layout helps you modify the CSV data into structured XML data, using the layout names and field names for both encoding and filtering. For example, the previous accounting-data log file can be represented in XML as follows:

```
<interface-accounting-record>
  <profile-layout>ifstats</profile-layout>
  <epoch-timestamp>1266020093</epoch-timestamp>
  <interface-name>fe-0/2/3.0</interface-name>
  <input-bytes>1592386</input-bytes>
</interface-accounting-record>
```

Multiple accounting profiles can be configured to record data in the same accounting-data log file. For example, you can have interface statistics and CPU statistics recorded in a single log file so that you can transfer these statistics together. Configuring multiple accounting profiles to update the same accounting-data log file has an impact on the format of the log file. This is because all the formats are defined and referenced in the same log file. The contents of a sample accounting-data log file that records the data of two accounting profiles are shown in the following sample output:

```
#fmt1 n1a,n1b,n1c
#fmt2 n2a,n2b,n2c
fmt1,v1a,v1b,v1c
fmt2,v2a,v2b,v2c
```

```
  fmt1,v3a,v3b,v3c
  fmt2,v4a,v4b,v4c
```

Therefore, this not only provides simplicity in the form of CSV output, but also provides sufficient information to produce XML content.

Related Documentation

- Configuring Accounting-Data Log Files

Commands for Viewing Accounting Profile Information

Two commands can be used to view accounting profile information.

The **show accounting profile *profile-name*** command displays the information regarding the profile. Sample output of a **show accounting profile *profile-name*** command follows:

```
user@host> show accounting profile ifstats
Profile ifstats
Sampling interval: 1 minute(s),  Profile Usage Count: 16
File ifstats: maximum size 2097152, maximum number 10, bytes written 20454
Transfer Interval: 30 minute(s),  Next Scheduled Transfer: 2010-02-16-14:44:24
Column Labels:
  profile-layout
  epoch-timestamp
  interface-name
  input-bytes

Interface Name              Next Scheduled Collection
fe-0/0/0.0                  2010-02-16-14:34:24
fe-0/0/1.0                  2010-02-16-14:34:27
fe-0/0/2.0                  2010-02-16-14:34:30
fe-0/0/3.0                  2010-02-16-14:34:33
```

The **show accounting records *profile-name*** command displays the records that are stored in the accounting-data log file of the profile. Sample output of a **show accounting records *profile-name*** command follows:

```
user@host> show accounting records ifstats
Timestamp: 1266329666, Interface Name: fe-0/0/1.0 (SNMP Index 137)
              3644676   Input Bytes
Timestamp: 1266329669, Interface Name: fe-0/0/2.0 (SNMP Index 138)
              3644012   Input Bytes
Timestamp: 1266329672, Interface Name: fe-0/0/3.0 (SNMP Index 139)
              3952848   Input Bytes
```

Related Documentation

- show accounting profile
- show accounting records

Remote Procedure Calls for Viewing Accounting Profile Information

You can access the **<get-accounting-profile-information>** and **<get-accounting-record-information>** remote procedure calls (RPCs) using XML APIs to fetch the profile information. The **<get-accounting-profile-information>** RPC provides the same information that the **show accounting profile *profile-name*** command provides.

Similarly, the **<get-accounting-record-information>** RPC provides the same information that the **show accounting records *profile-name*** command provides. The **since** argument to this RPC provides accounting statistics since the specified time.

Script Examples

Like the **jnxUtility** MIB, you can develop scripts to use the recorded accounting profiles information. The following sample Stylesheet Language Alternative Syntax (SLAX) code illustrates how to add an entry to an accounting-data log file:

```
var $add = <add-accounting-file-record> {
  <file> $file;
  <layout> $layout;
  <fields> "target,loss,min,max,ave,std";
  <data> $target _ "," _
      $res/packet-loss _ "," _
      $res/rtt-minimum _ "," _
      $res/rtt-maximum _ "," _
      $res/rtt-average _ "," _
      $res/rtt-stddev;
}
var $done = jcs:invoke($add);
```

In the preceding code block, **$file** is the name of the accounting-data log file that is configured at the **[edit accounting-options file]** hierarchy level. Additionally, **$layout** is the name assigned to the format of the accounting-data log file. Sample output for this code block follows:

```
#layout target,loss,min,max,ave,std
layout,10.1.2.3,5,48,540,218,25
```

In the preceding sample output, the string *layout* indicates that **$layout** is assigned the value **layout**.

The following sample SLAX code illustrates how to read data from an accounting-data log file:

```
var $rpc = <get-accounting-file-record-information> {
  <file> $file;
  <since> -86400;
}
var $out = jcs:invoke($rpc);
```

Similarly, you can develop complex scripts to archive or compress profile data, to provide a historical view of how the data changes over time, or to meet any of your custom requirements. For example:

- You can develop a script that works in two passes. The first pass saves the raw data of an accounting profile. The second pass inspects the saved data looking for values that are above the standard deviation. If such values are found, an SNMP trap is sent to inform the NMS about the heavy traffic.
- You can develop a script that records the path for an MPLS label-switched path (LSP), and then report when the LSP changes to a new path.

You can develop several such scripts that suit your requirements.