



---

## Junos OS

### Junos Telemetry Interface Feature Guide



---

Modified: 2017-05-25

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
[www.juniper.net](http://www.juniper.net)

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

*Junos OS Junos Telemetry Interface Feature Guide*  
Copyright © 2017, Juniper Networks, Inc.  
All rights reserved.

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

	About the Documentation . . . . .	ix
	Documentation and Release Notes . . . . .	ix
	Supported Platforms . . . . .	ix
	Using the Examples in This Manual . . . . .	ix
	Merging a Full Example . . . . .	x
	Merging a Snippet . . . . .	x
	Documentation Conventions . . . . .	xi
	Documentation Feedback . . . . .	xiii
	Requesting Technical Support . . . . .	xiii
	Self-Help Online Tools and Resources . . . . .	xiii
	Opening a Case with JTAC . . . . .	xiv
<b>Part 1</b>	<b>Junos Telemetry Interface Overview</b>	
<b>Chapter 1</b>	<b>Overview of the Junos Telemetry Interface . . . . .</b>	<b>3</b>
	Overview of the Junos Telemetry Interface . . . . .	3
	Telemetry Sensors and Data Models . . . . .	4
	Uses and Benefits . . . . .	4
<b>Part 2</b>	<b>Configuring Native Sensors</b>	
<b>Chapter 2</b>	<b>Export Format of Collected Data . . . . .</b>	<b>9</b>
	Understanding the Junos Telemetry Interface Export Format of Collected Data . . . . .	10
	Understanding the Sensor Data Encapsulation Format . . . . .	10
<b>Chapter 3</b>	<b>Configuring Junos Telemetry Interface (CLI Procedure) . . . . .</b>	<b>15</b>
	Configuring a Junos Telemetry Interface Sensor (CLI Procedure) . . . . .	15
	Configuring an Export Profile . . . . .	16
	Configuring a Streaming Server Profile . . . . .	18
	Configuring a Sensor Profile . . . . .	19
	Verifying Junos Telemetry Interface Sensor Configuration . . . . .	20
<b>Chapter 4</b>	<b>Junos Telemetry Interface Configuration Statements and Operational Commands . . . . .</b>	<b>23</b>
	export-profile (Junos Telemetry Interface) . . . . .	24
	lsp-telemetry (Junos Telemetry Interface) . . . . .	27
	sensor (Junos Telemetry Interface) . . . . .	28
	sensor-based-stats (Junos Telemetry Interface) . . . . .	33
	streaming-server (Junos Telemetry Interface) . . . . .	34
	show agent sensors . . . . .	36

<b>Chapter 5</b>	<b>Decoding Data . . . . .</b>	<b>39</b>
	Decoding Junos Telemetry Interface Data With UNIX Utilities . . . . .	39
	Preparing the Collector to Decode Data . . . . .	39
	Decoding Data on the Collector . . . . .	40
<b>Part 3</b>	<b>Configuring gRPC Sensors</b>	
<b>Chapter 6</b>	<b>OpenConfig and gRPC for Junos Telemetry Interface . . . . .</b>	<b>51</b>
	Understanding OpenConfig and gRPC on Junos Telemetry Interface . . . . .	51
	Network Agent Software . . . . .	51
	Using OpenConfig for Junos OS to Enable Junos Telemetry Interface . . . . .	52
	Using gRPC to Stream Data . . . . .	52
	Installing the Network Agent Package (Junos Telemetry Interface) . . . . .	54
	Configuring gRPC for the Junos Telemetry Interface . . . . .	57
	Guidelines for gRPC Sensors (Junos Telemetry Interface) . . . . .	59
	Supported gRPC Sensors . . . . .	59
<b>Part 4</b>	<b>Best Practices</b>	
<b>Chapter 7</b>	<b>Best Practices for Implementing the Junos Telemetry Interface . . . . .</b>	<b>81</b>
	Guidelines for Specifying Data Reporting Intervals Junos Telemetry Interface . . . . .	81
	How to Determine the Reporting Interval for a System Resource . . . . .	81
	Guidelines for Aggregating Junos Telemetry Interface Data . . . . .	82
	Aggregating Data Over Fixed Time Spans . . . . .	82
	Example: Aggregating Data for Gauge Metrics . . . . .	82
	Example: Aggregating Data for Cumulative Statistics . . . . .	83
	Aggregating Data From Multiple Sources . . . . .	84
	Example: Aggregating Data from Multiple Sources . . . . .	85
	Aggregating Data for Multiple Metrics . . . . .	85
	Example: Aggregating Multiple Metric Values . . . . .	86

# List of Figures

Part 1	Junos Telemetry Interface Overview	
Chapter 1	Overview of the Junos Telemetry Interface .....	3
	Figure 1: Telemetry Streaming for Performance Management .....	5



# List of Tables

	<b>About the Documentation</b> . . . . .	<b>ix</b>
	Table 1: Notice Icons . . . . .	xi
	Table 2: Text and Syntax Conventions . . . . .	xii
<b>Part 2</b>	<b>Configuring Native Sensors</b>	
<b>Chapter 2</b>	<b>Export Format of Collected Data</b> . . . . .	<b>9</b>
	Table 3: Individual Data Element Types in the gpb Message . . . . .	12
<b>Chapter 4</b>	<b>Junos Telemetry Interface Configuration Statements and Operational Commands</b> . . . . .	<b>23</b>
	Table 4: resource statement Options . . . . .	30
	Table 5: show agent sensors Output Fields . . . . .	36
<b>Part 3</b>	<b>Configuring gRPC Sensors</b>	
<b>Chapter 6</b>	<b>OpenConfig and gRPC for Junos Telemetry Interface</b> . . . . .	<b>51</b>
	Table 6: Telemetry RPCs . . . . .	53
	Table 7: gRPC Sensors . . . . .	60
<b>Part 4</b>	<b>Best Practices</b>	
<b>Chapter 7</b>	<b>Best Practices for Implementing the Junos Telemetry Interface</b> . . . . .	<b>81</b>
	Table 8: Telemetry Data Values . . . . .	83





# About the Documentation

- Documentation and Release Notes on page ix
- Supported Platforms on page ix
- Using the Examples in This Manual on page ix
- Documentation Conventions on page xi
- Documentation Feedback on page xiii
- Requesting Technical Support on page xiii

## Documentation and Release Notes

---

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

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

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

## Supported Platforms

---

For the features described in this document, the following platforms are supported:

- MX Series
- PTX Series
- QFX Series

## Using the Examples in This Manual

---

If you want to use the examples in this manual, you can use the **load merge** or the **load merge relative** command. These commands cause the software to merge the incoming configuration into the current candidate configuration. The example does not become active until you commit the candidate configuration.

If the example configuration contains the top level of the hierarchy (or multiple hierarchies), the example is a *full example*. In this case, use the **load merge** command.

If the example configuration does not start at the top level of the hierarchy, the example is a *snippet*. In this case, use the **load merge relative** command. These procedures are described in the following sections.

## Merging a Full Example

To merge a full example, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration example into a text file, save the file with a name, and copy the file to a directory on your routing platform.

For example, copy the following configuration to a file and name the file **ex-script.conf**. Copy the **ex-script.conf** file to the **/var/tmp** directory on your routing platform.

```
system {
  scripts {
    commit {
      file ex-script.xml;
    }
  }
}
interfaces {
  fxp0 {
    disable;
    unit 0 {
      family inet {
        address 10.0.0.1/24;
      }
    }
  }
}
```

2. Merge the contents of the file into your routing platform configuration by issuing the **load merge** configuration mode command:

```
[edit]
user@host# load merge /var/tmp/ex-script.conf
load complete
```

## Merging a Snippet

To merge a snippet, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration snippet into a text file, save the file with a name, and copy the file to a directory on your routing platform.

For example, copy the following snippet to a file and name the file **ex-script-snippet.conf**. Copy the **ex-script-snippet.conf** file to the **/var/tmp** directory on your routing platform.

```
commit {
```

```
file ex-script-snippet.xml; }
```

2. Move to the hierarchy level that is relevant for this snippet by issuing the following configuration mode command:

```
[edit]
user@host# edit system scripts
[edit system scripts]
```

3. Merge the contents of the file into your routing platform configuration by issuing the **load merge relative** configuration mode command:







```
[edit system scripts]
user@host# load merge relative /var/tmp/ex-script-snippet.conf
load complete
```

For more information about the **load** command, see [CLI Explorer](#).

## Documentation Conventions

[Table 1 on page xi](#) defines notice icons used in this guide.

**Table 1: Notice Icons**

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

[Table 2 on page xii](#) defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
<b>Bold text like this</b>	Represents text that you type.	To enter configuration mode, type the <b>configure</b> command:  user@host> <b>configure</b>
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> <b>show chassis alarms</b>  No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> <li>Introduces or emphasizes important new terms.</li> <li>Identifies guide names.</li> <li>Identifies RFC and Internet draft titles.</li> </ul>	<ul style="list-style-type: none"> <li>A policy <i>term</i> is a named structure that defines match conditions and actions.</li> <li><i>Junos OS CLI User Guide</i></li> <li>RFC 1997, <i>BGP Communities Attribute</i></li> </ul>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name:  [edit] root@# <b>set system domain-name</b> <i>domain-name</i>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> <li>To configure a stub area, include the <b>stub</b> statement at the [edit protocols ospf area area-id] hierarchy level.</li> <li>The console port is labeled <b>CONSOLE</b>.</li> </ul>
< > (angle brackets)	Encloses optional keywords or variables.	<b>stub</b> <default-metric <i>metric</i> >;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	<b>broadcast</b>   <b>multicast</b>  ( <i>string1</i>   <i>string2</i>   <i>string3</i> )
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	<b>rsvp { # Required for dynamic MPLS only</b>
[ ] (square brackets)	Encloses a variable for which you can substitute one or more values.	<b>community name members</b> [ <i>community-ids</i> ]
Indentation and braces ( { } )	Identifies a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i> ; retain; } } }
;(semicolon)	Identifies a leaf statement at a configuration hierarchy level.	

#### GUI Conventions

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

Convention	Description	Examples
<b>Bold text like this</b>	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none"> <li>In the Logical Interfaces box, select <b>All Interfaces</b>.</li> <li>To cancel the configuration, click <b>Cancel</b>.</li> </ul>
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select <b>Protocols&gt;Ospf</b> .

## Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can provide feedback by using either of the following methods:

- Online feedback rating system—On any page of the Juniper Networks TechLibrary site at <http://www.juniper.net/techpubs/index.html>, simply click the stars to rate the content, and use the pop-up form to provide us with information about your experience. Alternately, you can use the online feedback form at <http://www.juniper.net/techpubs/feedback/>.
- E-mail—Send your comments to [techpubs-comments@juniper.net](mailto:techpubs-comments@juniper.net). Include the document or topic name, URL or page number, and software version (if applicable).

## Requesting Technical Support

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

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <http://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

## Self-Help Online Tools and Resources

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

- Find CSC offerings: <http://www.juniper.net/customers/support/>
- Search for known bugs: <http://www2.juniper.net/kb/>
- Find product documentation: <http://www.juniper.net/techpubs/>
- Find solutions and answer questions using our Knowledge Base: <http://kb.juniper.net/>
- Download the latest versions of software and review release notes: <http://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <http://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <http://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <http://www.juniper.net/cm/>

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

## Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at <http://www.juniper.net/cm/>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <http://www.juniper.net/support/requesting-support.html>.

## PART 1

# Junos Telemetry Interface Overview

- Overview of the Junos Telemetry Interface on page 3





## CHAPTER 1

# Overview of the Junos Telemetry Interface

- [Overview of the Junos Telemetry Interface on page 3](#)

## Overview of the Junos Telemetry Interface

---

As the number of objects on the network and the metrics they generate have grown, the traditional models, such as SNMP, used to gather operational statistics for monitoring the health of a network, have imposed limits on network element scale and efficiency. The so-called pull model used by SNMP and the CLI, which requires additional processing to periodically poll the network element, directly limits scaling.

The Junos Telemetry Interface (JTI) overcomes these limits by relying on a so-called push model to deliver data asynchronously, which eliminates polling. A request to send data is sent once by a management station to stream periodic updates. As a result, JTI is highly scalable and can support the monitoring of thousands of objects in a network.



**NOTE:** Junos Telemetry Interface was introduced in Junos OS Release 15.1F3, on MX Series routers with interfaces configured on MPC1 through MPC6E, and on PTX Series routers with interfaces configured on FPC3. Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.

Starting with Junos OS Release 16.1R3, FPC1, FPC2, and dual Routing Engines on PTX Series routers are also supported.

Starting with Junos OS Release 17.2R1, QFX10000 and QFX5200 switches, and PTX1000 routers are also supported.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

- [Telemetry Sensors and Data Models on page 4](#)
- [Uses and Benefits on page 4](#)

## Telemetry Sensors and Data Models

The Junos Telemetry Interface enables you to provision sensors to collect and export data for various system resources, such as physical interfaces and firewall filters. Two data models, each of which uses a different mode of transport, are supported:

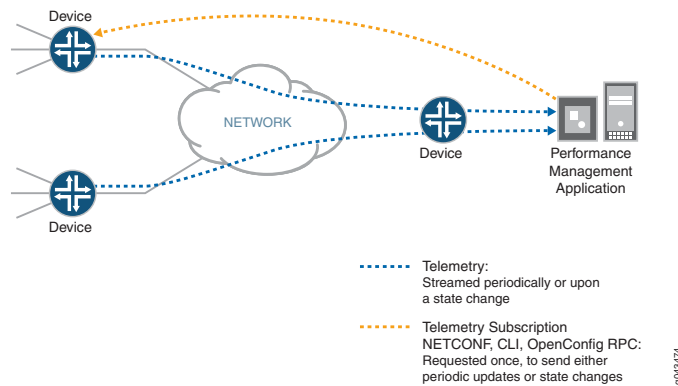
- An open and extensible data model defined by Juniper Networks. Data is generated as Google protocol buffers (gpb) structured messages. The files that define each **.proto** message are published on the Juniper Networks web site. Native sensors export data close to the source, such as the line card or network processing unit (NPU), using the User Datagram Protocol (UDP). Because this model features a distributed architecture, it scales easily.
- An OpenConfig data model that generates data as gpb messages in a universal key/value format. OpenConfig for Junos OS, which you must download, supports the YANG data models. gRPC remote procedure calls (gRPC) are used to provision sensors and to subscribe to and receive telemetry data. gRPC is based on TCP, and supports SSL encryption, so it is considered secure and reliable. If your Juniper Networks device is running a version of Junos OS with the upgraded FreeBSD kernel, this model requires you to download the Junos Network Agent package, which runs on the Routing Engine and provides interfaces to manage gRPC subscriptions. For other versions of Junos OS, Network Agent functionality is embedded in the software.

## Uses and Benefits

A primary function of the Junos Telemetry Interface is performance monitoring. Streaming data to a performance management system enables network administrators to measure trends in link and node utilization, and troubleshoot such issues as network congestion in real time, for example.

In a typical deployment, the network element, or device, streams duplicate data to two destination servers that function as performance management system collectors. Streaming data to two collectors provides redundancy. See [Figure 1 on page 5](#) for an illustration of how the performance management system collectors request data and how the device streams data. The device provisions sensors to collect and export data using command-line interface (CLI), configuration through NETCONF, or gRPC subscription calls. The collectors request data by initiating a telemetry subscription. Data is requested only once and is streamed periodically.

Figure 1: Telemetry Streaming for Performance Management



Other applications of the Junos Telemetry Interface include providing real-time data to support operational state synchronization between a network element and an external controller, such as the Northstar Controller, which automates the creation of traffic-engineering paths across the network. The NorthStar Controller can subscribe to telemetry data about certain network elements, such as label-switched path (LSP) statistics.

## Release History Table

Release	Description
17.2R1	Starting with Junos OS Release 17.2R1, QFX10000 and QFX5200 switches, and PTX1000 routers are also supported.
16.1R3	Starting with Junos OS Release 16.1R3, FPC1, FPC2, and dual Routing Engines on PTX Series routers are also supported.
15.1F5	Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.
15.1F3	Junos Telemetry Interface was introduced in Junos OS Release 15.1F3, on MX Series routers with interfaces configured on MPC1 through MPC6E, and on PTX Series routers with interfaces configured on FPC3.

## Related Documentation

- [Understanding the Junos Telemetry Interface Export Format of Collected Data on page 10](#)
- [Understanding OpenConfig and gRPC on Junos Telemetry Interface on page 51](#)



## PART 2

# Configuring Native Sensors

- [Export Format of Collected Data on page 9](#)
- [Configuring Junos Telemetry Interface \(CLI Procedure\) on page 15](#)
- [Junos Telemetry Interface Configuration Statements and Operational Commands on page 23](#)
- [Decoding Data on page 39](#)



## CHAPTER 2

# Export Format of Collected Data

- [Understanding the Junos Telemetry Interface Export Format of Collected Data on page 10](#)

## Understanding the Junos Telemetry Interface Export Format of Collected Data

---

The Junos Telemetry Interface supports two ways of exporting data in the Google protocol buffers (gpb) format:

- Through UDP from so-called native sensors that export data close to the source, such as the line card or network processing unit (NPU). Juniper Networks defines the data model, which is open and extensible.
- Through gRPC remote procedure calls (gRPC) that export data through the Routing Engine. The data model is defined by OpenConfig, which supports the use of vendor-neutral data models to configure and manage the network. OpenConfig for Junos OS supports the YANG data models. For platforms that are running a version of Junos OS based on an upgraded FreeBSD kernel only, you must install a separate package called Network Agent that functions as a gRPC server and terminates the RPC interfaces. For all other versions of Junos OS, the Network Agent functionality is embedded in the software. You must also install the OpenConfig for Junos OS module and the YANG models.

This section describes the format of data exported from native sensors using UDP. The data is encapsulated into a UDP header, which is in turn encapsulated in the IPv4 payload. This model of the Junos Telemetry Interface is based on a distributed architecture, through which the data generated by configured sensors is exported directly from the data plane, bypassing the control plane, and thus conserving these resources to perform other necessary functions.



**NOTE:** The Junos Telemetry Interface was introduced in Junos OS Release 15.1F3, on MX Series routers with interfaces configured on MPC1 through MPC6E, and on PTX Series routers with interfaces configured on FPC3. Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.

Starting with Junos OS Release 16.1R3, FPC1, FPC2, and dual Routing Engines on PTX Series routers are also supported.

Starting with Junos OS Release 17.2R1, QFX10000 and QFX5200 switches are also supported. On QFX5200 switches, only gRPC streaming is supported.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

- 
- [Understanding the Sensor Data Encapsulation Format on page 10](#)

## Understanding the Sensor Data Encapsulation Format

A native sensor exports data close to the source using UDP. Various types of telemetry data, such as physical interface statistics, firewall filter counter statistics, or statistics for label-switched paths (LSPs) can be exported. A sensor starts to emit data as soon as it is enabled.



The sensor data is represented as a single structured Google protocol buffers message, named **TelemetryStream**. The message, or **.proto** file, shown below, includes several attributes that identify the data source, such as a line card, a Packet Forwarding Engine, or a Routing Engine. The name of the configured sensor is also included. For more information about how to configure sensors, see [“Configuring a Junos Telemetry Interface Sensor \(CLI Procedure\)” on page 15](#). For a list of supported native sensors, see [sensor](#).

You must also download the **.proto** files for all the sensors supported to a streaming server or collector. From a Web browser, navigate to the All Junos Platforms software download URL on the Juniper Networks page: <http://www.juniper.net/support/downloads/>. After you select the name of the Junos OS platform and the release number, go to the **Tools** section and download the **Junos Telemetry Interface Data Model Files** package. For more information about configuring a streaming-server, see [streaming-server \(Junos Telemetry Interface\)](#).

### Google protocol buffers message Definition

Following is the message definition for **TelemetryStream** in the Google Protocol Buffers definition language. It shows several optional nested structures, such as **EnterpriseSensors**, which carry privately defined sensor data.

```
//
// This file defines the top level message used for all Juniper
// Telemetry packets encoded to the protocol buffer format.
// The top level message is TelemetryStream.
//

import "google/protobuf/descriptor.proto";

extend google.protobuf.FieldOptions {
    optional TelemetryFieldOptions telemetry_options = 1024;
}

message TelemetryFieldOptions {
    optional bool is_key           = 1;
    optional bool is_timestamp    = 2;
    optional bool is_counter      = 3;
    optional bool is_gauge        = 4;
}

message TelemetryStream {
    // router name or export IP address
    required string system_id      = 1 [(telemetry_options).is_key = true];

    // line card / RE (slot number)
    optional uint32 component_id   = 2 [(telemetry_options).is_key = true];

    // PFE (if applicable)
    optional uint32 sub_component_id = 3 [(telemetry_options).is_key = true];

    // configured sensor name
    optional string sensor_name    = 4 [(telemetry_options).is_key = true];

    // sequence number, monotonically increases for each
    // system_id, component_id, sub_component_id + sensor_name.
    optional uint32 sequence_number = 5;

    // timestamp (milliseconds since 00:00:00 UTC 1/1/1970)
```

```

        optional uint64 timestamp          = 6 [(telemetry_options).is_timestamp =
true];

        // major version
        optional uint32 version_major     = 7;

        // minor version
        optional uint32 version_minor     = 8;

        optional IETFSensors ietf         = 100;

        optional EnterpriseSensors enterprise = 101;
    }

    message IETFSensors {
        extensions 1 to max;
    }

    message EnterpriseSensors {
        extensions 1 to max;
    }

    extend EnterpriseSensors {
        // re-use IANA assigned numbers
        optional JuniperNetworksSensors juniperNetworks = 2636;
    }

    message JuniperNetworksSensors {
        extensions 1 to max;
    }

```

The **TelemetryStream** message also includes optional nested structures that carry different types of data. One structure carries enterprise, that is, privately defined data. Individual companies, such as Juniper Networks, define and maintain the attributes generated by enterprise sensors. Each company is assigned a unique attribute identifier. The current convention is to use IANA-assigned enterprise MIB identifiers for each attribute. For Juniper Networks, this assigned identifier is 2636.



**BEST PRACTICE:** To verify that a particular message type has been exported and received, check for those attributes under **TelemetryStream.enterprise.juniperNetworks** in the gpb message.

See [Table 3 on page 12](#) for descriptions of each element collected by sensor data, including semantics and corresponding schema.

**Table 3: Individual Data Element Types in the gpb Message**

Element Type	Description
Counter	An unsigned integer that increases monotonically. When it reaches its maximum value, it starts back at zero.

Table 3: Individual Data Element Types in the gpb Message (*continued*)

Element Type	Description
Gauge	An unsigned 32-bit or 64-bit integer that can increase or decrease in value. An example of the data represented by this element is the instantaneous value of a specific resource, such as queue depth or temperature.
Rate	Rate at which a base metric changes, such as a counter or a gauge. For this element type, units of measurement are defined explicitly (such as bits per second), as well as the interval over which the rate is collected.
Average	The average of several samples of a base metric. For example, an <i>average queue depth</i> data element would be calculated by averaging several elements of the queue depth. For this element type, we strongly recommend defining the number of measurements used to compute the average, as well as the time interval between the measurements. Otherwise, you should define explicitly the means by which this average value is calculated.
Peak	Maximum value among several samples of a base metric. For example, a <i>peak queue depth</i> element would be calculated by comparing several measurements of the queue depth and selecting the maximum. For this data element type, we strongly recommend that you define the number of measurements used to compute the peak value, as well as the time interval between measurements. Otherwise, define explicitly how this peak value is defined. You must also know whether this value is never cleared and thus represents the overall maximum value over all time.



**NOTE:** Each data element type also includes element subsets. For example, the data elements **Counter** and **Gauge** would include subsets for **rate**, **average**, and **peak** measurements.

## Release History Table

Release	Description
17.2R1	Starting with Junos OS Release 17.2R1, QFX10000 and QFX5200 switches are also supported. On QFX5200 switches, only gRPC streaming is supported.
16.1R3	Starting with Junos OS Release 16.1R3, FPC1, FPC2, and dual Routing Engines on PTX Series routers are also supported.
15.1F5	Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.
15.1F3	The Junos Telemetry Interface was introduced in Junos OS Release 15.1F3, on MX Series routers with interfaces configured on MPC1 through MPC6E, and on PTX Series routers with interfaces configured on FPC3.

## Related Documentation

- [Decoding Junos Telemetry Interface Data With UNIX Utilities on page 39](#)



## CHAPTER 3

# Configuring Junos Telemetry Interface (CLI Procedure)

- [Configuring a Junos Telemetry Interface Sensor \(CLI Procedure\) on page 15](#)

## Configuring a Junos Telemetry Interface Sensor (CLI Procedure)

---

Junos Telemetry Interface provides for the highly scalable streaming of telemetry information. Unlike previous monitoring systems, such as SNMP, which use the so-called pull model, the Junos Telemetry Interface uses the push model to collect data. The push model overcomes earlier scaling limits and reduces the processing required by the management station. You can enable monitoring and streaming of data for various system resources, such as physical and logical interfaces and firewall filters. To monitor a specific system resource, you configure a sensor. Each sensor configuration requires three main components:

- Sensor profile—Enables the system resource to monitor and allows you to set related parameters, such as the destination server to send data.
- Export profile—Specifies the attributes for the process of exporting collected data, such as the transport protocol to use and the interval at which to collect data.
- Streaming server profile—Specifies the server for collecting data and related parameters, including the destination IP address and port number.



**NOTE:** Junos Telemetry Interface was introduced in Junos OS Release 15.1F3 on MX Series routers with interfaces configured on MPC1 through MPC6E and on PTX Series routers with interfaces configured on FPC3. Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.

Starting with Junos OS Release 16.1R3, FPC1 and FPC2 on PTX Series routers are also supported.

Starting with Junos OS Release 17.2R1, QFX10000 and PTX1000 switches are also supported.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.



**BEST PRACTICE:** We recommend that you configure at least one export profile and at least one streaming server before you configure a sensor profile. This way you can associate an export profile and a streaming server with the sensor profile configuration.

Before you begin:

- Configure a connection from your Juniper Networks device to a server that is using in-band management interfaces.
- [Configuring an Export Profile on page 16](#)
- [Configuring a Streaming Server Profile on page 18](#)
- [Configuring a Sensor Profile on page 19](#)
- [Verifying Junos Telemetry Interface Sensor Configuration on page 20](#)

## Configuring an Export Profile

An export profile defines the parameters of the export process of data generated through the Junos Telemetry Interface. You must configure at least one export profile, but you can configure multiple export profiles. Each export profile can be associated with multiple sensor profiles. However, you can associate only one export profile with a specific sensor profile.

To configure an export profile:

1. Specify a name for the export profile.

```
[edit services analytics]
user@host# set export-profile name
```

For example, to specify an export-profile name of **export-params**:

```
[edit services analytics]
user@host# set export-profile export-params
```

2. Specify the source IP address of exported packets.

```
[edit services analytics export-profile name]
user@host# set local-address ip-address
```

For example, to specify a source IP address of 192.0.2.3 for an export profile with the name **export-params**:

```
[edit services analytics export-profile export-params]
user@host# set local-address 192.0.2.3
```

3. Specify the source port number of exported packets.

```
[edit services analytics export-profile name]
user@host# set local-port number
```

For example, to specify a source port number of 21111 for an export profile with the name **export-params**:

```
[edit services analytics export-profile export-params]
user@host# set local-port 21111
```

4. Specify the interval, in seconds, at which the sensor generates telemetry data.

```
[edit services analytics export-profile name]
user@host# set reporting-rate seconds
```

For example, to specify an interval of 20 seconds at which any sensor associated with the export-profile with the name **export-params** generates telemetry data :

```
[edit services analytics sensor export-profile export-params]
user@host# set reporting-rate 20
```

5. Specify the format to define the structure of the exported data.



**NOTE:** The only currently supported format is Google protocol buffers (gpb)

```
[edit services analytics export-profile name]
user@host# set format gpb
```

For example, to specify the Google protocol buffers format for exported data for an export-profile with the name **export-params**:

```
[edit services analytics export-profile export-params]
user@host# set format gpb
```

6. Specify the transport protocol to carry the telemetry data in the IP packets.

```
[edit services analytics export-profile name]
user@host# set transport protocol-name
```

For example, to specify the UDP as the transport protocol for telemetry data for an export profile with the name **export-params**:

```
[edit services analytics export-profile export-params]
user@host# set transport udp
```

7. (Optional) Specify the DiffServ code point (DSCP) value to assign to exported packets.



**NOTE:** The default value is 0 (zero).

```
[edit services analytics export-profile name]
user@host# set dscp value
```

For example, to specify a DSCP value of 20 for an export profile with the name **export-params**:

```
[edit services analytics export-profile export-params]
user@host# set dscp 20
```

8. (Optional) Specify a forwarding class to assign to exported packets.



**NOTE:** You can specify a forwarding class only for packets exported by Packet Forwarding Engine sensors. The default value is **best-effort**.

```
[edit services analytics export-profile name]  
user@host# set forwarding-class class-name
```

For example, to specify a forwarding class of **assured-forwarding** for an export-profile with the name **export-params**:

```
[edit services analytics export-profile name]  
user@host# set forwarding-class assured forwarding
```

## Configuring a Streaming Server Profile

A server profile defines the parameters of the server that collects exported telemetry data. You can define more than one server profile. You can also associate the same server profile with more than one sensor profile. Starting in Junos OS Release 15.1F6, you can associate more than one server with a specific sensor.

To define the profile of a streaming server to collect exported telemetry data:

1. Specify the name of the streaming sever.

```
[edit services analytics]  
user@host# set streaming-server server-name
```

For example, to specify a streaming-server name of **telemetry server**:

```
[edit services analytics]  
user@host# set streaming-server telemetry-server
```

2. Specify a destination IP address for the exported packets.

```
[edit services analytics streaming-server server-name]  
user@host# set remote-address ip-address
```

For example, to specify a destination address of 192.0.2.2 for a streaming server with the name **telemetry-server**:

```
[edit services analytics streaming-server telemetry-server]  
user@host# set remote-address 192.0.2.2
```

3. Specify a destination port number for the exported packets.

```
[edit services analytics streaming-server server-name]  
user@host# set remote-port number
```

For example, to specify a destination port number of 30000 for a streaming server with the name **telemetry-server**:

```
[edit services analytics streaming-server telemetry-server]  
user@host# set remote-port 30000
```



## Configuring a Sensor Profile

A sensor profile defines the parameters of the system resource to monitor and stream data. You can enable only one system resource to monitor for each sensor profile. Configure a different sensor profile for each system resource you want to monitor. You can, however, configure more than one sensor to monitor the same system resource. For example, you might want to configure different parameters for exporting data for the same system resource.

To configure a sensor profile:

1. Specify the name of the sensor.

```
[edit services analytics]
user@host# set sensor sensor-name
```

For example, to specify a sensor name of **interface-1**:

```
[edit services analytics]
user@host# set sensor interface-1
```

2. Specify the system resource to monitor and stream data.

```
[edit services analytics sensor sensor-name]
user@host# set resource resource-string-identifier
```

For example, to enable monitoring of logical interfaces for sensor **interface-1**:

```
[edit services analytics sensor interface-1]
user@host# set resource /junos/system/linecard/interface/logical/usage/
```



**NOTE:** You must enter the resource string exactly.

3. (Optional) Specify a regular expression to filter data for the system resource you specified in Step 2. If you do not specify a regular expression, the system resource is monitored globally, that is, systemwide.

```
[edit services analytics sensor sensor-name]
user@host# set resource-filter regular-expression
```

For example, to filter data only for Ethernet logical interfaces for sensor **interface-1**:

```
[edit services analytics sensor interface-1]
user@host# set resource-filter et-*
```

4. Specify the name of a export profile configured at the **[edit export-profile *profile-name*]** hierarchy level to associate with the sensor profile. This export profile defines the parameters for exporting telemetry data.

```
[edit services analytics sensor sensor-name]
user@host# set export-name export-profile-name
```

For example, to associate an export profile named **export-params** with a sensor named **interface-1**:

```
[edit services analytics sensor interface-1]
user@host# set export-name export-params
```

- Specify the name of a streaming server name configured at the `[edit services analytics streaming-server server-name]` hierarchy level to collect exported data.



**NOTE:** Starting in Junos OS Release 15.1F6, you can specify more than one streaming server for a sensor profile. To specify more than one streaming server for a sensor, you must enclose the names in brackets.

```
[edit services analytics sensor sensor-name]
user@host# set streaming-server server-name
```

For example, to associate a streaming server name **telemetry-server** with a sensor named **interface-1**:

```
[edit services analytics sensor interface-1]
user@host# set streaming-server telemetry-server
```

## Verifying Junos Telemetry Interface Sensor Configuration

**Purpose** Confirm your configuration.

**Action** From configuration mode, confirm your configuration by entering the **show services analytics** command. If your output does not display the intended configuration, repeat the instructions in this configuration procedure to correct the configuration.

```
user@host# show services analytics
streaming-server telemetry-server {
  remote-address 192.0.2.2;
  remote-port 30000;
}
export-profile export-params {
  local-address 192.0.2.3;
  local-port 21111;
  dscp 20;
  forwarding-class assured-forwarding;
  reporting-rate 20;
  format gpb;
  transport udp;
}
sensor interface-1 {
  server-name telemetry-server;
  export-name export-params;
  resource /junos/system/linecard/interface/logical/usage/;
  resource-filter et-*;
}
```

After you commit the configuration, verify that the sensor is enabled by issuing the **show agent sensors** operational command.

```
user@host> show agent sensors
```

## Sensor Information :

```

Name                : interface-1
Resource            : /junos/system/linecard/interface/logical/usage/
Version             : 1.0
Sensor-id           : 193570469
Resource-filter     : et-*

```

## Server Information :

```

Name                : telemetry-server
Scope-id            : 0
Remote-Address      : 192.0.2.2
Remote-port         : 30000

```

## Profile Information :

```

Name                : export-params
Rep-interval        : 300
Address             : 192.0.2.3
Port                : 21111
Timestamp           : 1
Format              : GPB
Transport           : UDP
DSCP                : 20
Forwarding-class    : assured-forwarding

```

## Release History Table

Release	Description
17.2R1	Starting with Junos OS Release 17.2R1, QFX10000 and PTX1000 switches are also supported.
16.1R3	Starting with Junos OS Release 16.1R3, FPC1 and FPC2 on PTX Series routers are also supported.
15.1F5	Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.
15.1F3	Junos Telemetry Interface was introduced in Junos OS Release 15.1F3 on MX Series routers with interfaces configured on MPC1 through MPC6E and on PTX Series routers with interfaces configured on FPC3.



## CHAPTER 4

# Junos Telemetry Interface Configuration Statements and Operational Commands

- `export-profile` (Junos Telemetry Interface) on page 24
- `lsp-telemetry` (Junos Telemetry Interface) on page 27
- `sensor` (Junos Telemetry Interface) on page 28
- `sensor-based-stats` (Junos Telemetry Interface) on page 33
- `streaming-server` (Junos Telemetry Interface) on page 34
- `show agent sensors`

## export-profile (Junos Telemetry Interface)

<b>Syntax</b>	<pre>export-profile <i>name</i> {   dscp <i>value</i>;   format <i>file-format</i>;   forwarding-class (assured-forwarding   best-effort   expedited-forwarding       network-control);   local-address <i>ip-address</i>;   local-port <i>source-port-number</i>;   &lt;payload-size <i>bytes</i>&gt;;   reporting-rate <i>seconds</i>;   transport <i>protocol-name</i>; }</pre>
<b>Hierarchy Level</b>	[edit services analytics]
<b>Release Information</b>	<p>Statement introduced in Junos OS Release 15.1F3.</p> <p><b>payload-size bytes</b> option introduced in Junos OS Release 16.1R3.</p> <p>Statement introduced in Junos OS Release 17.2R1 for QFX10000 switches and PTX1000 routers</p>
<b>Description</b>	<p>Configure the parameters of the export process for data generated through Junos Telemetry Interface sensors. You can create one or more export profiles. Each profile can be associated with one or more sensors that define the system resource to monitor and stream data. You can associate only one export profile with a specific sensor configuration.</p> <p>The IP layer delivers the exported data to the remote server. The export profile configuration allows you to specify a format for exported data, a transport protocol, the rate which the system generates data, and the local source port and IP address that are used to define the transport headers in the exported packets.</p> <p>To enable Junos Telemetry Interface, you must also configure a sensor that defines the parameters of the system resource to monitor and stream data, and a server to collect the data. To configure a sensor, include the <b>sensor sensor-name</b> statement at the [edit services analytics] hierarchy level. To configure the server that functions as a data collector, include <b>streaming-server server-name</b> statement at the [edit services analytics] hierarchy level.</p>



**NOTE:** Junos Telemetry Interface was introduced in Junos OS Release 15.1F3 on MX Series routers with interfaces configured on MPC1 through MPC6E and on PTX Series routers with interfaces configured on FPC3. Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.

Starting with Junos OS Release 16.1R3, FPC1 and FPC2 on PTX Series routers are also supported.

Starting with Junos OS Release 17.2R1, QFX10000 switches and PTX1000 routers are also supported.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

---

**Options**    *name*—Name of export profile.



**NOTE:** To associate this export profile with a configured sensor, include the name you configure for the export-profile statement at the [edit services analytics sensor *sensor-name* export-name] hierarchy level.

**dscp value**—Specify the DSCP value for the exported packets.

**Range:** 0 through 63.

**Default:** 0

**format gpb**—Specify the format to define the structure of exported data.

**gpb**—Google protocol buffers format.

**forwarding-class (assured-forwarding | best-effort | expedited-forwarding | network-control)**—(Packet Forwarding Engine sensors only) Specify the forwarding class for exported packets.

**Default:** best-effort

**local-address ip-address**—Specify the source address of exported packets.

**local-port number**—Specify the source port for the exported packets.

**payload-size bytes (Optional)** —Specify the maximum size of exported packets.



**NOTE:**

The payload-size option is supported only on the following sensors:

- /junos/system/linecard/interface/
- /junos/system/linecard/interface/logical/usage/
- /junos/system/linecard/firewall/

**Default:** 5000 bytes.

**Range:** 2000 through 9192 bytes.



**NOTE:** Junos Telemetry Interface does not export packets larger than 9192 bytes.

**reporting-rate seconds**—Specify the interval at which the Junos Telemetry Interface sensor generates data to export to the collector.

As the configured interval expires, the most recent sample collected by the sensor is gathered and forwarded to the server configured to collect data.





**NOTE:** For Packet Forwarding Engine sensors, the minimum reporting rate is 2 seconds.

**Range:** 1 through 3600 (1 hour)

**transport *protocol-name***—Specify the transport protocol to use to carry the telemetry data in the IP packets.

**udp**—User Datagram Protocol.

**Required Privilege Level** interface—To view this statement in the configuration.  
interface-control—To add this statement to the configuration.

**Related Documentation**

- [sensor on page 28](#)

## lsp-telemetry (Junos Telemetry Interface)

**Syntax** lsp-telemetry;

**Hierarchy Level** [edit routing-options]

**Release Information** Statement introduced in Junos OS Release 17.2R1 for MX Series routers, PTX3000 and PTX5000 routers, and QFX10000 switches.

**Description** Enable the collection of statistics LSP events and properties through the Junos Telemetry Interface. You must configure this statement when you configure a sensor to monitor and stream telemetry data for LSP events and properties. Only gRPC is supported for streaming data for LSP events and properties. For more information about provisioning data streaming of LSP events through gRPC, see "[Guidelines for gRPC Sensors \(Junos Telemetry Interface\)](#)" on page 59. The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

**Required Privilege Level** routing—To view this statement in the configuration.  
routing-control—To add this statement to the configuration.

**Related Documentation**

- [Overview of the Junos Telemetry Interface on page 3](#)
- [Understanding OpenConfig and gRPC on Junos Telemetry Interface on page 51](#)

## sensor (Junos Telemetry Interface)

**Syntax**    `sensor sensor-name {  
                   export-name export-profile-name;  
                   resource resource-string;  
                   <resource-filter regular expression>;  
                   server-name [ streaming-server-names ];  
                   }`

**Hierarchy Level**    `[edit services analytics]`

**Release Information**    Statement introduced in Junos OS Release 15.1F3.  
 Support for MPC7E, MPC8E, and MPC9E on MX Series routers added in Junos OS Release 15.1F5.  
 Support for FPC1 and FPC2 on PTX Series routers added in Junos OS Release 16.1R3.  
 Statement introduced in Junos OS Release 17.2R1 for QFX10000 switches and PTX1000 routers..

**Description**    Configure a Junos Telemetry Interface sensor, which defines the parameters of a system resource to monitor and stream data. You can configure more than one sensor to stream data for the same system resource. For example, you might want to configure different parameters for exporting data for the same system resource. Additionally, you can use regular expressions to filter the data collected. Examples include filters for logical and physical interfaces and LSP messages. To apply different filters to the same system resource, you configure multiple sensors. For example, you can configure multiple logical interface sensors and apply a different interface filter to each one.



**NOTE:** The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

**Options**    Each sensor configuration requires you to specify the following: sensor name, an export profile name, a resource identifier string that enables monitoring and streaming of data for the specified system resource, and a server name to collect data. A regular expression to filter data for the specified resource is optional.

**sensor-name**—Specify a name that defines the sensor configuration. For example, for a sensor configuration that monitors all LSP events, you might choose the name **lsp-mon-global**. For a sensor configuration that monitors events only for an LSP named A2B, you might choose the name **lsp-mon-A2B**.

**export-name export-profile-name**—Specify the name of an export profile that you configured at the `[edit services analytics export-profile name]` hierarchy level to associate with the sensor. This export profile defines the parameters for exporting telemetry data, such as a format for exported data and the rate at which data is generated for export.



**NOTE:** You can apply only one export profile to each sensor configuration.

The only supported transport protocol when you configure a sensor through the CLI is UDP.

**resource *resource-string***—Enable the system resource to monitor and stream data. Each string corresponds to a specific system resource. The format is a file path and must be entered exactly. You can associate only one ***resource-string*** with a ***sensor-name***. Configure a separate sensor for each system resource you want to monitor. The resource string to enable LSP monitoring can be modified to specify a specific LSP.



**NOTE:** You can configure more than one sensor to monitor the same system resource. Configuring different sensors for the same system resource allows you configure different parameters for monitoring that resource.

Table 4 on page 30 lists each supported ***resource-identifier-string***, a description of the system resource monitored, and additional configuration information.

Table 4: resource statement Options

resource string	Description	Release Information
/junos/services/label-switched-path/usage/	<p>Packet Forwarding Engine sensor for LSP statistics. Only ingress LSPs are supported. Bypass LSPs, including those configured as ingress LSPs, are not supported. On MX Series routers only, bidirectional LSPs for ultimate-hop popping (UHP) are also supported.</p> <p><b>NOTE:</b> You can modify <code>/junos/services/label-switched-path/usage/</code> to specify a specific LSP. Add <code>__instance__/<i>lsp-name</i></code> to the end of the resource string identifier. For example, to monitor and stream data for LSP statistics for an LSP named <code>mirror-to-murano-1</code>, enter the following:  <code>/junos/services/label-switched-path/usage/__instance__/mirror-to-murano-1</code>. If you do not specify a specific LSP name, the system resource monitors and streams data for all LSPs.</p> <p>When you enable a sensor for LSP statistics, you must also configure the <code>sensor-based-stats</code> statement at the <code>[edit protocols mpls]</code> hierarchy level. On MX Series routers must also operate in enhanced mode. If not enabled by default, configure either the <code>enhanced-ip</code> statement or the <code>enhanced-ethernet</code> statement at the <code>[edit chassis network-services]</code> hierarchy level.</p>	<p>Junos OS Release 15.1F6 and later.</p> <p>Junos OS Release 17.2R1 and later on QFX10000 switches and PTX1000 routers.</p>
/junos/system/linecard/cpu/memory/	<p>Packet Forwarding Engine sensor for CPU memory.</p>	<p>Junos OS Release 16.1R3 and later.</p> <p>Junos OS Release 17.2R1 and later on QFX10000 switches and PTX1000 routers.</p>
/junos/system/linecard/firewall/	<p>Packet Forwarding Engine sensor for firewall filter counters and policer counters. Each line card reports counters separately.</p> <p><b>NOTE:</b> Hierarchical policer statistics are collected for MX Series routers only. Traffic-class counter statistics are collected for PTX Series routers and QFX10000 switches only.</p> <p>Firewall counters are exported even if the interface to which the firewall filter is attached is down.</p>	<p>Junos OS Release 15.1F5 and later.</p> <p>Junos OS Release 17.2R1 and later on QFX10000 switches.</p>
/junos/system/linecard/interface/	<p>Packet Forwarding Engine sensor for physical interface traffic.</p> <p><b>NOTE:</b> For PTX Series routers, for a specific interface, queue statistics are exported for each line card. For MX series routers, interface queue statistics are exported only from the slot on which an interface is configured.</p> <p>For Aggregated Ethernet interfaces, statistics are exported for the member physical interfaces. You must aggregate the counters at the destination server, or</p>	<p>Junos OS Release 15.1F3 and later on PTX Series routers only. Support introduced for MX Series routers in Junos OS Release 15.1F5.</p> <p>Junos OS Release 17.2R1 and later on QFX10000 switches</p>

Table 4: resource statement Options (*continued*)

resource string	Description	Release Information
	<p>collector.</p> <p>If a physical interface is administratively down or operationally down, interface counters are not exported.</p> <p>Issuing an operational <b>clear</b> command, such as <b>clear interfaces statistics all</b>, does not reset statistics exported by the line card.</p>	and PTX1000 routers.
/junos/system/linecard/interface/logical/usage/	<p>Packet Forwarding Engine sensor for logical interface traffic.</p> <p><b>NOTE:</b> If a logical interface is operationally down, interface statistics continue to be exported.</p> <p>Issuing an operational <b>clear</b> command, such as <b>clear interfaces statistics all</b>, does not reset statistics exported by the line card.</p>	<p>Junos OS Release 15.1F5 and later.</p> <p>Junos OS Release 17.2R1 and later on QFX10000 switches.</p>
/junos/system/linecard/npu/memory/	Packet Forwarding Engine sensor for network processing unit (NPU) memory.	<p>Junos OS Release 16.1R3 and later.</p> <p>Junos OS Release 17.2R1 and later on QFX10000 switches.</p>
/junos/system/linecard/npu/utilization/	Packet Forwarding Engine sensor for NPU memory utilization and total memory available for each memory type.	<p>Junos OS Release 16.1R3 and later.</p> <p>Junos OS Release 17.2R1 and later on QFX10000 switches.</p>
/junos/npu-memory/	<p>Sensor that exports both NPU memory statistics from the Packet Forwarding Engine and flow-label statistics from the Routing Engine.</p> <p>To export only flow-label statistics, include the <b>/junos/npu-memory/flabel-memory/</b> resource string.</p> <p>To export only Packet Forwarding Engine NPU memory statistics, include <b>/junos/system/linecard/npu/memory/</b> resource string.</p>	<p>Junos OS Release 16.1R3 and later on PTX Series routers only.</p> <p><b>NOTE:</b> Junos OS Release 17.2R1 and later on PTX1000 routers.</p>
/junos/system/linecard/services/inline-jflow/	Packet Forwarding Engine sensor for performance metrics of the inline flow sampling process, such as the number of active flows and the number of exported flows.	Junos OS Release 16.1R3 and later on MX series and PTX series routers only.
/junos/system/linecard/optics/	Packet Forwarding Engine sensor for various optical performance metrics, such as transmit and receive power levels.	<p>junos OS Release 16.1R3 and later.</p> <p>Junos OS Release and later 17.2R1 on QFX10000 switches.</p>

Table 4: resource statement Options (*continued*)

resource string	Description	Release Information
/junos/system/linecard/qmon/	<p>Sensor for queue depth statistics for ingress and egress queue traffic. Statistics are exported directly from the line card.</p> <p><b>NOTE:</b> Issuing an operational <b>clear</b> command, such as <b>clear interfaces statistics all</b>, does not reset the statistics exported by the line card.</p>	Junos OS Release 17.1R1 and later on MX Series routers on MPC7E, MPC8E, and MPC9E only.
/junos/system/linecard/fabric/	<p>Sensor for fabric statistics</p> <p>The following types of statistics can be exported:</p> <ul style="list-style-type: none"> <li>Fabric statistics for Packet Forwarding Engine pairs (<b>resource-filter</b> option is not supported)</li> <li>FPC fabric statistics</li> <li>Control Board and Switch Fabric Board fabric statistics.</li> </ul>	Junos OS Release 17.2R1 and later on MX Series routers only.

**resource-filter *regular-expression***—(Optional) Specify a regular expression to filter data for a specific resource. For example, you can filter for a specific set of logical or physical interfaces, firewall filters, or LSP messages. When you configure a system resource to monitor and stream data globally—that is, systemwide—you do not need to include a regular expression.

Examples of regular expressions to filter data exported through sensor configuration:

- Logical interface statistics sensor—et-2/0/7:1\*
- LSP events sensor—lsp-from-A-to-B\*
- Firewall filter counters sensor—f\_testl\*

**server-name [ *streaming- server-names* ]**—Specify one or more servers to transport data for collection. Include at least one server-name configured at the **[edit services analytics *streaming-server* server-name]** hierarchy level.



**NOTE:** Starting in Junos OS Release 15.1F6, you can configure as many as four streaming servers for a single sensor configuration. In previous releases, you can specify only one streaming server for each configured sensor. To specify more than one streaming server for a sensor, you must enclose the names in brackets.


**Required Privilege Level**

interface—To view this statement in the configuration.

interface-control—To add this statement to the configuration.

**Related Documentation** • [export-profile on page 24](#)

## sensor-based-stats (Junos Telemetry Interface)

<b>Syntax</b>	sensor-based-stats;
<b>Hierarchy Level</b>	[edit protocols mpls]
<b>Release Information</b>	Statement introduced in Junos OS Release 15.1F6. Support added for QFX10000 switches and PTX1000 routers starting with Junos OS Release 17.2R1.
<b>Description</b>	<p>Enable the collection of LSP statistics for Junos Telemetry Interface. You must configure this statement when you configure a sensor to monitor and stream data for LSP statistics. To enable a sensor to stream data for LSP statistics through UDP, include the <b>resource /junos/services/label-switched-path/usage/</b> statement at the [edit services analytics <b>sensor</b> <i>sensor-name</i>] hierarchy level.</p> <p>For additional information about configuring an LSP statistics sensor to stream data through gRPC, see “<a href="#">Guidelines for gRPC Sensors (Junos Telemetry Interface)</a>” on page 59.</p>
<div>  <b>NOTE:</b> On MX Series routers, this statement is supported only on MPCs.         </div>	
<b>Options</b>	None
<b>Required Privilege Level</b>	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.
<b>Related Documentation</b>	• <a href="#">Understanding the Junos Telemetry Interface Export Format of Collected Data on page 10</a>

## streaming-server (Junos Telemetry Interface)

---

**Syntax**    `streaming-server streaming-server-name {  
                  remote-address ip-address;  
                  remote-port number;  
                  }`

**Hierarchy Level**    [edit services analytics]

**Release Information**    Statement introduced in Junos OS Release 15.1F3.  
Statement introduced in Junos OS Release 17.2R1 for QFX10000 switches and PTX1000 routers.

**Description**    For Junos Telemetry Interface, configure the parameters of the server that collects exported data streamed by a monitored system resource. You can configure more than one streaming server. To collect data, you must associate a configured server with one or more configured sensors. The sensor configuration defines the parameters to monitor a specific system resource. To configure a sensor, include the `sensor sensor-name` statement at the [edit services analytics] hierarchy level.

To configure the server that collects data, you must also configure a destination IP address and a destination port. Junos Telemetry Interface relies on neighbor reachability information to deliver packets to the destination address. That means that all policies, such as filtering, that apply to the packets for that destination also apply to the exported packets.



**NOTE:** Starting with Junos OS Release 15.1F6, you can also associate more than one server with a specific sensor configuration, which enables you to transmit streamed data for the same sensor to more than one server.

---



**NOTE:** Junos Telemetry Interface was introduced in Junos OS Release 15.1F3 on MX Series routers with interfaces configured on MPC1 through MPC6E and on PTX Series routers with interfaces configured on FPC3. Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.

Starting with Junos OS Release 16.1R3, FPC1 and FPC2 on PTX Series routers are also supported.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

---

**Options**    `streaming-server-name`—Specify a name for the server configured to collect data streamed through Junos Telemetry Interface. You can configure multiple streaming servers.



To associate as many as four server names with a sensor configuration, include each name at the **[edit services analytics sensor *sensor-name* streaming server [ *streaming-server-names* ] ]** hierarchy level. If you specify more than one streaming server, you must enclose the names in brackets.

**remote-address *ip-address***—Specify the destination address of the streaming server for exported packets.

**remote-port *number***—Specify a port number for the destination address of the streaming server for exported packets.

<b>Required Privilege Level</b>	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
---------------------------------	---

<b>Related Documentation</b>	<ul style="list-style-type: none"><li>• <a href="#">export-profile (Junos Telemetry Interface) on page 24</a></li></ul>
------------------------------	---

## show agent sensors

**Syntax** `show agent sensors`

**Release Information** Statement introduced in Junos OS Release 15.1F3  
Support added for QFX10000 switches, QFX5200 switches, and PTX1000 routers in Junos OS Release 17.2R1.

**Description** Display information about sensors configured for Junos Telemetry Interface.



**NOTE:** Junos Telemetry Interface was introduced in Junos OS Release 15.1F3 on MX Series routers with interfaces configured on MPC1 through MPC6E and on PTX Series routers with interfaces configured on FPC3. Starting in Junos OS Release 15.1F5, Junos Telemetry Interface is also supported on MPC7E, MPC8E, and MPC9E on MX Series routers.

Starting with Junos OS Release 16.1R3, FPC1, FPC2, and dual Routing Engines on PTX Series routers are also supported.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

**Required Privilege Level** view

**Related Documentation**

- [export-profile on page 24](#)
- [sensor on page 28](#)
- [streaming-server on page 34](#)

**List of Sample Output** [show agent sensors \(physical interface sensor\) on page 37](#)  
[show agent sensors \(firewall filter sensor\) on page 38](#)

**Output Fields** [Table 5 on page 36](#) lists the output fields for the **show agent sensors** command. Output fields are listed in the approximate order in which they appear.

**Table 5: show agent sensors Output Fields**

Field Name	Field Description
<b>Sensor Information</b>	Information about sensors configured to monitor system resources and stream data.
<b>Name</b>	Name of configured sensor.
<b>Resource</b>	Resource string used to configure and identify the system resource enabled to monitor and stream data..

Table 5: show agent sensors Output Fields (*continued*)

Field Name	Field Description
<b>Sensor-id</b>	Numerical identifier of the sensor.
<b>Server Information</b>	Information about servers configured to collect sensor data.
<b>Name</b>	Name of server.
<b>Scope-id</b>	
<b>Remote-Address</b>	Destination IP address for exported packets.
<b>Remote-port</b>	Destination port for exported packets.
<b>Profile information</b>	Information about export profiles for sensors.
<b>Name</b>	Name of export profile.
<b>Rep-interval</b>	Interval, in seconds, at which the sensor generates data to export.
<b>Address</b>	Source address of exported packets.
<b>Port</b>	Source port of exported packets.
<b>Format</b>	Format of exported data message: <b>GPB</b>
<b>DSCP</b>	Configured DSCP value for exported packets.
<b>Forwarding-class</b>	Configured forwarding class for exported packets.

## Sample Output

### show agent sensors (physical interface sensor)

```

user@host> show agent sensors
Sensor Information :
  Name                : S1
  Resource             : /junos/system/linecard/interface/

  Sensor-id           : 2641

  Server Information :

    Name              : JV5
    Scope-id          : 0
    Remote-Address     : 180.1.1.1
    Remote-port        : 3008

  Profile Information :

    Name              : EXP5

```

Rep-interval	: 2
Address	: 180.1.1.2
Port	: 30010
Timestamp	: 1
Format	: GPB
Transport	: UDP
DSCP	: 255
Forwarding-class	: 255

#### show agent sensors (firewall filter sensor)

```
user@host> show agent sensors
```

Sensor Information :

Name	: firewall-stats
Resource	:/junos/system/linecard/firewall/
Sensor ID	: 93390914

Server Information :

Name	: jvision-server
Scope ID	: 0
Remote-Address	: 160.1.1.1
Remote-port	: 2001

Profile Information :

Name	: export-common
Rep-interval	: 2
Address	: 160.1.1.2
Port	: 1000
Timestamp	: 1
Format	: GPB
Transport	: UDP
DSCP	: 255
Forwarding-class	: 255

## CHAPTER 5

# Decoding Data

- [Decoding Junos Telemetry Interface Data With UNIX Utilities on page 39](#)

### Decoding Junos Telemetry Interface Data With UNIX Utilities

---

You can use UNIX utilities to decode Junos Telemetry Interface data on a server, or collector, that is streaming data from a Juniper Networks device. The example in this section shows you how to decode a single packet of streamed data.

#### Preparing the Collector to Decode Data

This example requires the following:

- UNIX OS with the Netcat (nc) utility.
- Protocol buffers compiler.
- Junos Telemetry Interface protocol buffers files.

This procedure shows how to prepare the collector to decode data using the Ubuntu OS.

1. Install the Netcat utility.

```
sudo apt-get install netcat
```

2. Install the protocol buffers compiler.

```
sudo apt-get install protobuf-compiler
```

3. Install the protocol buffers developer's library.

```
sudo apt-get install libprotobuf-dev
```

4. Verify that the library files are installed.

```
ls /usr/include/google/protobuf/descriptor.proto  
/usr/include/google/protobuf/descriptor.proto
```

5. Download and install the latest version of the Junos Telemetry interface protocol buffers files.

From a Web browser, navigate to the All Junos Platforms software download URL on the Juniper Networks page: <http://www.juniper.net/support/downloads/>. After you select the name of the Junos OS platform and the release number, go to the **Tools** section and download the **Junos Telemetry Interface Data Model Files** package.

```
tar -xvzf junos-telemetry-interface-15.1F6.9.tgz
junos-telemetry-interface/telemetry_top.proto
junos-telemetry-interface/logical_port.proto
junos-telemetry-interface/lsp_mon.proto
junos-telemetry-interface/firewall.proto
junos-telemetry-interface/lsp_stats.proto
junos-telemetry-interface/port.proto
junos-telemetry-interface/NOTICE
junos-telemetry-interface/license.txt
```



**NOTE:** Be sure to note the location of the extracted files.

## Decoding Data on the Collector

This procedure shows you how to capture data, decode raw data, and use the protocol buffers files to decode data.

To decode data:

1. Capture the data.

Run netcat on a destination streaming telemetry server, or collector, in UDP listener mode to store all incoming datagrams into a file. Use the destination port number configured in streaming-server profile on your Juniper Networks device.

```
nc -ul 0.0.0.0 20000 > data.gpb
```



**NOTE:** This command stores datagrams into a file named **data.gpb**. Run this program to capture data. When you want to stop receiving data, stop with the program by sending the break signal (Control + C)

2. Decode raw data.



**NOTE:** This step is optional. It is not required if you know the encoded message type of the data.

Decode the message from the **data.gpb** file.

```
protoc --decode_raw < ../dump.gpb
1: "hillrock:160.1.1.25"
2: 0
4:
"S1:/junos/system/linecard/interface/logical/usage:/junos/system/linecard/interface/logical/usage/:PFE"
5: 65265
```

```

6: 1477686534474
7: 1
8: 1
101 {
  2636 {
    7 {
      1 {
        1: "et-0/0/4:2.32767"
        2: 1477642750
        3: 813
        4 {
          12: 0x37363732332e3165
        }
      }
    }
  }
}
.
.
.

```

The next nested structure under **2636** identifies the sensor type. The numerical value **2636** identifies the **JuniperNetworksSensor** message, which is defined in the **telemetry\_top.proto** file. In this example, the numerical identifier **7** corresponds to the **LogicalPort** message defined in the **logical\_port.proto** file. Use this information in the next step to generate more detailed output.

### 3. Decode the message to include field names.

Run the protocol buffers compiler with the decode option. Additionally, specify the top-level message type (**TelemetryStream**) and the file with the message definition, **logical\_port.proto**. You must also include the Goggle protocol buffers (gpb) library.

```

protoc --decode TelemetryStream logical_port.proto -I /usr/include -I . <
dump.gpb
system_id: "hillrock:160.1.1.25"
component_id: 0
sensor_name:
"SL:/junos/system/linecard/interface/logical/usage:/junos/system/linecard/interface/logical/usage:/PFE"
sequence_number: 65268
timestamp: 1477686536484
version_major: 1
version_minor: 1
enterprise {
  [juniperNetworks] {
    [jnprLogicalInterfaceExt] {
      interface_info {
        if_name: "et-0/0/4:2.32767"
        init_time: 1477642750
        snmp_if_index: 813
        parent_ae_name: "ae1.32767"
        ingress_stats {
          if_packets: 0
          if_octets: 0
        }
        egress_stats {
          if_packets: 0
          if_octets: 0
        }
        op_state {
          operational_status: "up"
        }
      }
    }
  }
}

```

```
    }  
  }  
  interface_info {  
    if_name: "et-0/0/7:3.0"  
    init_time: 1477642750  
    snmp_if_index: 520  
    parent_ae_name: "ae0.0"  
    ingress_stats {  
      if_packets: 61203309  
      if_octets: 6487548454  
    }  
    egress_stats {  
      if_packets: 87416547  
      if_octets: 9266153982  
    }  
    op_state {  
      operational_status: "up"  
    }  
  }  
  interface_info {  
    if_name: "et-0/0/13:0.0"  
    init_time: 1477642750  
    snmp_if_index: 2512  
    ingress_stats {  
      if_packets: 26266247  
      if_octets: 2784214806  
    }  
    egress_stats {  
      if_packets: 26247215  
      if_octets: 2781829290  
    }  
    op_state {  
      operational_status: "up"  
    }  
  }  
  interface_info {  
    if_name: "et-0/0/13:0.1"  
    init_time: 1477642750  
    snmp_if_index: 2522  
    ingress_stats {  
      if_packets: 26266249  
      if_octets: 2784214972  
    }  
    egress_stats {  
      if_packets: 26249115  
      if_octets: 2781935590  
    }  
    op_state {  
      operational_status: "up"  
    }  
  }  
  interface_info {  
    if_name: "et-0/0/13:0.2"  
    init_time: 1477642750  
    snmp_if_index: 2523  
    ingress_stats {  
      if_packets: 26266248  
      if_octets: 2784214912  
    }  
    egress_stats {  
      if_packets: 26249106
```



```
        if_octets: 2781935086
      }
      op_state {
        operational_status: "up"
      }
    }
  interface_info {
    if_name: "et-0/0/13:0.3"
    init_time: 1477642750
    snmp_if_index: 2524
    ingress_stats {
      if_packets: 26266248
      if_octets: 2784214820
    }
    egress_stats {
      if_packets: 26248520
      if_octets: 2781902320
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.4"
    init_time: 1477642750
    snmp_if_index: 2525
    ingress_stats {
      if_packets: 26266247
      if_octets: 2784214760
    }
    egress_stats {
      if_packets: 26247302
      if_octets: 2781834112
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.5"
    init_time: 1477642750
    snmp_if_index: 2526
    ingress_stats {
      if_packets: 26266247
      if_octets: 2784214760
    }
    egress_stats {
      if_packets: 26247209
      if_octets: 2781828904
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.6"
    init_time: 1477642750
    snmp_if_index: 2527
    ingress_stats {
      if_packets: 26266248
      if_octets: 2784214820
    }
  }
```

```
    }
    egress_stats {
      if_packets: 26247196
      if_octets: 2781828226
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.7"
    init_time: 1477642750
    snmp_if_index: 2528
    ingress_stats {
      if_packets: 26266247
      if_octets: 2784214760
    }
    egress_stats {
      if_packets: 26247203
      if_octets: 2781828618
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.8"
    init_time: 1477642750
    snmp_if_index: 2529
    ingress_stats {
      if_packets: 26266247
      if_octets: 2784214760
    }
    egress_stats {
      if_packets: 26247225
      if_octets: 2781829850
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.9"
    init_time: 1477642750
    snmp_if_index: 2530
    ingress_stats {
      if_packets: 26266247
      if_octets: 2784214760
    }
    egress_stats {
      if_packets: 26247209
      if_octets: 2781828954
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.32767"
    init_time: 1477642750
    snmp_if_index: 648
```

```
    ingress_stats {
      if_packets: 4
      if_octets: 240
    }
    egress_stats {
      if_packets: 0
      if_octets: 0
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/4:2.32767"
    init_time: 1477642750
    snmp_if_index: 813
    parent_ae_name: "ae1.32767"
    ingress_stats {
      if_packets: 0
      if_octets: 0
    }
    egress_stats {
      if_packets: 0
      if_octets: 0
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/7:3.0"
    init_time: 1477642750
    snmp_if_index: 520
    parent_ae_name: "ae0.0"
    ingress_stats {
      if_packets: 61206122
      if_octets: 6487846632
    }
    egress_stats {
      if_packets: 87420567
      if_octets: 9266580102
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.0"
    init_time: 1477642750
    snmp_if_index: 2512
    ingress_stats {
      if_packets: 26267458
      if_octets: 2784343172
    }
    egress_stats {
      if_packets: 26248420
      if_octets: 2781957020
    }
    op_state {
      operational_status: "up"
    }
  }
```

```
}
interface_info {
  if_name: "et-0/0/13:0.1"
  init_time: 1477642750
  snmp_if_index: 2522
  ingress_stats {
    if_packets: 26267460
    if_octets: 2784343338
  }
  egress_stats {
    if_packets: 26250320
    if_octets: 2782063320
  }
  op_state {
    operational_status: "up"
  }
}
interface_info {
  if_name: "et-0/0/13:0.2"
  init_time: 1477642750
  snmp_if_index: 2523
  ingress_stats {
    if_packets: 26267459
    if_octets: 2784343278
  }
  egress_stats {
    if_packets: 26250311
    if_octets: 2782062816
  }
  op_state {
    operational_status: "up"
  }
}
interface_info {
  if_name: "et-0/0/13:0.3"
  init_time: 1477642750
  snmp_if_index: 2524
  ingress_stats {
    if_packets: 26267460
    if_octets: 2784343292
  }
  egress_stats {
    if_packets: 26249725
    if_octets: 2782030050
  }
  op_state {
    operational_status: "up"
  }
}
interface_info {
  if_name: "et-0/0/13:0.4"
  init_time: 1477642750
  snmp_if_index: 2525
  ingress_stats {
    if_packets: 26267459
    if_octets: 2784343232
  }
  egress_stats {
    if_packets: 26248507
    if_octets: 2781961842
  }
}
```

```

    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.5"
    init_time: 1477642750
    snmp_if_index: 2526
    ingress_stats {
      if_packets: 26267459
      if_octets: 2784343232
    }
    egress_stats {
      if_packets: 26248414
      if_octets: 2781956634
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.6"
    init_time: 1477642750
    snmp_if_index: 2527
    ingress_stats {
      if_packets: 26267460
      if_octets: 2784343292
    }
    egress_stats {
      if_packets: 26248401
      if_octets: 2781955956
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.7"
    init_time: 1477642750
    snmp_if_index: 2528
    ingress_stats {
      if_packets: 26267459
      if_octets: 2784343232
    }
    egress_stats {
      if_packets: 26248408
      if_octets: 2781956348
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.8"
    init_time: 1477642750
    snmp_if_index: 2529
    ingress_stats {
      if_packets: 26267459
      if_octets: 2784343232
    }
    egress_stats {

```

```
        if_packets: 26248430
        if_octets: 2781957580
      }
      op_state {
        operational_status: "up"
      }
    }
  interface_info {
    if_name: "et-0/0/13:0.9"
    init_time: 1477642750
    snmp_if_index: 2530
    ingress_stats {
      if_packets: 26267459
      if_octets: 2784343232
    }
    egress_stats {
      if_packets: 26248414
      if_octets: 2781956684
    }
    op_state {
      operational_status: "up"
    }
  }
  interface_info {
    if_name: "et-0/0/13:0.32767"
    init_time: 1477642750
    snmp_if_index: 648
    ingress_stats {
      if_packets: 4
      if_octets: 240
    }
    egress_stats {
      if_packets: 0
      if_octets: 0
    }
    op_state {
      operational_status: "up"
    }
  }
}
}
```

**Related Documentation**

- [Configuring a Junos Telemetry Interface Sensor \(CLI Procedure\) on page 15](#)

## PART 3

# Configuring gRPC Sensors

- [OpenConfig and gRPC for Junos Telemetry Interface on page 51](#)





## CHAPTER 6

# OpenConfig and gRPC for Junos Telemetry Interface

- [Understanding OpenConfig and gRPC on Junos Telemetry Interface on page 51](#)
- [Installing the Network Agent Package \(Junos Telemetry Interface\) on page 54](#)
- [Configuring gRPC for the Junos Telemetry Interface on page 57](#)
- [Guidelines for gRPC Sensors \(Junos Telemetry Interface\) on page 59](#)

## Understanding OpenConfig and gRPC on Junos Telemetry Interface

---

Starting in Junos OS Release 16.1R3, you can use a set of remote procedure call (RPC) interfaces to configure the Junos Telemetry Interface and stream telemetry data using the gRPC framework. OpenConfig supports the use of vendor-neutral data models for configuring and managing multivendor networks. gRPC is an open source framework that provides secure and reliable transport of data.



**NOTE:** OpenConfig for Junos OS and gRPC are supported only on MPCs on MX Series and on PTX Series routers starting with Junos OS Release 16.1R3.

Starting with Junos OS Release 17.2R1, OpenConfig and gRPC are also supported on QFX10000 switches, QFX5200 switches, and PTX1000 routers.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

- 
- [Network Agent Software on page 51](#)
  - [Using OpenConfig for Junos OS to Enable Junos Telemetry Interface on page 52](#)
  - [Using gRPC to Stream Data on page 52](#)

## Network Agent Software

Implementing OpenConfig with gRPC for Junos Telemetry Interface requires that you download and install a package called Network Agent if your Juniper Networks device is running a version of Junos OS with Upgraded FreeBSD. For all other versions of Junos OS, the Network Agent functionality is embedded in the software. Network Agent functions as a gRPC server and terminates the OpenConfig RPC interfaces. It is also responsible for streaming the telemetry data according to the OpenConfig specification. To view the

OpenConfig specification for telemetry, see the [OpenConfig Telemetry specification](#). For more information about OpenConfig for Junos OS, see the *OpenConfig Feature Guide*.

The Network Agent component also supports server-based Secure Sockets Layer (SSL) authentication. Client-based SSL authentication is not supported. You must install SSL certificates on your Juniper Networks device.

For information about installing the Network Agent package, see “[Installing the Network Agent Package](#)” on page 54.

## Using OpenConfig for Junos OS to Enable Junos Telemetry Interface

OpenConfig for Junos OS specifies an RPC model to enable the Junos Telemetry Interface. You must download and install the OpenConfig for Junos OS package and YANG Models on your Juniper Networks device. Using a Web browser, navigate to the All Junos Platforms software download URL on the Juniper Networks webpage:

<http://www.juniper.net/support/downloads/>. From the **Network Management** tab, scroll down to select **OpenConfig**. Select the **Software** tab. Select the appropriate version of OpenConfig module. Two versions are available, one for devices running Junos OS with Upgraded FreeBSD and another for devices running all other versions of Junos OS. Select **YANG Models**. For more information see *Installing the OpenConfig Package*.

The programmatic interface **OpenConfigTelemetry** that is installed by the Network Agent package defines the telemetry gRPC service. The **telemetrySubscribe** RPC specifies the following subscription parameters:

- OpenConfig path that identifies the system resource to stream telemetry data, for example:  
`/interfaces/interface/state/counters/`
- Interval at which data is reported and streamed to the collector server, in milliseconds, for example:  
`sample_frequency = 4000`

The **telemetrySubscribe** RPC is used by a streaming server, or collector, to request an inline subscription from for data at the specified path. The device should then send telemetry data back on the same connection as the subscription request.

## Using gRPC to Stream Data

Per the OpenConfig specification, only gRPC-based transport is supported for streaming data. The gRPC server that is installed by the Network Agent package terminates the gRPC sessions from the management system that runs the client. RPC calls trigger the creation of Junos OS sensors that either stream data periodically or report events, which are then funneled onto the appropriate gRPC channel by Network Agent.

See [Table 6 on page 53](#) for a list and descriptions of the RPCs implemented to the support the Junos Telemetry Interface.

Table 6: Telemetry RPCs

RPC Name	Description
<b>telemetrySubscribe</b>	Specify telemetry parameters and stream data for the specified list of OpenConfig paths.
<b>getTelemetrySubscriptions</b>	Retrieve the list of subscriptions that are created through <b>telemetrySubscribe</b> .
<b>cancelSubscription</b>	Unsubscribe a subscription created through <b>telemetrySubscribe</b> .

Data streamed through gRPC is formatted in OpenConfig key/value pairs in Google protocol buffers (gpb) messages. In this universal format, keys are strings that correspond to the path of the system resources in the OpenConfig schema for the device being monitored. The values correspond to integers or strings that identify the operational state of the system resource, such as statistics counters and the state of a resource.

The following shows the universal key/value format:

```

message KeyValue {
    string key          = 1 [(telemetry_options).is_key = true];
    uint64 int_value    = 2;
    string str_value    = 3;
    string prefix_str   = 4;
}

message TelemetryStream {
    // router name or export IP address
    required string system_id = 1 [(telemetry_options).is_key = true];

    // line card / RE (slot number)
    optional uint32 component_id = 2 [(telemetry_options).is_key = true];

    // PFE (if applicable)
    optional uint32 sub_component_id = 3 [(telemetry_options).is_key = true];

    // timestamp (common to all entries in the kv array)
    optional uint64 timestamp = 4 [(telemetry_options).is_timestamp = true];

    // key / value pairs
    repeated KeyValue kv;
}

```

The following example shows how a set of counters for an interface can be represented:

```

key = "/interfaces/counters/rx-bytes",    int_value = 1000
key = "/interfaces/counters/tx-bytes",    int_value = 2000
key = "/interfaces/counters/rx-packets",  int_value = 10
key = "/interfaces/counters/rx-bytes",    int_value = 20
key = "/interfaces/counters/oper-state",  str_value = "up"

```

The Network Agent package provides a mapping table that maps field names to the OpenConfig key strings.

Release History Table

Release	Description
17.2R1	Starting with Junos OS Release 17.2R1, OpenConfig and gRPC are also supported on QFX10000 switches, QFX5200 switches, and PTX1000 routers.
16.1R3	Starting in Junos OS Release 16.1R3, you can use a set of remote procedure call (RPC) interfaces to configure the Junos Telemetry Interface and stream telemetry data using the gRPC framework.
16.1R3	OpenConfig for Junos OS and gRPC are supported only on MPCs on MX Series and on PTX Series routers starting with Junos OS Release 16.1R3.

**Related Documentation**

- [Installing the Network Agent Package \(Junos Telemetry Interface\) on page 54](#)
- [Understanding Junos OS with Upgraded FreeBSD](#)

## Installing the Network Agent Package (Junos Telemetry Interface)

Starting with Junos OS Release 16.1R3, the Junos Network Agent software package provides a framework to support OpenConfig and gRPC for the Junos Telemetry Interface on MX Series routers and PTX5000 routers. The Network Agent package functions as a gRPC server that terminates the OpenConfig remote procedure call (RPC) interfaces and streams the telemetry data according to the OpenConfig specification. The Junos Network Agent package, which runs on the Routing Engine, implements local statistics collection and reports data to active telemetry stream subscribers.

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

Starting with Junos OS Release 17.2R1, the Junos Network Agent Package is also supported on QFX10000 switches and QFX5200 switches.

The Junos Network Agent is available as a separate package only for Junos OS with Upgraded FreeBSD. For other versions of Junos OS, Network Agent functionality is embedded in the software. For more information about Junos OS with Upgraded FreeBSD, see *Understanding Junos OS with Upgraded FreeBSD*.

Network Agent for Junos OS software package has the following naming conventions:

- Package Name—This is **Network-Agent**.
- Architecture—This field indicates the CPU architecture of the platforms, such as **x86**.
- Application Binary Interface (ABI)—This field indicates the “word length” of the CPU architecture. Values include **32** for 32-bit architectures and **64** for 64-bit architectures.
- Release—This field indicates the Junos OS release number, such as **16.1R3.16**.
- Package release and spin number—This field indicates the package version and spin number, such as **C1.1**.

All Junos Network Agent packages are in tarred and gzipped (**.tgz**) format.



**NOTE:** Each version of the Network Agent package is supported on a single release of Junos OS only. The Junos OS version supported is identified by the Junos OS release number included in the Network Agent package name.

Examples of valid Network Agent package names including the following:

- **network-agent-x86-64-16.1R3.16-C1.0.tgz**
- **network-agent-x86-32-16.1R4.12-C1.1.tgz**

Before you begin:

- Install Junos OS Release 16.1R3 or later.
- Install the OpenConfig for Junos OS module and YANG Models. Using a Web browser, navigate to the All Junos Platforms software download URL on the Juniper Networks webpage: <http://www.juniper.net/support/downloads/>. From the **Network Management** tab, scroll down to select **OpenConfig**. Select the **Software** tab. Select the **OpenConfig Package (Junos with upgraded FreeBSD)**. Select **YANG Models**. For more information, see *Installing the OpenConfig Package*.
- Install Secure Sockets Layer (SSL) certificates of authentication on your Juniper Networks device.



**NOTE:** Only server-based SSL authentication is supported. Client-based authentication is not supported.

To download and install the Network Agent package:

1. Using a Web browser, navigate to the All Junos Platforms software download URL on the Juniper Networks webpage: <http://www.juniper.net/support/downloads/>.
2. Select the name of the Junos OS platform for the software that you want to download.
3. Select the release number (the number of the software version that you want to download) from the **Release** drop-down list to the right of the Download Software page.
4. Select the **Software** tab.
5. In the **Tools** section of the **Software** tab, select the **Junos Network Agent** package for the release.
6. Log in to the Juniper Networks authentication system using the username (generally your e-mail address) and password supplied by a Juniper Networks representative.

7. Download the software to a local host.
8. Copy the software to Juniper Networks device or to your internal software distribution site.
9. Install the new **network-telemetry** package on the device by issuing the **request system software add *package-name*** from the operational mode:

For example:

```
user@host > request system software add
network-telemetry-x86-64-16.1R3.16-C1.0.tgz
```



**NOTE:** The command uses the **validate** option by default. This option validates the software package against the current configuration as a prerequisite to adding the software package to ensure that the device reboots successfully. This is the default behavior when the software package being added is a different release.

10. Issue the **show version | grep na\ telemetry** command to verify that the Network Agent package was successfully installed.

```
user@host> show version | grep na\ telemetry
JUNOS na telemetry
[20161109.201405_builder_junos_161_r3]
```

For information about configuring gRPC services on your Juniper Networks device, see [“Configuring gRPC for the Junos Telemetry Interface” on page 57](#).

**Release History Table**

Release	Description
17.2R1	Starting with Junos OS Release 17.2R1, the Junos Network Agent Package is also supported on QFX10000 switches and QFX5200 switches.
16.1R3	Starting with Junos OS Release 16.1R3, the Junos Network Agent software package provides a framework to support OpenConfig and gRPC for the Junos Telemetry Interface on MX Series routers and PTX5000 routers.

**Related Documentation**

- [Understanding OpenConfig and gRPC on Junos Telemetry Interface on page 51](#)

## Configuring gRPC for the Junos Telemetry Interface

Starting with Junos OS Release 16.1R3 on MX Series routers and PTX3000 and PTX5000 routers, you can stream telemetry data for various network elements through gRPC, an open source framework for handling remote procedure calls based on TCP. The Junos Telemetry Interface relies on a so-called push model to deliver data asynchronously, which eliminates polling. For all Juniper devices that run a version of Junos OS with upgraded FreeBSD kernel, you must install the Junos Network Agent software package, which provides the interfaces to manage gRPC subscriptions. For Juniper Network devices that run other all other versions of the Junos OS, this functionality is embedded in the Junos OS software. For more information about installing the Junos Network Agent package, see [“Installing the Network Agent Package” on page 54](#).

The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

The Junos Telemetry Interface and gRPC streaming are supported on QFX10000 and QFX5200 switches, and PTX1000 routers starting with Junos OS Release 17.2R1.

Before you begin:

- Install Junos OS Release 16.1R3 or later on your Juniper Networks device.
- If your Juniper Networks device is running a version of Junos OS with an upgraded FreeBSD kernel, install the Junos Network Agent software package.
- Install the OpenConfig for Junos module. For more information see, *Installing the OpenConfig Package*.

To configure your system for gRPC services:

1. Specify the API connection setting either as unsecured or as based on Secure Socket Layer (SSL) technology. You can specify only one type of connection.

For example, to set the API connection as unsecured:

```
[edit system services]
user@host# set extension-service request-response grpc clear-text
```

For example, to set the API connection based on a SSL:

```
[edit system services]
user@host# set extension-service request-response grpc ssl
```

For an SSL-based connection, you must specify a local-certificate name. Optionally, you can specify an IP address to listen to for incoming connections (the default address is ::).

- a. Specify a local certificate-name, for example `jsd_certificate`:

```
[edit system services extension-service request-response grpc]
user@host# set ssl local-certificate jsd_certificate
```



**NOTE:** Enter the name of a certificate you have configured with local *certificate-name* statement at the [edit security certificates] hierarchy level.

- b. (Optional) Specify an IP address to listen to for incoming connections. for example, 192.0.2.0:

```
[edit system services extension-service request-response grpc]
user@host# set ssl ip-address 192.0.2.0
```



**NOTE:** If you do not specify an IP address, the default address of :: is used to listen for incoming connections.

2. Specify port 50051 for accepting incoming connections through gRPC.



**NOTE:** Port 50051 is the required port for gRPC streaming for both unsecured and SSL-based connections.

```
[edit system services extension-service request-response grpc]
user@host# set ssl port 50051
```

3. Specify client IP addresses from which notifications are allowed, for example, 0.0.0.0, which means any IP address:

```
[edit system services extension-service]
user@host# set notification allow-clients address 0.0.0.0
```

4. (Optional) Configure a WAN interface to your Juniper Networks device.



**BEST PRACTICE:** We recommend that you use a WAN interface to connect your Juniper Network device to the management station you configure to collect telemetry data.



Release History Table

Release	Description
17.2R1	The Junos Telemetry Interface and gRPC streaming are supported on QFX10000 and QFX5200 switches, and PTX1000 routers starting with Junos OS Release 17.2R1.
16.1R3	Starting with Junos OS Release 16.1R3 on MX Series routers and PTX3000 and PTX5000 routers, you can stream telemetry data for various network elements through gRPC, an open source framework for handling remote procedure calls based on TCP.

**Related  
Documentation**

- [Understanding OpenConfig and gRPC on Junos Telemetry Interface on page 51](#)
- [Importing SSL Certificates for Junos XML Protocol Support](#)

## Guidelines for gRPC Sensors (Junos Telemetry Interface)

Starting with Junos OS Release 16.1R3, the Junos Telemetry Interface supports gRPC remote procedure calls (gRPC) to provision sensors and to subscribe to and receive telemetry data on MX Series routers and PTX Series routers. The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

Starting with Junos OS Release 17.2R1, QFX10000 switches, QFX5200 switches, and PTX1000 routers are also supported. See [Table 7 on page 60](#) for information about which sensors are supported on which platforms.

The following sensors are supported with gRPC. To activate a sensor, use the corresponding resource path. Each resource path enables data streaming for the system resource globally, that is, systemwide. You can also modify each resource path, such as to specify a specific logical or physical interface. For example, to specify a specific interface, include the following at the end of the path: `[name='interface-name']/`

## Supported gRPC Sensors

See [Table 7 on page 60](#) for a description of supported gRPC sensors and the subscription path you use to provision the sensors.

Table 7: gRPC Sensors

resource path	Description
/junos/services/label-switched-path/usage/  mpls/lsp/constrained-path/tunnels/tunnel/	<p>Sensor for LSP statistics. Only ingress LSPs are supported. On MX Series routers only, the following are also supported: bidirectional LSPs for ultimate-hop popping (UHP).</p> <p>Starting with Junos OS Release 17.2R1, QFX10000 switches and PTX1000 routers are also supported.</p> <p>You can also specify an LSP name and source IP address at the end of the path: <code>[name='lsp-name', source=ip-address]</code></p> <p><b>NOTE:</b> When you enable a sensor for LSP statistics only, you must also configure the <code>sensor-based-stats</code> statement at the <code>[edit protocols mpls]</code> hierarchy level. MX Series routers should operate in enhanced mode. If not enabled by default, include either the <code>enhanced-ip</code> statement or the <code>enhanced-ethernet</code> statement at the <code>[edit chassis network-services]</code> hierarchy level.</p>
/mpls/lsp  /mpls/signaling-protocols	<p>Sensor for LSP events and properties.</p> <p>Supported on MX Series and PTX Series routers and QFX10000 switches starting with Junos OS Release 17.2R1.</p> <p>LSP events and properties are exported for ingress point-to-point LSPs, point-to-multipoint LSPs, bypass LSPs, and dynamically created LSPs.</p> <p>When you enable a sensor for LSP events and messages, you must also configured the <code>lsp-telemetry</code> statement at the <code>[edit routing-options]</code> hierarchy level.</p> <p>The following paths are also supported:</p> <ul style="list-style-type: none"> <li>• /mpls/lsp/constrained-path/tunnels/tunnel/p2p-tunnel-attributes/p2p-primary-paths/lsp-instances/state/notify-status</li> <li>• /mpls/lsp/constrained-path/tunnels/tunnel/p2p-tunnel-attributes/p2p-primary-paths/state/notify-status</li> <li>• /mpls/signaling-protocols/rsvp-te/sessions/session/state/notify-status</li> <li>• /mpls/lsp/constrained-path/tunnels/tunnel/p2p-tunnel-attributes/p2p-primary-paths/lsp-instances/state/bandwidth</li> <li>• /mpls/lsp/constrained-path/tunnels/tunnel/p2p-tunnel-attributes/p2p-primary-paths/lsp-instances/state/metric</li> <li>• /mpls/lsp/constrained-path/tunnels/tunnel/p2p-tunnel-attributes/p2p-primary-paths/state/explicit-path-name</li> <li>• /mpls/lsp/constrained-path/tunnels/tunnel/p2p-tunnel-attributes/p2p-primary-paths/lsp-instances/state/max-avg-bandwidth</li> </ul> <p><b>NOTE:</b> To specify a specific LSP name and source address, include <code>[name='lsp-name',source='address']</code> after <code>mpls/lsp/constrained-path-tunnels/tunnel/</code> in any of the supported paths. If do not include a specific LSP name, data is exported for all configured LSPs.</p>

Table 7: gRPC Sensors (*continued*)

resource path	Description
/junos/npu-memory/  /components/component [name="FPC<fpc-id>:NPU<npu-id>"]/properties/property	Sensor for network processing unit (NPU) memory, NPU memory utilization, and total memory available for each memory type  Supported on QFX10000 switches and PTX1000 routers starting with Junos OS Release 17.2R1.
/junos/system/linecard/cpu/memory/  /components/component [name="FPC<fpc-id>:CPU0"]/properties/property	Sensor for CPU memory.  Supported on QFX10000 switches and PTX1000 routers starting with Junos OS Release 17.2R1.
/bgp/neighbors/neighbor/  /bgp/peer-groups/peer-group/  /bgp/global/	Sensor for BGP peer information.  Supported on QFX10000 switches and QFX5200 switches starting with Junos OS Release 17.2R1.  <b>NOTE:</b> You can also include the following at the end path to <b>/bgp/neighbors/neighbor/</b> : <ul style="list-style-type: none"> <li>• state/session-state</li> <li>• state/messages/sent/update</li> <li>• state/messages/received/update</li> <li>• transport/state/local-address</li> <li>• transport/state/remote-address</li> <li>• state/peer-as</li> <li>• afi-safis/afi-safi/state/prefix-limit/state/max-prefixes</li> <li>• state/session-status</li> <li>• state/session-admin-status</li> <li>• state/session-established-transitions</li> <li>• state/interface-error</li> <li>• state/prefix-limited-exceeded</li> <li>• state/last-established</li> <li>• established-transitions</li> </ul> You can also include the following at the end path to <b>/bgp/global/</b> : <ul style="list-style-type: none"> <li>• afi-safis/afi-safi/state/total-prefixes</li> </ul> You can also include the following at the end path to <b>/bgp/peer-groups/peer-group/</b> : <ul style="list-style-type: none"> <li>• afi-safis/afi-safi/add-paths/eligible-prefix-policy</li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
<code>/junos/task-memory-information/</code>	<p>Sensor for memory utilization for routing protocol task.</p> <p>Supported on QFX10000 switches and QFX5200 switches starting with Junos OS Release 17.2R1.</p> <p>You can also include the following at the end path to <code>/junos/task-memory-information/</code>:</p> <ul style="list-style-type: none"> <li><code>task-memory-overall-report/task-size-block-list/task-size-block/tsb-size</code></li> <li><code>task-memory-overall-report/task-size-block-list/task-size-block/tsb-alloc-bytes</code></li> <li><code>task-memory-overall-report/task-size-block-list/task-size-block/tsb-allocs</code></li> <li><code>task-memory-overall-report/task-size-block-list/task-size-block/tsb-max-allocs</code></li> <li><code>task-memory-overall-report/task-size-block-list/task-size-block/tsb-max-bytes</code></li> <li><code>task-memory-overall-report/task-size-block-list/task-size-block/tsb-free-bytes</code></li> <li><code>task-memory-overall-report/task-memory-total-bytes</code></li> <li><code>task-memory-overall-report/task-memory-total-max-bytes</code></li> <li><code>task-memory-information/task-memory-overall-report/task-memory-total-free-bytes</code></li> <li><code>task-memory-allocator-report/task-block-list/task-block/tb-name</code></li> <li><code>task-memory-allocator-report/task-block-list/task-block/tb-size</code></li> <li><code>task-memory-allocator-report/task-block-list/task-block/tb-alloc-size</code></li> <li><code>task-memory-allocator-report/task-block-list/task-block/tb-alloc-blocks</code></li> <li><code>task-memory-allocator-report/task-block-list/task-block/tb-alloc-bytes</code></li> <li><code>task-memory-allocator-report/task-block-list/task-block/tb-max-alloc-blocks</code></li> <li><code>task-memory-allocator-report/task-lite-page-list/task-lite-page/tlp-name</code></li> <li><code>task-memory-allocator-report/task-lite-page-list/task-lite-page/tlp-alloc-bytes</code></li> <li><code>task-memory-allocator-report/task-memory-total-bytes</code></li> <li><code>task-memory-information/task-memory-allocator-report/task-memory-total-max-bytes</code></li> <li><code>task-memory-malloc-usage-report/task-malloc-list/task-malloc/tm-name</code></li> <li><code>task-memory-malloc-usage-report/task-malloc-list/task-malloc/tm-allocs</code></li> <li><code>task-memory-malloc-usage-report/task-malloc-list/task-malloc/tm-alloc-bytes</code></li> <li><code>task-memory-malloc-usage-report/task-malloc-list/task-malloc/tm-max-allocs</code></li> <li><code>task-memory-malloc-usage-report/task-malloc-list/task-malloc/tm-max-alloc-bytes</code></li> <li><code>task-memory-malloc-usage-report/task-malloc-list/task-malloc/tm-function-calls</code></li> <li><code>task-memory-malloc-usage-report/task-memory-total-bytes</code></li> <li><code>task-memory-malloc-usage-report/task-memory-total-max-bytes</code></li> <li><code>task-memory-max-dynamic-allocs</code></li> <li><code>task-memory-bss-bytes</code></li> <li><code>task-memory-max-bss-bytes</code></li> <li><code>task-memory-page-data-bytes</code></li> <li><code>task-memory-max-page-data-bytes</code></li> <li><code>task-memory-dir-bytes</code></li> <li><code>task-memory-max-dir-bytes</code></li> <li><code>task-memory-total-bytes-in-use</code></li> <li><code>task-memory-total-bytes-percent</code></li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
/junos/system/linecard/firewall/	Sensor for firewall filter counters and policer counters. Each line card reports counters separately.
/junos/firewall/firewall-stats/	<p>Supported on QFX10000 switches starting with Junos OS Release 17.2R1.</p> <p><b>NOTE:</b> Hierarchical policer statistics are collected for MX Series routers only. Traffic-class counter statistics are collected for PTX Series routers and QFX10000 switches only.</p> <p>Firewall counters are exported even if the interface to which the firewall filter is attached is operationally down.</p> <p>The following paths are also supported:</p> <ul style="list-style-type: none"> <li>• junos/firewall/firewall-stats/[name='filter-name']/timestamp</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/memory-usage/[name='memory-type']/allocated</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/counter-stats/[name='counter-name']/packets</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/counter-stats/[name='counter-name']/bytes</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/policer-stats/[name='policer-name']/out-of-spec-packets</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/policer-stats/[name='policer-name']/out-of-spec-bytes</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/policer-stats/[name='policer-name']/offered-packets</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/policer-stats/[name='policer-name']/offered-bytes</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/policer-stats/[name='policer-name']/transmitted-packets</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/policer-stats/[name='policer-name']/transmitted-bytes</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/hierarchical-policer-stats/[name='hierarchical-policer-name']/premium-packets (MX Series only)</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/hierarchical-policer-stats/[name='hierarchical-policer-name']/premium-bytes (MX Series only)</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/hierarchical-policer-stats/[name='hierarchical-policer-name']/aggregate-packets (MX Series only)</li> <li>• /junos/firewall/firewall-stats/[name='filter-name']/hierarchical-policer-stats/[name='hierarchical-policer-name']/aggregate-bytes (MX Series only)</li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
/interfaces/interface/	

Table 7: gRPC Sensors (*continued*)

resource path	Description
	<p>Sensor for physical interface traffic.</p> <p><b>NOTE:</b> For PTX Series routers, for a specific interface, queue statistics are exported for each line card. For MX series routers, interface queue statistics are exported only from slot on which an interface is configured.</p> <p>For Aggregated Ethernet interfaces, statistics are exported for the member physical interfaces. You must aggregate the counters at the destination server, or collector.</p> <p>If a physical interface is administratively down or operationally down, interface counters are not exported.</p> <p>Only fields with a non-zero value are exported.</p> <p>Supported on QFX10000 switches and PTX1000 routers starting with Junos OS Release 17.2R1.</p> <p>The following paths are also supported:</p> <ul style="list-style-type: none"> <li>• /interfaces/interface[name='interface-name']/parent_ae_name</li> <li>• /interfaces/interface[name='interface-name']/oper-status</li> <li>• /interfaces/interface[name='interface-name']/carrier-transitions</li> <li>• /interfaces/interface[name='interface-name']/last-change</li> <li>• /interfaces/interface[name='interface-name']/high-speed</li> <li>• /interfaces/interface[name='interface-name']/counters/out-octets</li> <li>• /interfaces/interface[name='interface-name']/counters/out-unicast-pkts</li> <li>• /interfaces/interface[name='interface-name']/counters/out-multicast-pkts</li> <li>• /interfaces/interface[name='interface-name']/counters/out-broadcast-pkts</li> <li>• /interfaces/interface[name='interface-name']/counters/out-errors</li> <li>• /interfaces/interface[name='interface-name']/counters/in-octets</li> <li>• /interfaces/interface[name='interface-name']/counters/in-unicast-pkts</li> <li>• /interfaces/interface[name='interface-name']/counters/in-multicast-pkts</li> <li>• /interfaces/interface[name='interface-name']</li> <li>• /interfaces/interface[name='interface-name']/counters/in-broadcast-pkts</li> <li>• /interfaces/interface[name='interface-name']/counters/in-errors</li> <li>• /interfaces/interface[name='interface-name']/in-pause-pkts</li> <li>• /interfaces/interface[name='interface-name']/out-pause-pkts</li> <li>• /interfaces/interface[name='interface-name']/in-queue [queue-number=queue_number]/</li> <li>• /interfaces/interface[name='interface-name']/in-queue [queue-number=queue_number]/ pkts</li> <li>• /interfaces/interface[name='interface-name']/in-queue [queue-number=queue_number]/bytes</li> <li>• /interfaces/interface[name='interface-name']/in-queue [queue-number=queue_number]/tail-drop-pkts</li> <li>• /interfaces/interface[name='interface-name']/in-queue [queue-number=queue_number]/ rl-drop-pkts</li> <li>• /interfaces/interface[name='interface-name']/in-queue [queue-number=queue_number]/ rl-drop-bytes</li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
	<ul style="list-style-type: none"> <li>• /interfaces/interface[name='interface-name']/in-queue[queue-number=quene_number]/avg-buffer-occupancy</li> <li>• /interfaces/interface[name='interface-name']/in-queue[queue-number=quene_number]/cur-buffer-occupancy</li> <li>• /interfaces/interface[name='interface-name']/in-queue[queue-number=quene_number]/peak-buffer-occupancy</li> <li>• /interfaces/interface[name='interface-name']/in-queue[queue-number=quene_number]/allocated-buffer-size</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/pkts</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/bytes</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/tail-drop-pkts</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/rl-drop-pkts</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/rl-drop-bytes</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/red-drop-pkts</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/red-drop-bytes</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/avg-buffer-occupancy</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/cur-buffer-occupancy</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/ peak-buffer-occupancy</li> <li>• /interfaces/interface[name='interface-name']/out-queue[queue-number=quene_number]/allocated-buffer-size</li> </ul>



Table 7: gRPC Sensors (*continued*)

resource path	Description
/interfaces/interface/subinterfaces/	Sensor for logical interface traffic.
/interfaces/interface[name='interface-name']/subinterfaces/	<p><b>NOTE:</b> If a logical interface is operationally down, interface statistics continue to be exported.</p> <p>Supported on QFX10000 switches starting with Junos OS Release 17.2R1.</p> <p>The following paths are also supported:</p> <ul style="list-style-type: none"> <li>• /interfaces/interface/subinterfaces/subinterface/name</li> <li>• /interfaces/interface/subinterfaces/subinterface/ifindex</li> <li>• /interfaces/interface/subinterfaces/subinterface/snmp_index</li> <li>• /interfaces/interface/subinterfaces/subinterface/admin_status</li> <li>• /interfaces/interface/subinterfaces/subinterface/oper_status</li> <li>• /interfaces/interface/subinterfaces/subinterface/last_change</li> <li>• /interfaces/interface/subinterfaces/subinterface/high_speed</li> <li>• /interfaces/interface/subinterfaces/subinterface/description</li> <li>• /interfaces/interface/subinterfaces/subinterface/enabled</li> <li>• /interfaces/interface/subinterfaces/subinterface/subunit</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/in_octets</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/in_unicast_pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/in_broadcast_pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/in_multicast_pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/in_discards</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/in_errors</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/in_unknown_protos</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/out_octets</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/out_unicast_pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/out_broadcast_pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/out_multicast_pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/out_discards</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/out_errors</li> <li>• /interfaces/interface/subinterfaces/subinterface/oob_states/last_clear</li> </ul>
/junos/system/linecard/optics/	<p>Sensor for various optical interface performance metrics, such as transmit and receive power levels.</p> <p>Supported on QFX10000 switches starting with Junos OS Release 17.2R1.</p>

Table 7: gRPC Sensors (*continued*)

resource path	Description
<code>/junos/rsvp-interface-information/</code>	<p>Sensor for events and properties for RSVP interfaces.</p> <p><b>NOTE:</b> For 100 RSVP logical interfaces, configure a sampling interval equal to 60 seconds. For 200 RSVP logical interfaces, configure a sampling interval equal to 180 seconds.</p> <p>Supported on QFX10000 switches and QFX5200 switches starting with Junos OS Release 17.2R1.</p> <p>You can also add the following to the end path for <code>/junos/rsvp-interface-information/</code>:</p> <ul style="list-style-type: none"> <li>• <code>active-count</code></li> <li>• <code>rsvp-interface/interface-name</code></li> <li>• <code>rsvp-interface/index</code></li> <li>• <code>rsvp-interface/rsvp-status</code></li> <li>• <code>rsvp-interface/authentication-flag</code></li> <li>• <code>rsvp-interface/aggregate-flag</code></li> <li>• <code>rsvp-interface/ack-flag</code></li> <li>• <code>rsvp-interface/protect-flag</code></li> <li>• <code>rsvp-interface/hello-interval</code></li> <li>• <code>rsvp-interface/interface-address</code></li> <li>• <code>message-statistics/rsvp-message</code></li> <li>• <code>rsvp-interface/message-statistics/messages-sent</code></li> <li>• <code>rsvp-interface/message-statistics/messages-received</code></li> <li>• <code>rsvp-interface/message-statistics/messages-sent-5seconds</code></li> <li>• <code>rsvp-interface/message-statistics/messages-received-5seconds</code></li> <li>• <code>rsvp-interface/rsvp-telink/active-reservation</code></li> <li>• <code>rsvp-interface/rsvp-telink/preemption-count</code></li> <li>• <code>rsvp-interface/rsvp-telink/update-threshold</code></li> <li>• <code>rsvp-interface/rsvp-telink/subscription</code></li> <li>• <code>rsvp-interface/rsvp-telink/static-bandwidth</code></li> <li>• <code>rsvp-interface/rsvp-telink/available-bandwidth</code></li> <li>• <code>rsvp-interface/rsvp-telink/reserved-bandwidth/bandwidth-priority</code></li> <li>• <code>rsvp-interface/rsvp-telink/reserved-bandwidth/total-reserved-bandwidth</code></li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
/components/component/	
/chassis-inventory/chassis-modules/chassis-module/	

Table 7: gRPC Sensors (*continued*)

resource path	Description
	<p>Sensor for operational state of Routing Engines, power supply modules, Switch Fabric Boards, Control Boards, Switch Interface Boards, Modular Interface Cards, and Physical Interface Cards.</p> <p><b>NOTE:</b></p> <p>Supported on QFX10000 switches and QFX5200 switches starting with Junos OS Release 17.2R1.</p> <p>You can also add the following to each of the paths::</p> <ul style="list-style-type: none"> <li>• name</li> <li>• cidr</li> <li>• version</li> <li>• part_number</li> <li>• serial_number</li> <li>• description</li> <li>• clei_code</li> <li>• model</li> <li>• vendor_name</li> <li>• properties/property/state</li> <li>• properties/property/state_offline_reason (MX Series only)</li> <li>• properties/property/power_usage</li> <li>• properties/property/power_maximum</li> <li>• properties/property/temperature_intake</li> <li>• properties/property/temperature_exhaust_a (not supported on PTX1000 and PTX3000 routers)</li> <li>• properties/property/temperature_exhaust_b (not supported on PTX1000 and PTX3000 routers)</li> <li>• properties/property/temperature_exhaust (not supported on PTX1000 and PTX5000 routers)</li> <li>• properties/property/cpu_utilization_total</li> <li>• properties/property/memory_dram_used</li> <li>• properties/property/memory_utilization_heap</li> <li>• properties/property/memory_utilization_buffer</li> <li>• properties/property/uptime</li> </ul> <p>The following paths are also supported only for Routing Engine statistics:</p> <ul style="list-style-type: none"> <li>• properties/property/mastership-state</li> <li>• properties/property/mastership-priority</li> <li>• properties/property/temperature-cpu</li> <li>• properties/property/memory-dram-installed</li> <li>• properties/property/cpu-utilization-user</li> <li>• properties/property/cpu-utilization-background</li> <li>• properties/property/cpu-utilization-kernel</li> <li>• properties/property/cpu-utilization-idle</li> <li>• properties/property/reboot-reason</li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
	<p>The following paths are also supported for power modules:</p> <ul style="list-style-type: none"> <li>• <code>properties/property/power-zone-upper-capacity</code></li> <li>• <code>properties/property/power-zone-upper-maximum</code></li> <li>• <code>properties/property/power-zone-upper-allocated</code></li> <li>• <code>properties/property/power-zone-upper-remaining</code></li> <li>• <code>properties/property/power-zone-upper-usage</code></li> <li>• <code>properties/property/power-zone-lower-capacity</code></li> <li>• <code>properties/property/power-zone-lower-maximum</code></li> <li>• <code>properties/property/power-zone-lower-allocated</code></li> <li>• <code>properties/property/power-zone-lower-remaining</code></li> <li>• <code>properties/property/power-zone-lower-usage</code></li> <li>• <code>properties/property/power-zone-0-capacity</code></li> <li>• <code>properties/property/power-zone-0-maximum</code></li> <li>• <code>properties/property/power-zone-0-allocated</code></li> <li>• <code>properties/property/power-zone-0-remaining</code></li> <li>• <code>properties/property/power-zone-0-usage</code></li> <li>• <code>properties/property/power-zone-1-capacity</code></li> <li>• <code>properties/property/power-zone-1-maximum</code></li> <li>• <code>properties/property/power-zone-1-allocated</code></li> <li>• <code>properties/property/power-zone-1-remaining</code></li> <li>• <code>properties/property/power-zone-1-usage</code></li> <li>• <code>properties/property/power-system-capacity</code></li> <li>• <code>properties/property/power-system-allocated</code></li> <li>• <code>properties/property/power-system-remaining</code></li> <li>• <code>properties/property/power-system-usage</code></li> <li>• <code>properties/property/temperature-ambient</code></li> </ul> <p>The following paths are supported for either Switch Fabric Board or Control Boards or both:</p> <ul style="list-style-type: none"> <li>• <code>properties/property/temperature-zone-0-intake</code> (SFB only)</li> <li>• <code>properties/property/temperature-zone-0-intake-a</code> (both SFB and CB)</li> <li>• <code>properties/property/temperature-zone-1-intake-b</code> (both SFB and CB)</li> <li>• <code>properties/property/temperature-zone-0-exhaust</code> (SFB only)</li> <li>• <code>properties/property/temperature-zone-1-exhaust</code> (SFB only)</li> <li>• <code>properties/property/temperature-zone-0-intake-c</code> (CB only)</li> <li>• <code>properties/property/temperature-zone-0-exhaust-a</code> (CB only)</li> <li>• <code>properties/property/temperature-zone-1-exhaust-b</code> (CB only)</li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
<code>/lACP/</code>	<p>Sensor for operational state of aggregated Ethernet interfaces configured with the Link Aggregation Control Protocol.</p> <p>Supported on QFX10000 switches and QFX5200 switches starting with Junos OS Release 17.2R1.</p> <p>You can also add the following to the end of the path for <code>/lACP/</code>:</p> <ul style="list-style-type: none"> <li><code>state/system-priority</code></li> <li><code>interfaces/interface[name='aggregate-interface-name']/state/</code></li> <li><code>interfaces/interface[name='aggregate-interface-name']/members/member[interface='interface-name']/state/</code></li> <li><code>interfaces/interface[name='aggregate-interface-name']/members/member[interface='interface-name']/state/counters/</code></li> <li><code>interfaces/interface[name='aggregate-interface-name']/members/member[interface='interface-name']/state/port-num</code></li> <li><code>interfaces/interface[name='aggregate-interface-name']/members/member[interface='interface-name']/state/partner-port-num</code></li> <li><code>interfaces/interface[name='aggregate-interface-name']/members/member[interface='interface-name']/state/mux-state</code></li> </ul>
<code>/lldp/</code>	<p>Sensor for operational state of Ethernet interfaces enabled with the Link Layer Discovery Protocol.</p> <p>Supported on QFX10000 switches and QFX5200 switches starting with Junos OS Release 17.2R1.</p> <p>You can also add the following to the end of the path for <code>/lldp/</code>:</p> <ul style="list-style-type: none"> <li><code>state/</code></li> <li><code>state/enabled</code></li> <li><code>state/hello-timer</code></li> <li><code>state/system-name</code></li> <li><code>state/system-description</code></li> <li><code>state/chassis-id</code></li> <li><code>state/loc-port-id-type</code></li> <li><code>interfaces/interface[name='interface-name']/state/</code></li> <li><code>interfaces/interface[name='interface-name']/state/counters/</code></li> <li><code>interfaces/interface[name='interface-name']/neighbors/</code></li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
<code>/arp-information/</code>	<p>Sensor for Address Resolution Protocol (ARP) statistics.</p> <p>Supported on QFX10000 and QFX5200 switches starting with Junos OS Release 17.2R1.</p> <p>You can also add the following to the end path for <code>/arp-information/</code></p> <ul style="list-style-type: none"> <li>• <code>ipv4</code></li> <li>• <code>ipv4/neighbors</code></li> <li>• <code>ipv4/neighbors/neighbor</code></li> <li>• <code>ipv4/neighbors/neighbor/ip</code></li> <li>• <code>ipv4/neighbors/neighbor/link-layer-address</code></li> <li>• <code>ipv4/neighbors/neighbor/origin</code></li> <li>• <code>ipv4/neighbors/neighbor/host-name</code></li> <li>• <code>ipv4/neighbors/neighbor/rtb-id</code></li> <li>• <code>ipv4/neighbors/neighbor/state</code></li> <li>• <code>ipv4/neighbors/neighbor/expiry</code></li> <li>• <code>ipv4/neighbors/neighbor/ispublish</code></li> <li>• <code>ipv4/neighbors/neighbor/interface-name</code></li> <li>• <code>ipv4/neighbors/neighbor/logical-router-id</code></li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
<code>/interfaces/interface[name='interface-name']/</code>	



Table 7: gRPC Sensors (*continued*)

resource path	Description
	<p>Sensor for Routing Engine internal interfaces.</p> <p><b>NOTE:</b> On MX Series routers, you can specify the following interfaces: fxp0, em0, and em1</p> <p>On PTX Series routers, you can specify the following interfaces: em0, ixlv0, ixlv1</p> <p>On PTX Series routers with dual Routing Engines, you can specify the following interfaces: em0, ixgbe0, ixgbe1</p> <p>The following end paths are also supported:</p> <ul style="list-style-type: none"> <li>• <code>/interfaces/interface/state/type</code></li> <li>• <code>/interfaces/interface/state/mtu</code></li> <li>• <code>/interfaces/interface/state/name</code></li> <li>• <code>/interfaces/interface/state/description</code></li> <li>• <code>/interfaces/interface/state/enabled</code></li> <li>• <code>/interfaces/interface/state/ifindex</code></li> <li>• <code>/interfaces/interface/state/admin-status</code></li> <li>• <code>/interfaces/interface/state/oper-status</code></li> <li>• <code>/interfaces/interface/state/last-change</code></li> <li>• <code>/interfaces/interface/state/speed</code></li> <li>• <code>/interfaces/interface/state/counters/in-octets</code></li> <li>• <code>/interfaces/interface/state/counters/in-unicast-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/in-broadcast-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/in-multicast-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/in-discards</code></li> <li>• <code>/interfaces/interface/state/counters/in-errors</code></li> <li>• <code>/interfaces/interface/state/counters/in-unknown-protos</code></li> <li>• <code>/interfaces/interface/state/counters/out-octets</code></li> <li>• <code>/interfaces/interface/state/counters/out-unicast-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/out-broadcast-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/out-multicast-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/out-discards</code></li> <li>• <code>/interfaces/interface/state/counters/out-errors</code></li> <li>• <code>/interfaces/interface/state/counters/last-clear</code></li> <li>• <code>/interfaces/interface/state/counters/in-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/in-sec-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/in-sec-octets</code></li> <li>• <code>/interfaces/interface/state/counters/in-pause-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/out-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/out-sec-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/out-sec-octets</code></li> <li>• <code>/interfaces/interface/state/counters/out-pause-pkts</code></li> <li>• <code>/interfaces/interface/state/counters/in-drops</code></li> <li>• <code>/interfaces/interface/state/counters/in-frame-errors</code></li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
	<ul style="list-style-type: none"> <li>• /interfaces/interface/state/counters/in-runs</li> <li>• /interfaces/interface/state/counters/in-lchan-errors</li> <li>• /interfaces/interface/state/counters/in-l-mismatch-errors</li> <li>• /interfaces/interface/state/counters/in-fifo-errors</li> <li>• /interfaces/interface/state/counters/in-giants</li> <li>• /interfaces/interface/state/counters/in-resource-errors</li> <li>• /interfaces/interface/state/counters/out-drops</li> <li>• /interfaces/interface/state/counters/carrier-transitions</li> <li>• /interfaces/interface/state/counters/mtu-errors</li> <li>• /interfaces/interface/state/counters/out-resource-errors</li> <li>• /interfaces/interface/subinterfaces/subinterface/index</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/index</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/name</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/description</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/enabled</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/ifindex</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/admin-status</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/oper-status</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/last-change</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-octets</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-unicast-pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-broadcast-pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-multicast-pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-discards</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-errors</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/in-unknown-protos</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/out-octets</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/out-unicast-pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/out-broadcast-pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/out-multicast-pkts</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/out-discards</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/out-errors</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/last-clear</li> <li>• /interfaces/interface/subinterfaces/subinterface/state/counters/out-pkts</li> </ul>

Table 7: gRPC Sensors (*continued*)

resource path	Description
/nd6-information/	<p>Sensor for Network Discovery Protocol (NDP) table state.</p> <p>Supported on QFX10000 and QFX5200 switches starting with Junos OS Release 17.2R1.</p> <p>You can also add the following to the end path for <b>nd6-information/</b></p> <ul style="list-style-type: none"> <li>• ipv6</li> <li>• ipv6/neighbors</li> <li>• ipv6/neighbors/neighbor</li> <li>• ipv6/neighbors/neighbor/ip</li> <li>• ipv6/neighbors/neighbor/link-layer-address</li> <li>• ipv6/neighbors/neighbor/origin</li> <li>• ipv6/neighbors/neighbor/isrouter</li> <li>• ipv6/neighbors/neighbor/state</li> <li>• ipv6/neighbors/neighbor/rtb-id</li> <li>• ipv6/neighbors/neighbor/issecure</li> <li>• ipv6/neighbors/neighbor/ispublish</li> <li>• ipv6/neighbors/neighbor/expiry</li> <li>• ipv6/neighbors/neighbor/interface-name</li> <li>• ipv6/neighbors/neighbor/logical-router-id</li> </ul>
/ipv6-ra/	Sensor for NDP router-advertisement statistics.

## Release History Table

Release	Description
17.2R1	Starting with Junos OS Release 17.2R1, QFX10000 switches, QFX5200 switches, and PTX1000 routers are also supported.
16.1R3	Starting with Junos OS Release 16.1R3, the Junos Telemetry Interface supports gRPC remote procedure calls (gRPC) to provision sensors and to subscribe to and receive telemetry data on MX Series routers and PTX Series routers. The Junos Telemetry Interface is not supported on MX80 and MX104 routers.

## Related Documentation

- [Understanding OpenConfig and gRPC on Junos Telemetry Interface on page 51](#)



## PART 4

# Best Practices

- [Best Practices for Implementing the Junos Telemetry Interface on page 81](#)



## CHAPTER 7

# Best Practices for Implementing the Junos Telemetry Interface

- [Guidelines for Specifying Data Reporting Intervals Junos Telemetry Interface on page 81](#)
- [Guidelines for Aggregating Junos Telemetry Interface Data on page 82](#)

### Guidelines for Specifying Data Reporting Intervals Junos Telemetry Interface

---

The Junos Telemetry Interface enables you to provision sensors to collect and export data for various system resources without involving polling. A request to send data is sent once by a management station to stream periodic updates.

You can configure telemetry sensors to report data at a specified interval either through the command-line interface (CLI) or through the OpenConfig for Junos **telemetrySubscribe** remote procedure call (RPC). To configure using the CLI, include the **reporting-rate seconds** statement at the **[edit services analytics export-profile profile-name]** hierarchy level. For the **telemetrySubscribe** RPC, specify the sampling interval parameter, in milliseconds. In both cases, the interval specifies the amount of time between each subsequent export of data.

### How to Determine the Reporting Interval for a System Resource

To determine the appropriate reporting interval for a specific system resource, follow these guidelines:

- Identify the required export interval for a given object, such as an interface.
- Identify the maximum number of objects reported by the sensor, such as the number of physical interfaces configured on a line card.
- Identify the minimum number of objects reported on each interval for a given sensor.
- Use the following formula to determine the best reporting interval:
  - $\text{Reporting interval} = \text{Required Export Interval Per Object} * \text{Minimum Number of objects reported on each Interval} / \text{Maximum Number of Objects}$ .

Consider this example. There is a business requirement to report interface statistics every 30 seconds. At every interval, 10 interface records are reported, and the total number of interfaces is 96 for each line card. Using the reporting-interval formula, the reporting

interval should be 3.125 seconds. Currently, the reporting interval can be configured only as a multiple of 2, in seconds. Therefore, for this example, configure the reporting interval as 2 seconds in the CLI or 2000 milliseconds in the OpenConfig RPC.



**TIP:** The same metric might be reported more than once over a 30-second interval. For the purposes of effective visualization and data manipulation, it is quite common to aggregate data over fixed time spans.

**Related Documentation** • [Overview of the Junos Telemetry Interface on page 3](#)

---

## Guidelines for Aggregating Junos Telemetry Interface Data

---

One important feature of the Junos Telemetry Interface is that data processing occurs at the collector that streams data, rather than the device. Data is not automatically aggregated, but it can be aggregated for analysis.

Data aggregation is useful in the following scenarios:

- Data for the same metric over fixed spans of time, such as, the average number physical interface ingress errors over a 30-second interval.
- Data from different sources (such as multiple line cards) for the same metric, such as label-switched path (LSP) statistics or filter counter statistics.
- Data from multiple sources, such as input and output statistics for aggregated Ethernet interfaces.

The follow sections describe how to perform data aggregation for various scenarios. The examples in these sections use the InfluxDB time-series database to accept queries on telemetry data. InfluxDB is an open source database written in Go specifically to handle time-series data.

### Aggregating Data Over Fixed Time Spans

Aggregating data for the same metric over fixed spans of time is a common and useful way to detect trends. Metrics can include gauges, that is, single values, or cumulative counters. You might also want to aggregate data continuously.

---

#### Example: Aggregating Data for Gauge Metrics

---

In this example, data for

`JuniperNetworksSensors.jnpr_interface_ext.interface_stats.egress_queue_info.current_buffer_occupancy` from `port.proto` is written to the InfluxDB database with tags that identify the host name, an interface name and corresponding queue number and measurement called `current_buffer_occupancy`. See [Table 8 on page 83](#) for the specific values used in this example.



Table 8: Telemetry Data Values

Time Stamp (seconds)	Value	Tags
1458704133	1547	queue_number=0,interface_name='xe-1/0/0',host='sjc-a'
1458704143	3221	queue_number=0,interface_name='xe-1/0/0',host='sjc-a'
1458704155	4860	queue_number=0,interface_name='xe-1/0/0',host='sjc-a'
1458704166	6550	queue_number=0,interface_name='xe-1/0/0',host='sjc-a'

Each measurement data point has a timestamp and recorded value. In this example, the tag **queue\_number** is the numerical identifier of the interface queue.

To aggregate this data over 30-second intervals, use the following influxDB query:

```
select mean(value) from current_buffer_occupancy
  where time >= $time_start and time <= $time_end and
        queue_number='0' and interface_name='xe-1/0/0' and host='sjc-a'
 group by time(30s)
```

For **\$time\_start** and **\$time\_end**, specify the actual range of time.

#### Example: Aggregating Data for Cumulative Statistics

Some Junos Telemetry Interface sensors report cumulative counter values, such as the number of ingress packets, defined as

**JuniperNetworksSensors.jnpr\_interface\_ext.interface\_stats.ingress\_stats.packets.**

It is common to derive traffic rates from packet or byte counters. Unlike with gauge metrics, the initial data point in the series for cumulative counters is used only to set the baseline.

Use the following guidelines to create a database query for cumulative statistics:

- Calculate the cumulative value for a specific time interval. You can calculate either an average among several data points recorded during the time interval, or you can interpolate a value. All data points should belong to the same series. If a counter reset has occurred between the two data points reported at different times, do not use both data points.
- Determine the appropriate value for the previous time interval. If a counter has been reset since the last update, declare that value as unavailable.
- If the previous interval is available, calculate the difference between the data points and the traffic rate.

These guidelines are summarized in the following influxDB query. This query assumes that data is stored in the measurement **ingress\_packets**. The query uses the same tags as the gauge metric example as well as the tag for counter initialization time, **init\_time**.

The query uses average values over a 30-second time interval. It calculates the rate for the metrics that have the same counter initialization.

```
select non_negative_derivative(mean(value)) from ingress_packets
  where time >= $time_start and time <= $time_end and
         interface_name='xe-1/0/0' and host='sjc-a'
  group by time(30s), init_time
```

Use the following query to calculate the number of packets received over an interval of time, without deriving the rate.

```
select difference(mean(value)) from ingress_packets
  where time >= $time_start and time <= $time_end and
         interface_name='xe-1/0/0' and host='sjc-a'
  group by time(30s), init_time
```

In some cases, more than one aggregated data point is returned by the query for a particular time interval. For example, four data points are available for a time interval. Two data points have `init_time t0`, and the other two have `init_time t1`. You can run a query that uses the last change timestamp tag, `last_change`, instead of `init_time`, to calculate the difference and to derive the rate between the two data points with the same last change timestamp.

```
select difference(mean(value)) from ingress_packets
  where time >= $time_start and time <= $time_end and
         interface_name='xe-1/0/0' and host='sjc-a'
  group by time(30s), last_change
```



**TIP:** These queries can all be run as continuous queries and can periodically populate new time-series measurements.

---

## Aggregating Data From Multiple Sources

Certain metrics are reported from multiple line cards or packet forwarding engines. It is useful to aggregate data derived from different sources in the following scenarios:

- Packet and byte counts for label-switched paths (LSPs) are reported separately by each line card. However, a view of LSP paths for the entire device is required for path computation element controllers.
- For Juniper Networks devices that support virtual output queues, the tail drop or random early detection drop statistics for each queue are reported separately by each line card for every physical interface. It is useful to be able to aggregate the statistics for all the line cards for an interface.
- Filter counters for a firewall filter attached to a forwarding table or to an aggregated Ethernet interface are reported separately by each line card. It is useful to aggregate the statistics for all the line cards.

To aggregate data from multiple sources, perform the following:

1. Aggregate data for a specific period of time for each source, for example, each line card.
2. Aggregate the data you derive for each source in *step 1*.

For data stored in an InfluxDB database, you can complete *step 1* in the procedure by running a continuous query and populating a new measurement. We strongly recommend that you group the data points according to each source. For example, for LSP statistics, the **component\_id** in the the gpb message identifies the line card sending the data. Group the data points based on each unique **component\_id**.

### Example: Aggregating Data from Multiple Sources

In this example, you run two queries to derive the LSP packet rate for data from all line cards.

First, you run the following continuous query on the measurement named **lsp\_packet\_count** for each **component\_id** tag and the **counter\_name** tag. Each unique **component\_id** tag corresponds to a different line card. This query populates a new measurement, **lsp\_packet\_rate**.

```
select non_negative_derivative(mean(value)) as value from lsp_packet_count
into lsp_packet_rate
group by time(30s), component_id, counter_name, host
```



**NOTE:** The LSP statistics sensor does not report counter initialization time.

Use the new measurement derived from this continuous query—**lsp\_packet\_count**—to run the following query, which aggregates data from all line cards for packet rates for an LSP named **lsp-sjc-den-1**.

```
select sum(value) from lsp_packet_rate
where counter_name='lsp-sjc-den-1', host='sjc-a'
```



**NOTE:** Because this query does not group data according to the **component\_id** tag, or line card, the LSP packet rates from all components, or line cards, are returned.

## Aggregating Data for Multiple Metrics

It can be useful to aggregate metrics for multiple values. For example, for aggregated Ethernet interfaces, you would typically want to track packet and byte rates for each interface member as well as interface utilization for the aggregated link.

### Example: Aggregating Multiple Metric Values

---

In this example, you run the following two queries:

- Continuous query to derive ingress packet counts for each member link in an aggregated Ethernet interface
- Query to aggregate packet count data for all the member links that belong to the same aggregated Ethernet interface

The following continuous query derives a measurement, **ingress\_packets**, for each member link in an aggregated Ethernet interface. The **interface\_name** tag identifies each member interface. You also use the **parent\_ae\_name** tag to identify membership in a specific aggregated Ethernet interface. Grouping each member link with the **parent\_ae\_name** tag ensures that data is collected only for current member links. For example, an interface might change its membership during the reporting interval. Grouping member interfaces with the specific aggregated Ethernet interface means that data for the member link will not be transferred to the new aggregated Ethernet interface of which it is now a member.

```
select difference(mean(value)) as value from ingress_packets
into ingress_packets_difference
group by time(30s), component_id, interface_name, host, parent_ae_name
```

The following query aggregates data for the ingress packets for the aggregated Ethernet interface, that is all member links.

```
select sum(value) from ingress_packets_difference
where parent_ae_name='ae0' and host='sjc-a'
```



**NOTE:** This query aggregates data for aggregated Ethernet interface ae0. The **parent\_ae\_name** tag does not verify the actual member links.

---

#### Related Documentation

- [Overview of the Junos Telemetry Interface on page 3](#)