

Juniper Extension Toolkit API Guide

Published
2024-06-13

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, 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.

Juniper Extension Toolkit API Guide

Copyright © 2024 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 <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide | iv

1

About JET APIs

Overview of JET APIs | 2

JET Notification API | 4

About This Guide

Use this guide to learn more about Juniper Extension Toolkit (JET) APIs. For how to develop JET applications, see the [Juniper Extension Toolkit Developer Guide](#). For information about gRPCs that use the JET architecture to run on Junos devices, see the [gRPC Network Services User Guide](#).

1

CHAPTER

About JET APIs

[Overview of JET APIs | 2](#)

[JET Notification API | 4](#)

Overview of JET APIs

IN THIS SECTION

- [Notification API | 2](#)
- [Base Types APIs | 2](#)
- [Service APIs | 3](#)

JET provides APIs that extend the functionality of Junos OS and Junos OS Evolved. These APIs fall into three main categories: the notification API, base type APIs, and service APIs. The APIs are defined by proto definition files. You can view these files in multiple ways:

- View the full proto definition files on the [JET GitHub repository](#).
- To use the proto definition files, you will need to download and extract the IDL file for the release you are using. See [Set Up the JET VM](#) for instructions.

Notification API

The notification API provides interfaces that allow you to subscribe to events and designate a callback function to receive events when they occur. See ["JET Notification API" on page 4](#) for more information about this API.

Base Types APIs

- JNX Common Address API
- JNX Common Base Types API

Service APIs

Service APIs provide interfaces to access the control plane on the device and a management interface to run operational and configuration commands.

Routing

- BFD Service APIs
- BGP Route Service API
- ECMP Tracer Flow Monitoring APIs
- Flexible Tunnel Profile
- Flexible Tunnel Service
- MPLS Service API
- PRPD Common API
- PRPD Service API
- RIB Service API
- Routing Interface Service API

Firewall

Firewall Service API

Interfaces

Interfaces Service API

Infrastructure

Infrastructure Service APIs

Management

- JNX Authentication Service API
- JNX Management Service API

- JNX Registration Service API
- Versioning Service APIs

Network Address Translation (NAT)

MAP-E Service API

RELATED DOCUMENTATION

| [gRIBI](#)

JET Notification API

IN THIS SECTION

- [JSON Format of JET Notification Messages | 4](#)
- [Subscribing to Events | 5](#)
- [Programming JET Notification for Non-Python Languages | 7](#)

The JET Notification API allows you to subscribe to events and designate a callback function to receive events when they occur. These Python interfaces provide an interface to the MQTT notification system that, for languages other than Python, must be handled outside of JET (see "[Programming JET Notification for Non-Python Languages](#)" on page 7). For details about MQTT and Python, see <https://mosquitto.org/>.

JSON Format of JET Notification Messages

JET notification is delivered in JSON format. The JSON message has two parts: the header and the attributes. The header is common for all events. It contains the event ID, hostname, time, severity, and facility of the event. The attributes contain information about the event and vary depending event's topic name.

All kernel rtsock events will have `info` as the severity and `KERNEL` as the facility. For syslog events, the severity and facility will be same as that of the Junos OS syslog messages.

The following is an example event JSON file.

```
{
  "jet-event": {
    "event-id": "KERNEL_EVENT_IFD_ADD",
    "hostname": "mydevice",
    "time": "2016-01-07",
    "severity": "info",
    "facility": "KERNEL",
    "attributes": {
      "name": "ge-0/0/0",
      "snmp-id": 520,
      "flags": 8
    }
  }
}
```

See [Package JET Applications](#) for more information about JSON.

Subscribing to Events

Applications developed using JET can subscribe to the events listed in [Table 1 on page 5](#). A *topic* is an endpoint to which the clients connect. A topic acts as the central distribution hub for publishing and subscribing messages. Topics are simple, hierarchical strings, encoded in UTF-8, delimited by a forward slash.

Table 1: Junos Event Topics

Events	Topic	Event Information Returned
Physical Interface (IFD)	<ul style="list-style-type: none"> <code>/junos/events/kernel/interfaces/ifd/add/<i>ifdname</i></code> <code>/junos/events/kernel/interfaces/ifd/change/<i>ifdname</i></code> <code>/junos/events/kernel/interfaces/ifd/delete/<i>ifdname</i></code> 	name, snmp-id, flags

Table 1: Junos Event Topics (*Continued*)

Events	Topic	Event Information Returned
Logical Interface (IFL)	<ul style="list-style-type: none"> • /junos/events/kernel/interfaces/ifl/add/ <i>iflname</i> • /junos/events/kernel/interfaces/ifl/change/ <i>iflname</i> • /junos/events/kernel/interfaces/ifl/delete/ <i>iflname</i> 	name, subunit, snmp-id, flags
Family (IFF)	<ul style="list-style-type: none"> • /junos/events/kernel/interfaces/iff/add/ <i>iflname/family-type</i> • /junos/events/kernel/interfaces/iff/change/ <i>iflname/family-type</i> • /junos/events/kernel/interfaces/iff/delete/ <i>iflname/family-type</i> 	name, subunit, family, table-name, flags
Address	<ul style="list-style-type: none"> • /junos/events/kernel/interfaces/ifa/add/ <i>iflname/family-type/address</i> • /junos/events/kernel/interfaces/ifa/change/ <i>iflname/family-type/address</i> • /junos/events/kernel/interfaces/ifa/delete/ <i>iflname/family-type/address</i> 	name, subunit, family, local-address, destination-address, broadcast-address, flags
Firewall	<ul style="list-style-type: none"> • /junos/events/kernel/firewall/filter/add/ <i>filtername</i> • /junos/events/kernel/firewall/filter/change/ <i>filtername</i> • /junos/events/kernel/firewall/filter/delete/ <i>filtername</i> 	name, version, client-id, filter-type, protocol, interface-name, flags
Route	<ul style="list-style-type: none"> • /junos/events/kernel/route/add/ <i>family/prefix-with-length</i> • /junos/events/kernel/route/change/ <i>family/prefix-with-length</i> • /junos/events/kernel/route/delete/ <i>family/prefix-with-length</i> 	table-name, logical-router-name, address-family, route-type, route-prefix, arrayof(nexthop-address), flags

Table 1: Junos Event Topics (Continued)

Events	Topic	Event Information Returned
Route-table	<ul style="list-style-type: none"> • /junos/events/kernel/route-table/add/ <i>tablename/</i> <i>lname</i> • /junos/events/kernel/route-table/change/ <i>tablename/</i> <i>lname</i> • /junos/events/kernel/route-table/delete/ <i>tablename/</i> <i>lname</i> 	name, logical-router-name, address-family, flags
Syslog	/junos/events/syslog/ <i>event-id</i>	arrayof(attribute-value pairs)

Programming JET Notification for Non-Python Languages

Many of high-level languages have an MQTT library available. A JET application can use the corresponding library to connect to the MQTT broker running on Junos OS and subscribe to events. For example, here is a sample JAVA program connecting to an MQTT broker and subscribing to events: [Example MQTT Messaging in Java](#).

For details on example MQTT libraries for different languages, see [Table 2 on page 7](#).

Table 2: MQTT Libraries by Language

Language Name	License Link
C, Mosquitto	FreeBSD
C++, Mosquitto	FreeBSD
Python, Mosquitto	FreeBSD
Ruby, Ruby-mqtt	MIT License

Table 2: MQTT Libraries by Language *(Continued)*

Language Name	License Link
Java, Eclipse Paho Java	Eclipse Public License
Go, Eclipse Paho Go	Eclipse Public License
C#, MqttDotNet	Custom License

RELATED DOCUMENTATION[Set Up the JET VM](#)