Engineering
Simplicity

Junos® OS

Flow-Based and Packet-Based Processing User Guide for Security Devices

Published
2025-12-15

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

YEAR 2000 NOTICE

END USER LICENSE AGREEMENT

Table of Contents

About This Guide

Use this guide to configure and monitor the flow of traffic or packet, on a device using flow-based processing and packet-based forwarding. Also, for using an extensive set of flow-based security features which include policies, screens, network address translation (NAT), and other flow-based services.

1

CHAPTER

Overview

IN THIS CHAPTER

Traffic Processing on SRX Series Firewalls Overview

IN THIS SECTION

Junos OS for security devices integrates network security and routing capabilities of Juniper Networks. Packets that enter and exit a device undergo both packet-based and flow-based processing.

Understanding Traffic Processing on Security Devices

IN THIS SECTION

Junos OS for security devices integrates the world-class network security and routing capabilities of Juniper Networks. Junos OS includes a wide range of packet-based filtering, class-of-service (CoS) classifiers, and traffic-shaping features as well as a rich, extensive set of flow-based security features including policies, screens, network address translation (NAT), and other flow-based services.

Traffic that enters and exits a security device is processed according to features you configure, such as packet filters, security policies, and screens. For example, the software can determine:

- Whether the packet is allowed into the device
- Which firewall screens to apply to the packet
- The route the packet takes to reach its destination
- Which CoS to apply to the packet, if any
- Whether to apply NAT to translate the packet’s IP address
- Whether the packet requires an Application Layer Gateway (ALG)

Packets that enter and exit a device undergo both packet-based and flow-based processing:

- Flow-based packet processing treats related packets, or a stream of packets, in the same way. Packet treatment depends on characteristics that were established for the first packet of the packet stream, which is referred to as a flow.

  For the distributed processing architecture of the services gateway, all flow-based processing occurs on the SPU and sampling is multi-thread aware. Packet sequencing is maintained for the sampled packets.

- Packet-based, or stateless, packet processing treats packets discretely. Each packet is assessed individually for treatment.

  For the distributed processing architecture of the services gateway, some packet-based processing, such as traffic shaping, occurs on the NPU. Some packet-based processing, such as application of classifiers to a packet, occurs on the SPU.

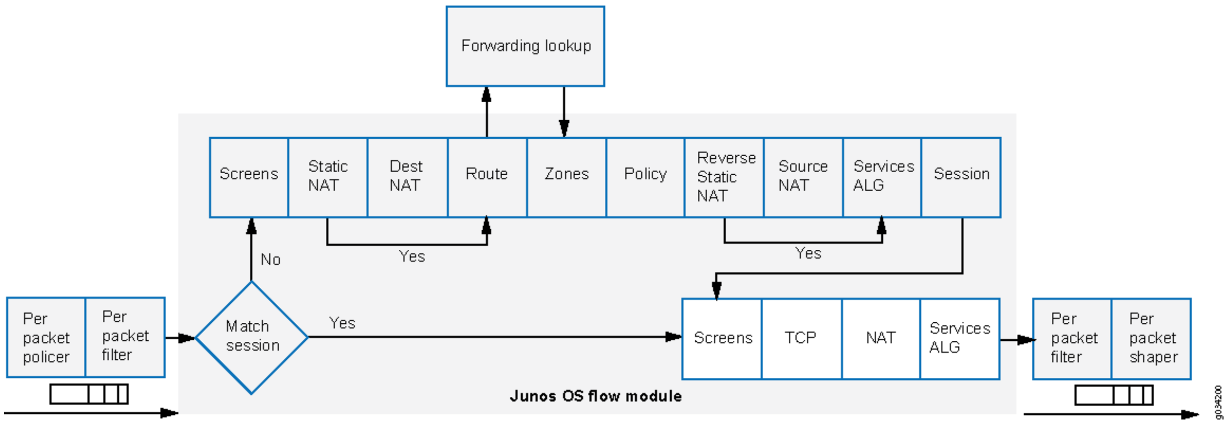
This topic includes the following sections:

Understanding Flow-Based Processing

A packet undergoes flow-based processing after packet-based filters and some screens have been applied to it. All flow-based processing for a single flow occurs on a single Services Processing Unit (SPU). An SPU processes the packets of a flow according to the security features and other services configured for the session.

Figure 1 on page 4 shows a conceptual view of how flow-based traffic processing occurs on services gateway.

Figure 1: Traffic Flow for Flow-Based Processing



A flow is a stream of related packets that meet the same matching criteria and share the same characteristics. Junos OS treats packets belonging to the same flow in the same manner.

Configuration settings that determine the fate of a packet—such as the security policy that applies to it, if it requires an Application Layer Gateway (ALG), if NAT is applied to translate the packet's source and/or destination IP address—are assessed for the first packet of a flow.

To determine if a flow exists for a packet, the NPU attempts to match the packet's information to that of an existing session based on the following match criteria:

- Source address
- Destination address
- Source port
- Destination port
- Protocol
- Unique session token number for a given zone and virtual router

Zones and Policies

The security policy to be used for the first packet of a flow is cached in a flow table for use with the same flow and closely related flows. Security policies are associated with zones. A zone is a collection of interfaces that define a security boundary. A packet's incoming zone, as determined by the interface through which it arrived, and its outgoing zone, as determined by the forwarding lookup, together determine which policy is used for packets of the flow.

Flows and Sessions

Flow-based packet processing, which is stateful, requires the creation of sessions. A session is created for the first packet of a flow for the following purposes:

- To store most of the security measures to be applied to the packets of the flow.
- To cache information about the state of the flow.

  For example, logging and counting information for a flow is cached in its session. (Some stateful firewall screens rely on threshold values that pertain to individual sessions or across all sessions.)
- To allocate required resources for the flow for features such as NAT.
- To provide a framework for features such as ALGs and firewall features.

Most packet processing occurs in the context of a flow, including:

- Management of policies, NAT, zones, and most screens.
- Management of ALGs and authentication.

Understanding Packet-Based Processing

A packet undergoes packet-based processing when it is removed from the queue on its input interface and before it is added to the queue on its output interface.

Packet-based processing applies stateless firewall filters, CoS features, and some screens to discrete packets.

- When a packet arrives at an interface, sanity checks, packet-based filters, some CoS features, and some screens are applied to it.
- Before a packet leaves the device, any packet-based filters, some CoS features, and some screens associated with the interface are applied to the packet.

Filters and CoS features are typically associated with one or more interfaces to influence which packets are allowed to transit the system and to apply special actions to packets as necessary.

The following topics describe the kinds of packet-based features that you can configure and apply to transit traffic.

Stateless Firewall Filters

Also referred to as access control lists (ACLs), stateless firewall filters control access and limit traffic rates. They statically evaluate the contents of packets transiting the device from a source to a destination, or packets originating from or destined for the Routing Engine. A stateless *firewall filter* evaluates every packet, including fragmented packets.

You can apply a stateless firewall filter to an input or output interface, or to both. A filter contains one or more terms, and each term consists of two components—match conditions and actions. By default, a packet that does not match a firewall filter is discarded.

You can plan and design stateless firewall filters to be used for various purposes—for example, to limit traffic to certain protocols, IP source or destination addresses, or data rates. Stateless firewall filters are executed on the NPU.

Class-of-Service Features

CoS features allow you to classify and shape traffic. CoS features are executed on the NPU.

- Behavior aggregate (BA) classifiers—These classifiers operate on packets as they enter the device. Using behavior aggregate classifiers, the device aggregates different types of traffic into a single forwarding class to receive the same forwarding treatment. BA classifiers allow you to set the forwarding class and loss priority of a packet based on the Differentiated Service (DiffServ) value.

- Traffic shaping—You can shape traffic by assigning service levels with different delay, *jitter*, and packet loss characteristics to particular applications served by specific traffic flows. Traffic shaping is especially useful for real-time applications, such as voice and video transmission.

Screens

Some screens, such as denial-of-service (DoS) screens, are applied to a packet outside the flow process. They are executed on the Network Processing Unit (NPU).

Understanding the Default Processing Behavior for IPv4 Traffic

IN THIS SECTION

Flow-based processing mode is required for security features such as zones, screens, and firewall policies to function. For drop mode processing, the traffic is dropped directly, it is not forwarded. It differs from packet-mode processing for which the traffic is handled but no security processes are applied.

Use Feature Explorer to confirm platform and release support for specific features.

Review the "Understanding the Default Processing Behavior for IPv4 Traffic" on page 6 section for notes related to your platform.

Configuring an SRX Series Device as a Border Router

When an SRX Series Firewall of any type is enabled for flow-based processing or drop mode, to configure the device as a border router you must change the mode to packet-based processing for MPLS. In this case, to configure the SRX Series Firewall to packet mode for MPLS, use the `set security forwarding-options family mpls mode packet-based` statement.

Platform-Specific IPv4 Traffic Behavior

Use Feature Explorer to confirm platform and release support for specific features.

Use the following table to review platform-specific behavior for your platform:

Platform	Difference
SRX Series Firewall	• On SRX300 series devices that support IPv4 traffic, you must reboot the device when you switch between flow mode, packet mode, and drop mode. For the SRX300 series devices that support IPv4 traffic, the default processing mode is set to drop mode because of memory constraints. In this case, you must reboot the device after changing the processing mode from the drop mode default to flow-based processing mode or packet-based processing mode—that is, between modes on these devices. • On SRX4100, SRX4200, SRX5400, SRX5600, SRX5800, and vSRX Virtual Firewall devices that support IPv4 traffic, you do not need to reboot the device when you switch between flow mode, packet mode, and drop mode.

Understanding Traffic Processing on SRX320 Devices

IN THIS SECTION

This topic describes the process that the SRX320 Services Gateways undertake in establishing a session for packets belonging to a flow that transits the device. The flow services of the SRX320 devices are single-threaded and non-distributed. Although they differ from the other SRX Series Firewalls in this respect, the same flow model is followed and the same command line interface (CLI) is implemented.

To illustrate session establishment and the packet “walk” including the points at which services are applied to the packets of a flow, the example described in the following sections uses the simple case of a unicast session:

Understanding Flow Processing and Session Management

This topic explains how a session is set up to process the packets composing a flow. In the following topic, the SPU refers to the data plane thread of the SRX320 Firewall.

At the outset, the data plane thread fetches the packet and performs basic sanity checks on it. Then it processes the packet for stateless filters and CoS classifiers and applies some screens.

Understanding First-Packet Processing

To determine if a packet belongs to an existing flow, the device attempts to match the packet’s information to that of an existing session based on the following six match criteria:

- Source address
- Destination address
- Source port
- Destination port
- Protocol

- Unique token from a given zone and virtual router

The SPU checks its session table for an existing session for the packet. If no existent session is found, the SPU sets up a session for the flow. If a session match is found, the session has already been created, so the SPU performs fast-path processing on the packet.

Understanding Session Creation

In setting up the session, the SPU executes the following services for the packet:

- Screens
- Route lookup
- Policy lookup
- Service lookup
- NAT, if required

After a session is set up, it is used for all packets belonging to the flow. Packets of a flow are processed according to the parameters of its session. For the remainder of the steps entailed in packet processing, proceed to Step 1 in “Fast-Path Processing”. All packets undergo fast-path processing.

Understanding Fast-Path Processing

If a packet matches a session, Junos OS performs fast-path processing as described in the following steps. After a session has been set up for the first packet in a flow, also undergoes fast-path processing. All packets undergo fast-path processing.

1. The SPU applies flow-based security features to the packet.
   - Configured screens are applied.
   - TCP checks are performed.
   - Flow services, such as NAT, ALG, and IPsec are applied, if required.
2. The SPU prepares the packet for forwarding and transmits it.
   - Routing packet filters are applied.
   - Traffic shaping is applied.
   - Traffic prioritizing is applied.
   - Traffic scheduling is applied.

- The packet is transmitted.

Understanding Traffic Processing on SRX4600 Devices

IN THIS SECTION

The Juniper Networks SRX4600 Firewall integrates flow-based security and routing services, including advanced security and threat mitigation and traditional stateful firewall security. The Junos OS flow-based infrastructure provides the foundation and framework for Layer 4 through Layer 7 application-based services. The SRX4600 Firewall is designed to be deployed as an integrated firewall at the large enterprise data center edge and data center core, and the campus edge. It can also be deployed as an LTE security gateway and a Gi/SGi firewall.

This topic includes the following content:

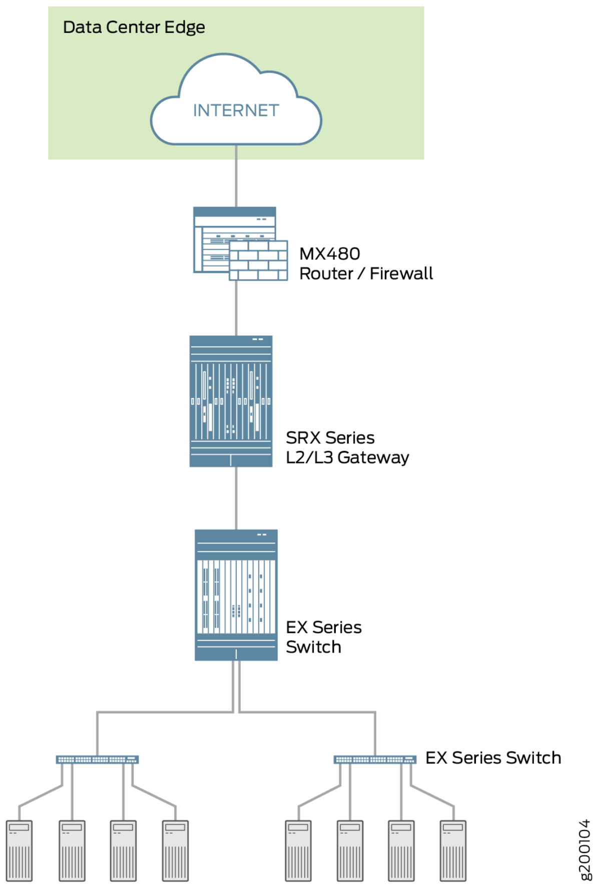
Understanding Deployment Scenarios for the SRX4600 Firewall and Its Features

The SRX4600 Firewall can be deployed in many areas to secure your environment and its resources. It is often used to protect the data center edge and core in the following ways:

- Deploying the SRX4600 Firewall as a Data Center Edge Firewall

  You can deploy the SRX4600 Firewall at the edge of your data center to provide the applications and services that it hosts with optimum protection. Every data center has an ingress point to allow clients access to the data center's services, but malicious aggressors can take advantage of it to launch attacks against these services. A large amount of traffic coming into the data center is ingress internet traffic. For that reason alone, deploying robust, multi-layered security at the data center edge is essential. The SRX4600 Firewall effectively and reliantly blocks attacks, and it allows you to configure the system to thwart specific kinds of attacks. The SRX4600 Firewall supports Juniper's Software-Defined Secure Network (SDSN) framework, including Juniper Advanced Threat Prevention Cloud (ATP Cloud), which is built around automated and actionable intelligence that can be shared quickly to recognize and mitigate threats. Figure 2 shows the SRX4600 Firewall deployed at the data center edge in conjunction with an MX480 router and EX Series switches.

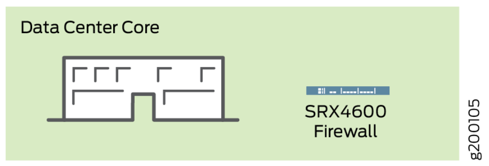
Figure 2: Deploying the SRX4600 Firewall at the Data Center Edge



- Deploying the SRX4600 Firewall at the Data Center Core

  You can deploy the SRX4600 Firewall at the data center core to provide enhanced security and to ensure that compliance requirements are met. Data center processing has become increasingly dynamic necessitating clear network definition and compliance requirements enforcement. To ensure compliance, you can use the SRX4600 Firewall to segment your overall network into individual server networks and secure traffic within them. The SRX4600 Firewall provides high availability and automation, and its high performance Layer 3 and Layer 4 services meet the security requirements of the data center core. Figure 3 shows the SRX4600 Firewall deployed as a multi-layered firewall at the data center core.

Figure 3: Deploying the SRX4600 Firewall at the Data Center Core



In addition to its advanced anti-malware features, the SRX4600 Firewall supports the following features:

- Stateful firewall
- Application security suite
- Content Security (Sophos AV, Web filtering, antispam)
- IDP
- High availability (Chassis cluster)
    - Dual HA control ports (10G)
    - MACsec support for HA ports
- Ethernet interfaces through QSFP28 (100G/40G/4x10G modes), QSFP+ (40G/4x10G modes) and SFP+ (10G mode)
- IPsec VPN, including AutoVPN and Group VPNv2
- QoS and network services
- J-Web
- Routing policies with multicast

Flow-Based Processing and Session Fundamentals

To understand flow processing on the SRX4600 Firewall, it is important to understand the fundamentals of flow.

A *flow* is a stream of related packets that meet the same matching criteria and share the same characteristics. Junos OS treats packets that belong to the same flow in the same way. The architecture of an SRX Series services gateway and how it handles packet flows are tightly coupled. Consequently, in part, flow is implemented differently across the family of SRX Series Firewalls because of their architectural differences.

Flow-based packet processing, which is stateful, requires the creation of *sessions*. Sessions are created based on routing and other traffic classification information to store information and allocate resources for a flow. Sessions cache information about the state of the flow, and they store most of the security measures to be applied to packets of the flow. Because of the architectural differences across devices, sessions are also managed differently by different devices.

Regardless of these differences, conceptually the flow process is the same across all services gateways, and sessions serve the same purposes and have the same features.

Flow and Session Underlying Components Implemented Across SRX Series Firewalls

SRX Series Firewalls use the same infrastructure components to support flow and manage sessions, but not all devices implement all of them.

To understand flow, it is essential to understand the following components and how they are used:

- The Services Processing Unit (SPU)

  An SPU manages the session for a packet flow. It applies security features and other services to the packet. It also applies packet-based stateless firewall filters, classifiers, and traffic shapers to the packet.

- The central point (CP)

  The central point is an SPU that the system uses to allocate resources and distribute session management among SPUs. When the first packet of a flow is processed, the central point determines which SPU to use for that packet's session. The SRX4600 Firewall does not implement a central point.

- The Network Processing Unit (NPU) and the Network Processing session

  An NPU is a processor that runs on an I/O card (IOC) and processes packets discretely. When a flow is created, subsequent packets of the flow are matched to the session on the NPU. The NPU handles additional processing such as TCP sequence check, time-to-live (TTL) processing, and Layer 2 header translation. An NPU improves performance in that extra packet forwarding between a session-SPU and a hash-SPU is avoided. The SRX4600 Firewall implements an NPU.

The SRX4600 Firewall flow architecture has been improved to optimize use of the SRX4600 device's advanced multi-core Xeon™ Processors. The SRX4600 Firewall implements the use of a dedicated session thread to circumvent problems such as management of out-of-order packets in a flow. It utilizes the network processing session to ensure that packets are forwarded to the right, dedicated thread. Packets are distributed to different threads in accord with the hash-based session distribution model.

Understanding Traffic Processing on SRX5000 Line Devices

IN THIS SECTION

Junos OS on SRX5000 devices is a distributed, parallel processing, high-throughput and high-performance system. The distributed parallel processing architecture of the SRX5000 line of services gateways includes multiple processors to manage sessions and run security and other services processing. This architecture provides greater flexibility and allows for high throughput and fast performance.

NOTE: In SRX1400, SRX3400, SRX3600, SRX5400, SRX5600, and SRX5800 devices, IKE negotiations involving NAT traversal do not work if the IKE peer is behind a NAT device that will change the source IP address of the IKE packets during the negotiation. For example, if the NAT device is configured with DIP, it changes the source IP because the IKE protocol switches the UDP port from 500 to 4500.

The I/O cards (IOCs) and Services Processing Cards (SPCs) on SRX5000 line devices contain processing units that process a packet as it traverses the device. An IOC has one or more Network Processing Units (NPUs), and a SPC has one or more Services Processing Units (SPUs).

These processing units have different responsibilities. All flow-based services for a packet are executed on a single SPU. The responsibilities of these NPUs are not clearly delineated in regard to the other kind of services that run on them. .)

For example:

- An NPU processes packets discretely. It performs sanity checks and applies some screens that are configured for the interface, such as denial-of-service (DoS) screens, to the packet.

- An SPU manages the session for the packet flow and applies security features and other services to the packet. It also applies packet-based stateless firewall filters, classifiers, and traffic shapers to the packet.
- An NPU forwards a packet to the SPU using the hash algorithm. However, for some applications, like ALG, the system will need to query the application central point to determine on which SPU the packet should be processed.

These discrete, cooperating parts of the system, including the central point, each store the information identifying whether a session exists for a stream of packets and the information against which a packet is matched to determine if it belongs to an existing session.

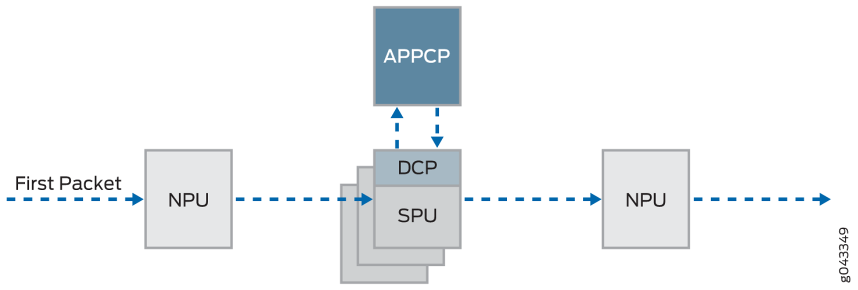
This architecture allows the device to distribute processing of all sessions across multiple SPUs. It also allows an NPU to determine if a session exists for a packet, to check the packet, and to apply screens to the packet. How a packet is handled depends on whether it is the first packet in a flow.

The following sections describe the processing architecture using SRX5400, SRX5600, and SRX5800 devices as an example:

Understanding First-Packet Processing

Figure 4 on page 15 illustrates the path the first packet in a flow takes as it enters the device—the NPU determines that no session exists for the packet, and the NPU sends the packet to the distributed central point to set up a distributed central point session. The distributed central point then sends a message to the application central point to select the SPU to set up a session for the packet and to process the packet. The distributed central point then sends the packet to that SPU. The SPU processes the packet and sends it to the NPU for transmission from the device. (This high-level description does not address application of features to a packet.)

Figure 4: First-Packet Processing



After the first packet in a flow has traversed the system and a session has been established for it, it undergoes fast-path processing.

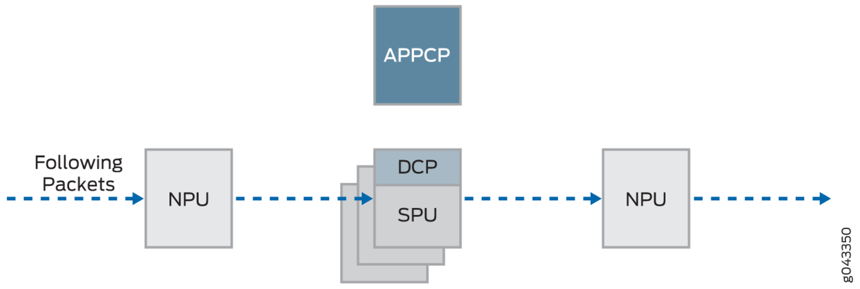
Subsequent packets in the flow also undergo fast-path processing; in this case, after each packet enters the session and the NPU finds a match for it in its session table, the NPU forwards the packet to the SPU that manages its session.

Figure 5 on page 16 illustrates fast-path processing. This is the path a packet takes when a flow has already been established for its related packets. (It is also the path that the first packet in a flow takes after the session for the flow that the packet initiated has been set up.) After the packet enters the device, the NPU finds a match for the packet in its session table, and it forwards the packet to the SPU that manages the packet's session. Note that the packet bypasses interaction with the central point.

Understanding Fast-Path Processing

The following section explains how a session is created and the process a packet undergoes as it transits the device.

Figure 5: Fast-Path Processing



Here is an overview of the main components involved in setting up a session for a packet and processing packets both discretely and as part of a flow as they transit the SRX5400, SRX5600, and SRX5800 devices.

- Network Processing Units (NPUs)—NPUs reside on IOCs. They handle packet sanity checking and application of some screens. NPUs maintain session tables that they use to determine if a session exists for an incoming packet or for reverse traffic.

  The NPU session table contains an entry for a session if the session is established on an SPU for a packet that had previously entered the device via the interface and was processed by this NPU. The SPU installs the session in the NPU table when it creates the session.

  An NPU determines if a session exists for a packet by checking the packet information against its session table. If the packet matches an existing session, the NPU sends the packet and the metadata for it to the SPU. If there is no session, the NPUs sends the packet to one SPU which is calculated using the hash algorithm.

- Services Processing Units (SPUs)—The main processors of the SRX5400, SRX5600, and SRX5800 devices reside on SPCs. SPUs establish and manage traffic flows and perform most of the packet processing on a packet as it transits the device. Each SPU maintains a hash table for fast session lookup. The SPU applies stateless firewall filters, classifiers, and traffic shapers to traffic. An SPU performs all flow-based processing for a packet and most packet-based processing. Each multicore SPU processes packets independently with minimum interaction among SPUs on the same or different SPC. All packets that belong to the same flow are processed by the same SPU.

  The SPU maintains a session table with entries for all sessions that it established and whose packets it processes. When an SPU receives a packet from an NPU, it checks its session table to ensure that the packet belongs to it. It also checks its session table when it receives a packet from the distributed central point and sends a message to establish a session for that packet to verify that there is not an existing session for the packet.

- Central point—The central point architecture is divided into two modules, the application central point and the distributed central point. The application central point is responsible for global resource management and loading balancing, while the distributed central point is responsible for traffic identification (global session matching). The application central point functionality runs on the dedicated central point SPU, while the distributed central point functionality is distributed to the rest of the SPUs. Now the central point sessions are no longer on the dedicated central point SPU, but with the distributed central point on other flow SPUs.

- Routing Engine—The Routing Engine runs the control plane.

Understanding the Data Path for Unicast Sessions

This section describes the process of establishing a session for packets belonging to a flow that transits the device.

To illustrate session establishment and the packet “walk” including the points at which services are applied to the packets in a flow, this example uses the simple case of a unicast session.

This packet “walk” brings together the packet-based processing and flow-based processing that Junos OS performs on the packet.

Session Lookup and Packet-Match Criteria

To determine if a packet belongs to an existing flow, the device attempts to match the packet’s information to that of an existing session based on the following six match criteria:

- Source address

- Destination address

- Source port

- Destination port
- Protocol
- Unique token from a given zone and virtual router

Understanding Session Creation: First-Packet Processing

This section explains how a session is set up to process the packets composing a flow. To illustrate the process, this section uses an example with a source “a” and a destination “b”. The direction from source to destination for the packets of the flow is referred to as (a ->b). The direction from destination to source is referred to as (b->a).

Step 1. A Packet Arrives at an Interface on the Device And the NPU Processes It.

This section describes how a packet is handled when it arrives at an SRX Series Firewall ingress IOC.

1. The packet arrives at the device’s IOC and is processed by the NPU on the IOC.
2. The NPU performs basic sanity checks on the packet and applies some screens configured for the interface to the packet.
3. The NPU checks its session table for an existing session for the packet. (It checks the packet’s tuple against those of packets for existing sessions in its session table.)
    a. If no existing session is found, the NPU forwards the packet to the hash SPU.
    b. If a session match is found, the session has already been created on an SPU that was assigned to it, so the NPU forwards the packet to the SPU for processing along with the session ID.

Example: Packet (a ->b) arrives at NPU1. NPU1 performs sanity checks and applies DoS screens to the packet. NPU1 checks its session table for a tuple match, and no existing session is found. NPU1 forwards the packet to an SPU.

Step 2. The Distributed Central Point Creates a Session with a "Pending” State.

When an NPU receives a packet, the NPU send it to the distributed central point, based on the hash algorithm. The distributed central point then looks up the distributed central point session table and creates an entry if needed.

This process entails the following parts:

1. The distributed central point checks its session table to determine if a session exists for the packet received from the NPU. (An NPU forwards a packet to the distributed central point because it cannot find an existing session for the packet)

2. If there is no entry that matches the packet in the distributed central point session table, the distributed central point creates a pending wing for the session. The distributed central point then sends a query message to the application central point to select an SPU to be used for the session.

3. On receiving the query message, the application central point checks its gate table to determine if a gate exists for the packet. If a gate is matched or some other session distribution algorithm is triggered, the application central point selects another SPU to process the packet; otherwise, the SPU (that is, the distributed central point SPU) is selected. Finally, the application central point sends a query response to the distributed central point.

4. On receiving the query response, the distributed central point forwards the first packet in flow to the selected SPU in a message directing the SPU to set up a session locally to be used for the packet flow. For example, the distributed central point creates a pending wing (a ->b) for the session. The application central point selects SPU1 to be used for it. The distributed central point sends SPU1 the (a->b) packet along with a message to create a session for the distributed central point.

Example: The distributed central point creates a pending wing (a ->b) for the session. It selects SPU1 to be used for it. It sends SPU1 the (a->b) packet along with a message to create a session for it.

Step 3. The SPU Sets Up the Session.

Each SPU, too, has a session table, which contains information about its sessions. When the SPU receives a message from the distributed central point to set up a session, it checks its session table to ensure that a session does not already exist for the packet.

1. If there is no existing session for the packet, the SPU sets up the session locally.

2. The SPU sends a message to the distributed central point directing it to install the session.

> **NOTE**: During first-packet processing, if NAT is enabled, the SPU allocates IP address resources for NAT. In this case, the first-packet processing for the session is suspended until the NAT allocation process is completed.

The SPU adds to the queue any additional packets for the flow that it might receive until the session has been installed.

Example: SPU1 creates the session for (a ->b) and sends a message back to the distributed central point directing it to install the pending session.

Step 4. The Distributed Central Point Installs the Session.

The distributed central point receives the install message from the SPU.

1. The distributed central point sets the state for the session’s pending wing to active.

2. The distributed central point installs the reverse wing for the session as an active wing.

> **NOTE**: For some cases, such as NAT, the reverse wing may be installed on a different distributed central point from the init wing distributed central point.

3. It sends an acknowledge (ACK) message to the SPU, indicating that the session is installed.

Example: The distributed central point receives a message from SPU1 to install the session for the (a->b) wing. It sets the session state for the (a->b) wing to active. It installs the reverse wing (b->a) for the session and makes it active; this allows for delivery of packets from the reverse direction of the flow: destination (b) to be delivered to the source (a).

Step 5. The SPU Sets Up the Session on the Ingress and Egress NPUs.

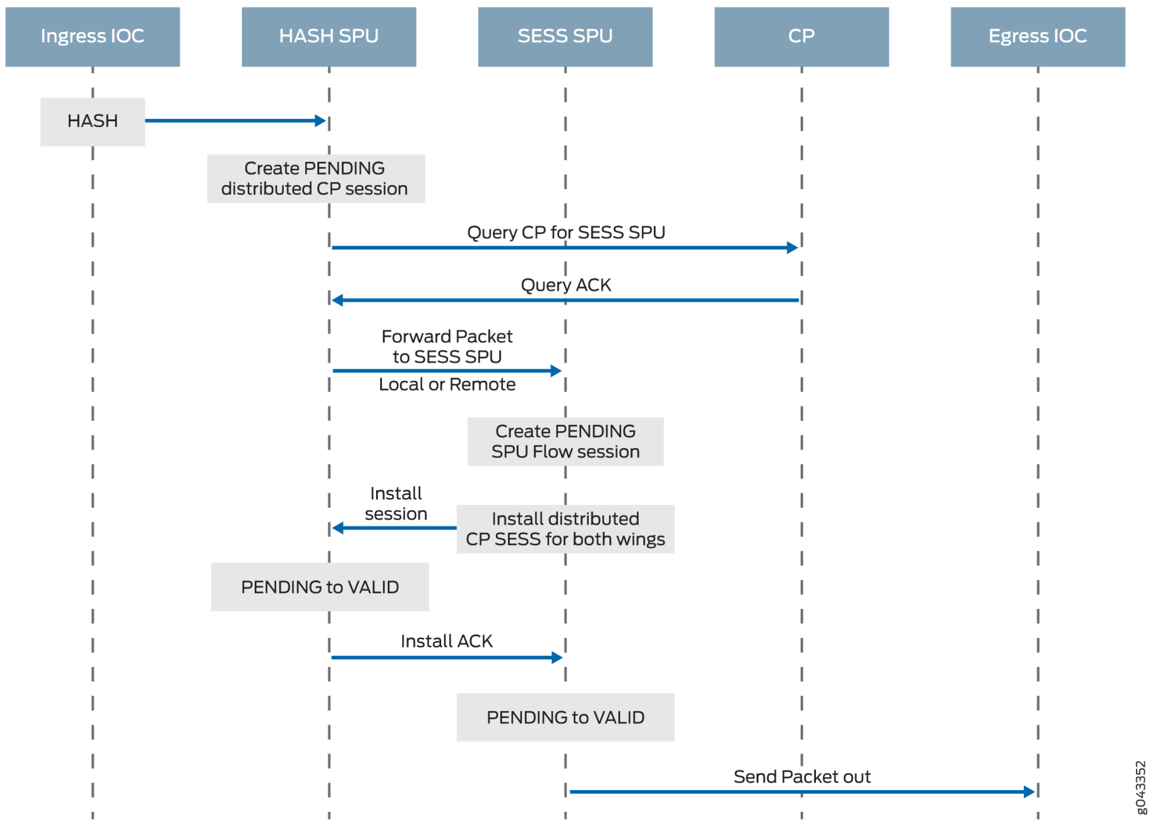
NPUs maintain information about a session for packet forwarding and delivery. Session information is set up on the egress and ingress NPUs (which sometimes are the same) so that packets can be sent directly to the SPU that manages their flows and not to the distributed central point for redirection.

Step 6. Fast-Path Processing Takes Place.

For the remainder of the steps entailed in packet processing, proceed to Step 1 in "Understanding Fast-Path Processing" on page 21.

Figure 6 on page 21 illustrates the first part of the process that the first packet in a flow undergoes after it reaches the device. At this point a session is set up to process the packet and the rest of the packets belonging to its flow. Subsequently, it and the rest of the packets in the flow undergo fast-path processing.

Figure 6: Session Creation: First-Packet Processing



Understanding Fast-Path Processing

All packets undergo fast-path processing. However, if a session exists for a packet, the packet undergoes fast-path processing and bypasses the first-packet process. When there is already a session for the packet's flow, the packet does not transit the central point.

Here is how fast-path processing works: NPUs at the egress and ingress interfaces contain session tables that include the identification of the SPU that manages a packet's flow. Because the NPUs have this session information, all traffic for the flow, including reverse traffic, is sent directly to that SPU for processing.

To illustrate the fast-path process, this section uses an example with a source "a" and a destination "b". The direction from source to destination for the packets of the flow is referred to as (a->b). The direction from destination to source is referred to as (b->a).

Step 1. A Packet Arrives at the Device and the NPU Processes It.

This section describes how a packet is handled when it arrives at a services gateway's IOC.

1. The packet arrives at the device's IOC and is processed by the NPU on the card.

   The NPU performs sanity checks and applies some screens, such as denial-of-service (DoS) screens, to the packet.

2. The NPU identifies an entry for an existing session in its session table that the packet matches.

3. The NPU forwards the packet along with metadata from its session table, including the session ID and packet tuple information, to the SPU that manages the session for the flow, applies stateless firewall filters and CoS features to its packets, and handles the packet's flow processing and application of security and other features.

Example: Packet (a ->b) arrives at NPU1. NPU1 performs sanity checks on the packet, applies DoS screens to it, and checks its session table for a tuple match. It finds a match and that a session exists for the packet on SPU1. NPU1 forwards the packet to SPU1 for processing.

Step 2. The SPU for the Session Processes the Packet.

Most of a packet's processing occurs on the SPU to which its session is assigned. The packet is processed for packet-based features such as stateless firewall filters, traffic shapers, and classifiers, if applicable. Configured flow-based security and related services such as firewall features, NAT, ALGs, and so on, are applied to the packet. (For information on how security services are determined for a session.

1. Before it processes the packet, the SPU checks its session table to verify that the packet belongs to one of its sessions.

2. The SPU processes the packet for applicable features and services.

Example: SPU1 receives packet (a->b) from NPU1. SPU1 checks its session table to verify that the packet belongs to one of its sessions. Then it processes packet (a ->b) according to input filters and CoS features that apply to its input interface. The SPU applies the security features and services that are configured for the packet's flow to it, based on its zone and policies. If any are configured, it applies output filters, traffic shapers and additional screens to the packet.

Step 3. The SPU Forwards the Packet to the NPU.

1. The SPU forwards the packet to the NPU.

2. The NPU applies any applicable screens associated with the interface to the packet.

Example: SPU1 forwards packet (a ->b) to NPU2, and NPU2 applies DoS screens.

Step 4. The Interface Transmits the Packet from the Device.

Example: The interface transmits packet (a->b) from the device.

Step 5. A Reverse Traffic Packet Arrives at the Egress Interface and the NPU Processes It.

This step mirrors Step 1 exactly in reverse. See Step 1 in this section for details.

Example: Packet (b->a) arrives at NPU2. NPU2 checks its session table for a tuple match. It finds a match and that a session exists for the packet on SPU1. NPU2 forwards the packet to SPU1 for processing.

Step 6. The SPU for the Session Processes the Reverse Traffic Packet.

This step is the same as Step 2 except that it applies to reverse traffic. See Step 2 in this section for details.

Example: SPU1 receives packet (b->a) from NPU2. It checks its session table to verify that the packet belongs to the session identified by NPU2. Then it applies packet-based features configured for the NPU1's interface to the packet. It processes packet (b->a) according to the security features and other services that are configured for its flow, based on its zone and policies.

Step 7. The SPU Forwards the Reverse Traffic Packet to the NPU.

This step is the same as Step 3 except that it applies to reverse traffic. See Step 3 in this section for details.

Example: SPU1 forwards packet (b->a) to NPU1. NPU1 processes any screens configured for the interface.

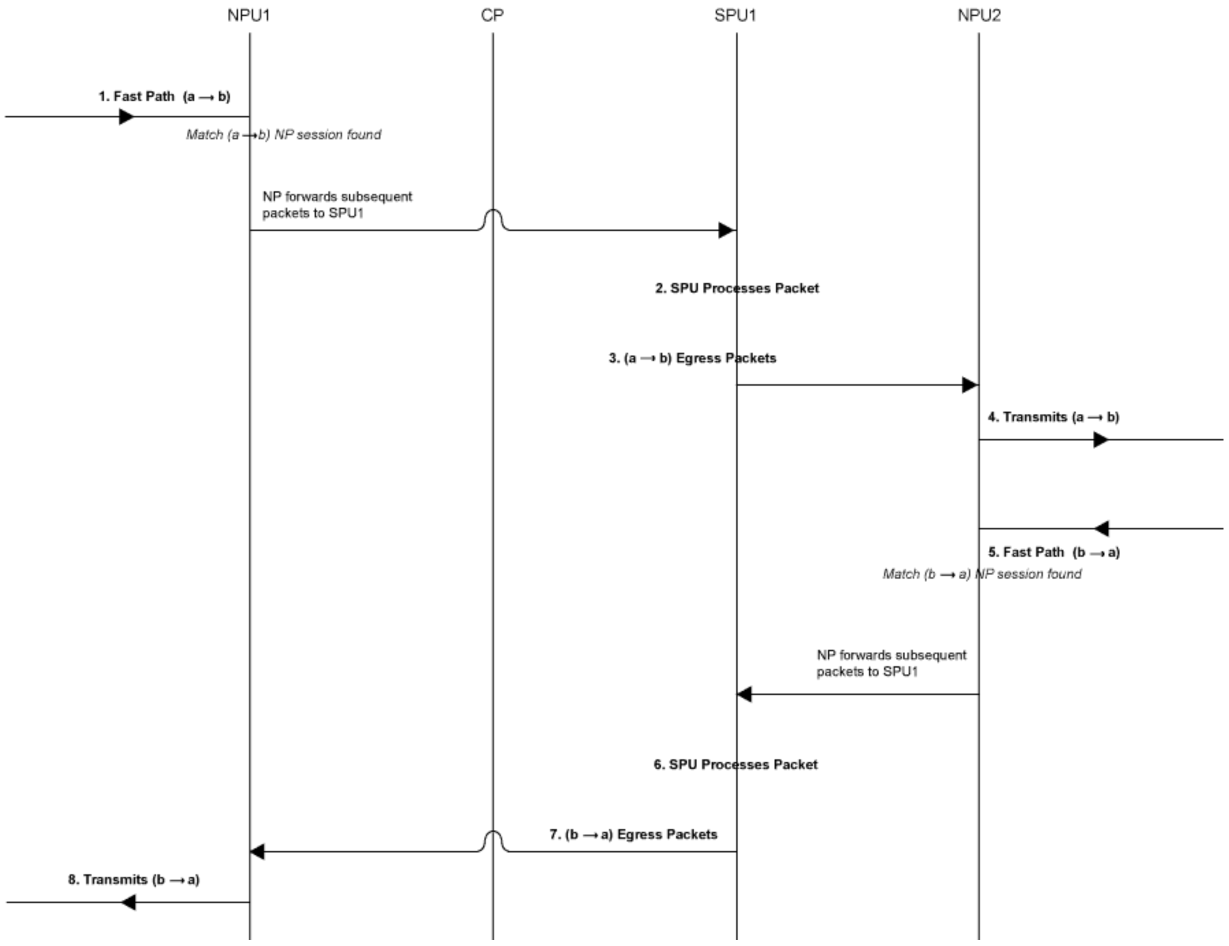
8. The Interface Transmits the Packet from the Device.

This step is the same as Step 4 except that it applies to reverse traffic. See Step 4 in this section for details.

Example: The interface transmits packet (b->a) from the device.

Figure 7 on page 24 illustrates the process a packet undergoes when it reaches the device and a session exists for the flow that the packet belongs to.

Figure 7: Packet Walk for Fast-Path Processing


NPU1
CP
SPU1
NPU2
1. Fast Path (a → b)
Match (a → b) NP session found
NP forwards subsequent packets to SPU1
2. SPU Processes Packet
3. (a → b) Egress Packets
4. Transmits (a → b)
5. Fast Path (b → a)
Match (b → a) NP session found
NP forwards subsequent packets to SPU1
6. SPU Processes Packet
7. (b → a) Egress Packets
8. Transmits (b → a)

Understanding Services Processing Units

For a given physical interface, the SPU receives ingress packets from all network processors in the network processor bundle associated with the physical interface. The SPU extracts network processor bundle information from the physical interface and uses the same 5-tuple hash algorithm to map a flow to a network processor index. To determine the network processor, the SPU does a lookup on the network processor index in the network processor bundle. The SPU sends egress packets to the physical interface's local Physical Interface Module (PIM) for the outward traffic.

NOTE: The network processor and the SPU use the same 5-tuple hash algorithm to get the hash values for the packets.

Understanding Scheduler Characteristics

For SRX5400, SRX5600, and SRX5800 devices, the IOC supports the following hierarchical scheduler characteristics:

- IFL – The configuration of the network processor bundle is stored in the physical interface data structure. For example, SRX5400, SRX5600, and SRX5800 devices have a maximum of 48 PIMs. The physical interface can use a 48-bit bit-mask to indicate the PIM, or the network processor traffic from this physical interface is distributed in addition to the physical interface's primary network processor.

  On SRX5000 line devices, the iflset functionality is not supported for aggregated interfaces like *reth*.

- IFD – The *logical interface* associated with the physical interface of a network processor bundle is passed to all the IOCs that have a PIM in the network processor bundle.

Understanding Network Processor Bundling

The network processor bundling feature is available on SRX5000 line devices. This feature enables distribution of data traffic from one interface to multiple network processors for packet processing. A primary network processor is assigned for an interface that receives the ingress traffic and distributes the packets to several other secondary network processors. A single network processor can act as a primary network processor or as a secondary network processor to multiple interfaces. A single network processor can join only one network processor bundle.

Network Processor Bundling Limitations

Network processor bundling functionality has the following limitations:

- Network processor bundling allows a total of 16 PIMs per bundle and 8 different network processor bundle systems.
- You need to reboot the device to apply the configuration changes on the bundle.
- Network processor bundling is below the reth interface in the overall architecture. You can choose one or both interfaces from the network processor bundle to form the reth interface.
- If the IOC is removed from a network processor bundle, the packets forwarded to the PIM on that IOC are lost.
- When the network processor bundle is enabled, the ICMP, UDP, and TCP sync flooding thresholds no longer apply to an interface. Packets are distributed to multiple network processors for processing. These thresholds apply to each network processor in the network processor bundle.
- Network processor bundling is not supported in Layer 2 mode.

- Because of memory constraints on the network processor, the number of network processor bundled ports that are supported per PIM is limited. Within the network processor bundle, each port needs to have a global port index. The global port index is calculated using the following formula:

  Global_port_index = (global_pic * 16) + port_offset

- Link aggregation groups (LAGs) and redundant Ethernet interface LAGs in *chassis cluster* implementations can coexist with network processor bundling. However, neither LAGs nor redundant Ethernet interface LAGs can overlap with or share physical links with a network processor bundle.

Understanding Session Cache

IN THIS SECTION

Overview

The SRX5K-MPC (IOC2), SRX5K-MPC3-100G10G (IOC3), and SRX5K-MPC3-40G10G (IOC3) on SRX5400, SRX5600, and SRX5800 devices support session cache and selective installation of the session cache.

Session cache is used to cache a conversation between the network processor (NP) and the SPU on an IOC. A conversation could be a session, GTP-U tunnel traffic, IPsec VPN tunnel traffic, and so on. A conversation has two session cache entries, one for incoming traffic and the other for reverse traffic. Depending on where the traffic ingress and egress ports are, two entries might reside in the same network processor or in different network processors. IOCs support session cache for IPv6 sessions.

A session cache entry is also called a *session wing*.

Session cache on the IOC leverages Express Path (formerly known as *services offloading*) functionality and helps prevent issues such as high latency and IPsec performance drop.

A session cache entry records:

- To which SPU the traffic of the conversion should be forwarded
- To which egress port the traffic of the conversion should be forwarded in Express Path mode
- What processing to do for egress traffic, for example, NAT translation in Express Path mode

Other traffic was hashed to SPUs based on their 5-tuple key information. VPN traffic employed the concept of the anchored SPU, which did not necessarily coincide with the functions of the flow SPU. The network processor could only forward the packets to the flow SPU based on the 5-tuple hash. The flow SPU then forwarded the packet to the anchored SPU. This created an extra hop for VPN traffic, which wasted the switch fabric bandwidth and reduced the VPN throughput roughly by half. This performance reduction occurred because the traffic still had to go back to the flow SPU after processing on the anchored SPU.

The session cache table is now extended on IOC to support the NP sessions. Express Path traffic and NP traffic share the same session cache table on IOCs. Express Path traffic is forwarded by the IOC itself either locally or to another IOC, because the traffic does not require any services from the SPU. NP traffic is forwarded to the SPU specified in the session cache for further processing. All the session cache entries are shared by both Express Path session traffic and NP traffic.

To enable session cache on the IOCs you need to run the `set chassis fpc` *`<fpc-slot>`* `np-cache` command.

NOTE: The IOC2 and the IOC3 utilize the delay sessions delete mechanism. The same sessions (sessions with the same five tuples) that are deleted and then reinstalled immediately are not cached on the IOCs.

Selective Session Cache Installation

To avoid high latency, improve IPSec performance, and to better utilize the valuable resources, certain priority mechanisms are applied to both flow module and the IOC.

The IOCs maintain and monitor session cache usage threshold levels. The IOCs also communicate the session cache usage to the SPU, so that when a certain session cache usage threshold is reached, the SPU only sends session cache installation requests for selective high-priority traffic sessions.

Applications like IDP, ALG need to process packets in order. One SPU has multiple flow threads to handle packets belong to one session, load balancing thread (LBT), and packet-ordering thread (POT) packet order can make sure traffic pass through firewall in order, it cannot guarantee application to process packets that belong to same session in order. Flow serialization provides the method that only one SPU flow thread processing packets belong to the same session at one time, so applications can receive, process and send out packet in order. Other flow threads can do flow serialization processing for other sessions at the same time.

The following four priority levels are used to determine which type of traffic can install session cache on the IOCs:

- **Priority 1 (P1)**— IPSec and Express Path qualified traffic
- **Priority 2 (P2)**— Fragmentation ordering
- **Priority 3 (P3)**— NAT/SZ (Session serialization) traffic traffic
- **Priority 4(P3)**— All other types of traffic

The IOCs maintain and monitor the threshold levels for session cache usage and update the current real-time session cache usage to the SPU. The SPU requests the IOC to install the session cache for certain high-priority traffic sessions. Session cache usage for high-priority traffic sessions is defined in table:

Table 1: Session Cache Installation Bars

Traffic Type	0% < utilization < 25%	25% < utilization < 50%	50% < utilization < 75%	75% < utilization < 100%
IPsec and Express Path traffic	Yes	Yes	Yes	Yes
Fragmentation Ordering traffic	Yes	Yes	Yes	No
NAT/SZ traffic	Yes	Yes	No	No
Other traffic	Yes	No	No	No

To conserve session entries on the IOC, the flow module selectively installs sessions on the IOC. To facilitate the session install selection, the IOC maintains corresponding thresholds to provide an indication to the flow module (on how full the session cache table is on the IOCs).Two bits in the meta header are added to indicate the current cache table utilization status. All packets going to the SPU will carry these two status bits to inform the flow module of the utilization of the cache table on the IOC.

IPsec VPN Session Affinity Enhancement Using Session Cache

SRX Series Firewalls are fully distributed systems, and an IPsec tunnel is allocated and anchored to a specific SPU. All the traffic that belongs to an IPsec tunnel is encrypted and decrypted on its tunnel-anchored SPU. In order to achieve better IPsec performance, IOC improves the flow module to create sessions for IPsec tunnel-based traffic (before encryption and after decryption) on its tunnel-anchored SPU, and installs session cache for the sessions so that the IOC can redirect the packets directly to the same SPU to minimize packet-forwarding overhead. Express Path traffic and NP traffic share the same session cache table on IOCs.

You need to enable session cache on the IOCs and set the security policy to determine whether a session is for Express Path (formerly known as *services offloading*) mode on the selected Flexible PIC Concentrator (FPC).

To enable IPsec VPN affinity use, the `set security flow load-distribution session-affinity ipsec` command.

NOTE: To enable IPsec VPN affinity, you must also enable the session cache on IOCs by using the `set chassis fpc <fpc-slot> np-cache` command.

Fragmentation Packet Ordering Using NP Session Cache

A session might consist of both normal and fragmented packets. With hash-based distribution, 5-tuple and 3-tuple key can be used to distribute normal and fragmented packets to different SPUs, respectively. On SRX Series Firewalls, all the packets of the session are forwarded to a processing SPU. Due to forwarding and processing latency, the processing SPU might not guarantee packet ordering of the session.

Session cache on the IOCs ensure ordering of packets of a session with fragmented packets. A session cache entry is allocated for normal packets of the session and a 3-tuple key is used to find the fragmented packets. On receipt of the first fragmented packet of the session, the flow module allows the IOC to update the session cache entry to remember the fragmented packets for the SPU. Later, IOC forwards all subsequent packets of the session to the SPU to ensure ordering of packets of a session with fragmented packets.

Configuring IOC to NPC Mapping

An Input/Output card (IOC) to Network Processing Card (NPC) mapping requires you to map one IOC to one NPC. However, you can map multiple IOCs to a single NPC. To balance the processing power in the NPC on the SRX3400, and SRX3600 Services Gateways, the chassis process (daemon) runs an algorithm that performs the mapping. It maps an IOC to an NPC that has the least amount of IOCs mapped to it. You can also use the command-line interface (CLI) to assign a specific IOC to a specific NPC. When you configure the mapping, the chassis process will first use your configuration, then apply the least-number NPC algorithm for the rest of the IOCs.

NOTE: Platform support depends on the Junos OS release in your installation.

To configure the IOC to NPC mapping:

```
[edit]
set chassis ioc-npc-connectivity {
    ioc slot-number npc (none | slot-number);
}
```

NOTE: You must restart the chassis control after you commit the `set chassis ioc-npc-connectivity` command.

Understanding Flow Processing on SRX5K-SPC3 Devices

IN THIS SECTION

The service processing card SRX5K-SPC3 is introduced to improve the performance of security services on the SRX5000 security services gateway. The SPC3 card supports higher throughput, maintains its reliability as it preserves the chassis cluster functionality and scalability for service processing.

The SPC3 card provides support for the following security features:

- Application layer gateway (ALG). [See ALG Overview]
- Advanced anti-malware (Juniper ATP Cloud). [See Juniper Sky Advanced Threat Prevention Administration]
- Application security suite. [See Application Security User Guide for Security Devices]
- Flow-based packet processing implementation

- GPRS tunneling protocol (GTP) and stream control transmission protocol (SCTP). [See General Packet Radio Service User Guide for Security Devices]
- High availability (chassis cluster). [See Chassis Cluster User Guide for SRX Series Devices]
- Intrusion detection and prevention (IDP). [See Intrusion Detection and Prevention Overview]
- Network address translation (NAT). [See Network Address Translation User Guide for Security Devices]
- Stateful firewall
- SSL proxy. [See SSL Proxy]
- Firewall user authentication. [See Authentication and Integrated User Firewalls User Guide for Security Devices]
- Content Security (antivirus, web filtering, content filtering, and antispam). [See UTM User Guide for Security Devices]

The security flow is enhanced to support SPC3 card with all the existing security features that are supported on the SPC2 card.

NOTE: The following limitations apply for the SPC3 card in Junos OS Release 18.2R1-S1:

- Interoperability of SPC3 card and SPC2 card is not supported.
- IPsec VPN functionality is not supported with SPC3 card.

On SRX5000 line devices, SPC3 card interoperates with I/O cards (IOC2, IOC3), Switch Control Board (SCB2, SCB3), Routing Engines and SPC2 cards.

Starting in Junos OS Release 18.4R1, a mix of of SPC3 and SPC2 cards is supported on SRX5000 line devices.

If you are adding the SPC3 cards on SRX5000 line of devices, the new SPC3 card must installed in the lowest-numbered slot of any SPC. The SPC3 card is installed in the original lowest-numbered slot provides the central point (CP) functionality in mixed-mode.
For example, if your services gateway contains a mix of SPC2 and SPC3 cards, an SPC3 must occupy the lowest-numbered slot of any SPC in the chassis. This configuration ensures that the central point (CP) functionality in mixed-mode is performed by the SPC3 card.

On SRX5000 line devices operating in mixed-mode, flow processing is shared between SPC3 and SPC2 cards. Central Point processing takes place on the lowest number SPC slot for which an SPC3 card is installed.

NOTE: When SRX Series Firewalls are operating in a chassis cluster mode, SPC3 and SPC2 cards must be installed in the same slot locations on each chassis.

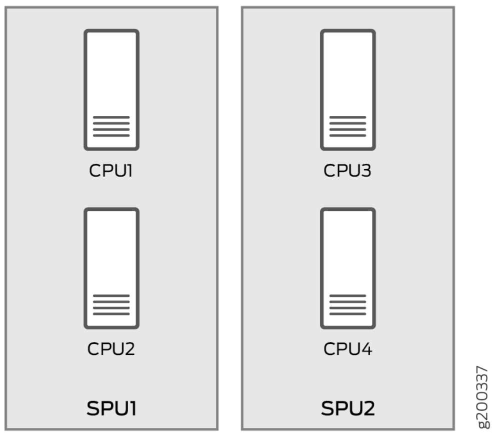
Understanding SPC3 Software Architecture

SPC3 flow architecture is same as CP-Lite architecture. The SPC3 physically has two Services Processing Units (SPU) and each SPU has two CPUs.

When you install one or two SPC3s, traffic processing utilizes 75% of the first SPC. When you install three or more SPC3s, the traffic processing utilizes 50% of the first SPC.

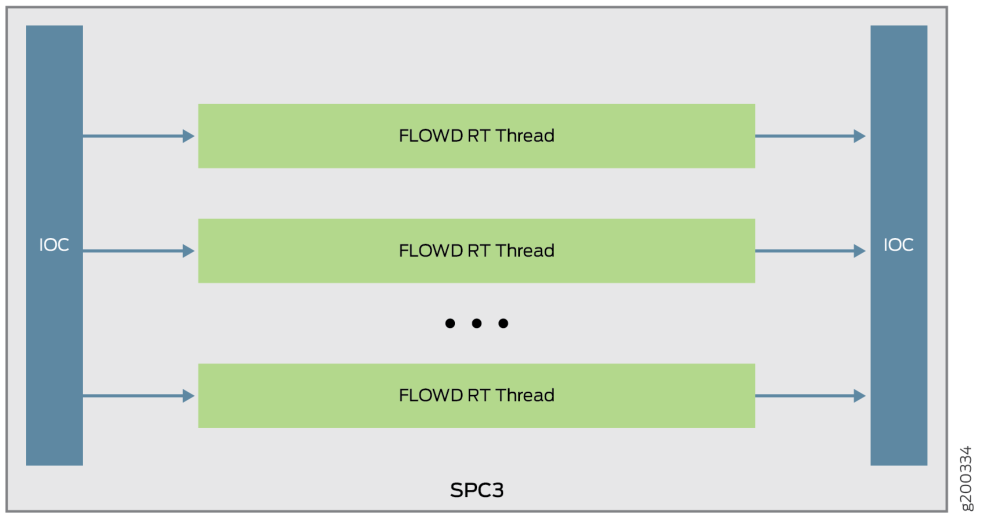
The way the IOC hashes the packets to process the flow is changed. Figure shows the packet flow of SRX Series Firewall with SPC3.

Figure 8: Packet flow on SPC3



On SPC3, packets are distributed from IOC to each core directly. Since the IOC directly hashes packets to the flowd RT thread, the original LBT thread is removed. The packets are now delivered to the flowd thread instead of SPU. If the security flow installs NP sessions, instead of SPU ID, the session thread ID is used by IOC to forward packets to correct thread associate with the session.

Figure 9: Packet flow through flowd thread



Understanding Load Distribution

All packets that come through a revenue port will be distributed to different SPUs based on hash algorithm, which is same as the existing SRX5000 Line devices hash based on CP-Lite architecture. The hash method varies for different types of traffic. The table below lists hash methods.

Table 2: Load Distribution - Hash Methods

Protocol		Ports	Hash Method
TCP		L4 src port and dst port	Hashed by 5-tuple
UDP	Normal	L4 src port and dst port	Hashed by 5-tuple
	GTP	L4 src port and dst port	Hashed by 5-tuple
	IKE	L4 src port and dst port	Hashed by IP pair

Table 2: Load Distribution - Hash Methods *(Continued)*

Protocol	Ports	Hash Method
ICMP	1. ICMP version 4 info message ICMP_ECHO/ICM_ECHOREPLY id/seq ICMP_TSTAMP/ICMP_TSTAMPREPLY id/seq ICMP_IREQ/ICMP_IREQREPLY id/seq ICMP_MASKREQ/ ICMP_MASKREPLY 0x00010001 2. ICMP version 6 info message ICMP6_ECHO_REPLY/ ICMP6_ECHO_REQUEST id/seq 3. ICMP error message Match by embedded IP 4. All others 0x00010001	ICMP info is hashed by 5-tuple; ICMP error is hashed by 3-tuple (no ports info)
SCTP	L4 src port and dst port	Hashed by 5-tuple
ESP	SPI	Hashed by IP pair
AH	SPI	Hashed by IP pair
GRE	If PPTP alg is enabled, sport = call id; dport = 0 By default, port is 0x00010001	Hashed by 3-tuple
PIM	By default, PIM ports 0x00010001	Hashed by 3-tuple
FRAGMENT	First fragment, has the normal ports None first fragment, no ports	Hashed by 3-tuple
Other IP packet	Ports 0x00010001	Hashed by 3-tuple

Table 2: Load Distribution - Hash Methods *(Continued)*

Protocol	Ports	Hash Method
NONE IP	Not applicable	Hashed by Mac address and Ethernet Type (Vlan ID)

Understanding NP Session and Service Offload (SOF)

Network processor (NP) session is an IOC-based session that allows and establishes the SPU sessions. The packets that pass the NP session has the following advantages:

- Avoids session lookup on SPU to gain better performance.
- Avoids extra packet forwarding between session SPU and hash SPU.

Service offload is a special type of NP session to provide low-latency feature for session that needs basic firewall service. Packets that hits the SOF session on an IOC bypass the packet processing on SPU and is directly forwarded by IOC. The following traffic types support service offload:

- Basic firewall (without plugin and fragments), IPv4 and IPv6 TCP, UDP traffic
- IPv4 NAT
- 1Fan-in and 1Fan-out Multicast
- ALGs such as FTP data session

Understanding J-Flow support on SPC3

J-Flow is the juniper version of industry standard traffic monitoring mechanism. It provides a feature to export snapshot of network traffic statistics to the remote server for network monitoring and further data processing. J-Flow supports v5, v8 and v9 format. All these three versions are supported on SPC3.

Understanding Datapath Debug SPU Support (E2E)

Datapath debug provides filter based end-to-end (E2E) packet debug feature on SRX5000 Line devices. It traces packet path and dump packet content.

On SPC3, JEXEC is the only E2E event type that is supported and the following E2E action types are supported:

- Count
- Dump

- Trace
- Trace-summary

Understanding Fragmentation Handling, ISSU, and ISHU Support

On SPC3, fragmented packets are forwarded to “fragment core” in a specific PFE based on its header tuple values. After receiving a fragmented packet, flow performs defragmentation and forwards the packet to its session core. The flow logic does not change and remains the same.

While performing the ISSU, the virtual SPUs are synchronized to related virtual SPU IDs. The ISHU support is based on CP-Lite architecture. Basically, two ISHU operations are supported:

- Insert a new SPC to secondary node.
- Replace an SPC on secondary node, and the number of SPCs should be same as that of primary node.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.4R1	Starting in Junos OS Release 18.4R1, a mix of of SPC3 and SPC2 cards is supported on SRX5000 line devices.
18.2R1-S1	Starting in Junos OS Release 18.2R1-S1, a new service processing card (SPC3) is introduced for the SRX5000 line devices. The introduction of the new card improves the scalability and performance of the device and maintains its reliability as it preserves the chassis cluster functionality. The SPC3 card supports higher throughput and scalability for service processing.
15.1X49-D70	
15.1X49-D10	Starting with Junos OS Release 15.1X49-D10 and Junos OS Release 17.3R1, the session cache of the sessions in the IOC helps to solve certain performance issues.
15.1X49-D10	Starting with Junos OS Release 15.1X49-D10, the SRX5K-MPC (IOC2) and the IOC3 support VPN session affinity through improved flow module and session cache
12.1X48-D30	Starting in Junos OS Release 12.3X48-D30, on the IOC2, VPN session affinity through session cache is supported

Central Point Architecture in Security Devices Overview

IN THIS SECTION

The central point delegates the session processing to one of the SPUs. When a session is not established, the central point selects an SPU to establish the session for the flow, based on load-balancing criteria. If the session already exists, the central point forwards packets for that flow to the SPU hosting it.

Understanding SRX Series Firewalls Central Point Architecture

IN THIS SECTION

The central point (CP) architecture has two basic flow functionalities: load balancing and traffic identification (global session matching). As described in this topic, the central point architecture is implemented either in centric mode, in which all session distribution and session matching is performed by the central point, or in mixed-mode, in which a percentage of Services Processing Unit (SPU) is dedicated to performing the central point functionality.

The central point's main function is to delegate session processing to one of the SPUs. If the session has not yet been established, the central point selects an SPU to establish the session for the flow, based on load- balancing criteria. If the session already exists, the central point forwards packets for that flow to the SPU hosting it. It also redirects packets to the correct SPU in the event that the NPU fails to do so.

The central point maintains a global session table with information about the owner SPU of a particular session. It functions as a central repository and resource manager for the whole system.

NOTE: The central point architecture is also implemented in CP-lite mode in which session management is offloaded from the central point to SPUs for performance and session scaling improvement. CP-lite is not discussed in this topic.

The SRX Series Firewall type in conjunction with the Junos OS release determine which mode is supported.

The central point forwards a packet to its Services Processing Unit (SPU) upon session matching, or distributes traffic to an SPU for security processing if the packet does not match any existing session. The central point architecture is implemented in CP centric mode, in which all session distribution and session matching is performed by the CP or in combo mode

On some SRX Series Firewalls, an entire SPU cannot be dedicated for central point functionality, but a certain percentage of the SPU is automatically allocated for central point functionality and the rest is allocated for normal flow processing. When an SPU performs the function of central point as well as normal flow processing, it is said to be in combination, or *mixed,* mode.

The percentage of SPU dedicated to the central point functionality depends on the number of SPUs in the device. Based on the number of SPUs, there are three modes available on the SRX Series Firewalls—small central point, medium central point, and large central point.

In small central point mode, a small percentage of an SPU is dedicated to central point functionality and the rest is dedicated to the normal flow processing. In medium central point mode, an SPU is almost equally shared for central point functionality and normal flow processing. In large central point mode, an entire SPU is dedicated to central point functionality. In mixed-mode, the central point and SPU share the same load-balancing thread (LBT) and packet-ordering thread (POT) infrastructure.

This topic includes the following sections:

Load Distribution in mixed Mode

The central point maintains SPU mapping table (for load distribution) that lists live SPUs with the logic SPU IDs mapped to the physical Trivial Network Protocol (TNP) addresses mapping. In mixed-mode, the SPU that hosts the central point is included in the table. The load distribution algorithm is adjusted based on session capacity and processing power to avoid overloading of sessions.

Sharing Processing Power and Memory in mixed Mode

The CPU processing power in a mixed-mode SPU is shared based on the platform and the number of SPUs in the system. Similarly, the CPU memory is also shared between the central point and SPU.

An SPU has multiple cores (CPUs) for networking processing. In "small" SPU mixed-mode, CPU functionality takes a small portion of the cores, whereas "medium" SPU mixed-mode requires a larger portion of cores. The processing power for central point functionalities and flow processing is shared, based on the number of Services Processing Cards (SPC), as shown in Table 3 on page 39. Platform support depends on the Junos OS release in your installation.

Table 3: mixed Mode Processing

SRX Series Firewall	Central point mode with 1 SPC or SPC2	Central point mode with 2 or more SPCs or SPC2s	Central point mode with 1 or 2 SPC3s	Central point mode with more than 2 SPC3s
SRX5600	Large	Large	Medium	Large
SRX5800	Large	Large	Medium	Large
SRX5400	Large	Large	Medium	Large

Understanding Enhancements to Central Point Architecture for the SRX5000 Line

IN THIS SECTION

Previously, for the SRX5000 line of services gateways, the central point was a bottleneck in device performance and scaling. When more Services Processing Cards (SPCs) were integrated into the system, the overall processing power increased linearly, but the system connections per second (cps) remained constant and could not be improved because of the single centralized point in the system. This severely impacted the overall system utilizations in both capacity and cps.

The new central point architecture prevents data packets from going through the central point by off-loading session management functionalists to the Services Processing Unit (SPU). Therefore, data packets are directly forwarded from the network processing unit to the SPU instead of going through the central point.

The central point architecture is divided into two modules, the application central point and the distributed central point. The application central point is responsible for global resource management and loading balancing, while the distributed central point is responsible for traffic identification (global session matching). The application central point functionality runs on the dedicated central point SPU, while the distributed central point functionality is distributed to the rest of the SPUs. Now the central point sessions are no longer on the dedicated central point SPU, but with distributed central point on other flow SPUs.

The central point for SRX5000 line refers to the application central point, or the distributed central point or both, with respect to global resource management and load balancing, it refers to the application central point, whereas with respect to traffic identification and session management, it refers to the distributed central point (sometimes referred to the SPU as well).

The SNMP log and SNMP trap were generated by the central point with rate limit. Now, the SNMP log and SNMP trap are generated by the SPU or central point. As there is more than one SPU, the number of SNMP log and traps generated are more. To verify the number of connections per second (CPS) on the device run `SNMP MIB walk nxJsNodeSessionCreationPerSecond` command. The SNMP polling mechanism calculates the CPS value based on the average number of CPS in the past 96 seconds. So, if the CPS is not constant, the number of CPS reported is inaccurate.

Understanding Central Point Session Limit Performance Enhancements

The flow session connection tuple consists of a 32-bit connection tag that is used to uniquely identify GTP-U sessions and SCTP sessions that are not distinguishable by the six part tuple only. You can configure the system to include the session connection tag tuple to identify GTP-U sessions and SCTP sessions by adding the session connection tag to the standard six tuples that identify a session. The system determines the DCP for GTP-U/SCTP by hashing the session connection tag.

The central point architecture distributes GTP-U traffic handled by a gateway GPRS support node (GGSN) and SGSN pair on all SPUs by switching to tunnel endpoint identifier (TEID)-based hash distribution. To handle load-balancing issues, tag-based hash distribution is used to ensure even distribution of SCTP traffic from different associations among all SPUs. (The connection tag for GTP-U is the TEID and for SCTP is the vTag.)

Understanding Central Point Architecture Flow Support for GTP and SCTP

IN THIS SECTION

The central point architecture provides enhanced support for GPRS tunneling protocol, control (GTP-C), GPRS tunneling protocol, user plane (GTP-U), and Stream Control Transmission Protocol (SCTP).

The central point architecture, which is supported on the SRX5400, SRX5600, and SRX5800 devices, is enhanced to address the GTP-C message rate-limiting to protect gateway GPRS support node (GGSN) from GTP-C message flood, to prevent GTP-C packet drop issues during SGSN handover, and to distribute GTP-U traffic handled by a GGSN and SGSN pair on all SPUs by switching to tunnel endpoint identifier (TEID)-based hash distribution. Use the `enable-gtpu-distribution` command to enable or disable GTP-U session distribution. By default, the `enable-gtpu-distribution` command is disabled.

Connection-tag to flow session tuple is introduced to resolve GTP/SCTP load balance issue. All session including Distributed CP (DCP) session and SPU session are modified to accommodate connection-tag. The session creation have following tuple: src-ip, dst-ip, src-port, dst-port, protocol, session-token and connection tag.

The GTP ALG requires GTP-C sessions to be fixed by hashing GGSN IP addresses. The GTP ALG deny GTP-C session creation if the first packet is of uncertain direction, which will cause packet drop. To prevent the GTP-C packets from being dropped, a new flow session is created and the GTP-C traffic is allowed to pass even if the GGSN or SGSN direction is not determined. Later, the GGSN IP is determined using the correct SPU to create the flow session and age out the old session. The intermittent packets hitting the old session will be forwarded to the new SPU and be processed on the new session.

To handle load-balancing issues, tag-based hash distribution is used to ensure even distribution of GTP-U/SCTP traffic among all SPUs. A 32-bit connection tag is introduced that uniquely identifies the GTP-U and the SCTP sessions. The connection tag for GTP-U is the TEID and for SCTP is the vTag. The default connection-tag is 0. The connection tag remains 0 if it is not used by the sessions. Flow will determine connection tag for GTP-U/SCTP sessions and distribute them by hashing connection tag.

A SCTP association is a connection between two SCTP endpoints. Each SCTP endpoint identifies the association with a tag. During association setup (4-way handshakes), two SCTP endpoints exchange their own tags for packet receiving. During 4-way handshake, the receiver of INIT/INIT-ACK records the value of itag, and places into the vtag field of every SCTP packet that transmit within this association. Then the peer uses the vtag to validate the sender of this packet.

Flow sessions created after CP-Lite as follows:

SPU is selected by hash(tag), the Client to Server traffic is handled on hash (tagB) SPU then forwarded to hash (tagA) SPU. Server to Client traffic is handled on hash (tagA) SPU directly.

1. After receive INIT packet, on hash (tagA) SPU:

   DCP-session A1: client=> server, SCTP, Conn ID: 0x0;

   Session A1: client=> server, SCTP, Conn ID: 0x0;

   On hash (tagB) SPU: no session.

2. After receive INIT-ACK packet, on hash (tagA) SPU:

   DCP-session A1: client=> server, SCTP, Conn ID: 0x0;

   DCP-session A2: server => client, SCTP, Conn ID: tagA;

   Session A1: client=> server, SCTP, Conn ID: 0x0;

   Session A2: server => client, SCTP, Conn ID: tagA;

   On hash (tagB) SPU: no session.

3. After receive COOKIE-ECHO packet, on hash (tagA) SPU:

   DCP-session A1: client=> server, SCTP, Conn ID: 0x0;

   DCP-session A2: server => client, SCTP, Conn ID: tagA;

   Session A1: client=> server, SCTP, Conn ID: 0x0;

   Session A2: server => client, SCTP, Conn ID: tagA;

   Session A3: client=> server, SCTP, Conn ID: tagB;

   On hash (tagB) SPU:

   DCP-session: client => server, SCTP, Conn ID: tag B

4. After receive COOKIE-ACK packet, flow sessions have no change.

5. After handshake succeeds, HEARBEAT will be send on all paths.

Understanding the Flow Session Connection Filter Option

The flow session connection tuple consists of a 32-bit connection tag that is used to uniquely identify GTP-U sessions and SCTP sessions that are not distinguishable by the six part tuple only. You can configure the system to include the session connection tag tuple to identify GTP-U sessions and SCTP sessions by adding the session connection tag to the standard six tuples that identify a session. The system determines the DCP for GTP-U/SCTP by hashing the session connection tag.

The central point architecture distributes GTP-U traffic handled by a gateway GPRS support node (GGSN) and SGSN pair on all SPUs by switching to tunnel endpoint identifier (TEID)-based hash distribution. To handle load-balancing issues, tag-based hash distribution is used to ensure even distribution of SCTP traffic from different associations among all SPUs. (The connection tag for GTP-U is the TEID and for SCTP is the vTag.)

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D70	Starting in Junos OS 15.1X49-D70 and Junos OS Release 17.3R1, a new session connection (conn-tag) tag option is available to allow you to add a flow filter to further distinguish GRPS tunneling protocol, user plane (GTP-U) flow sessions and Stream Control Transmission Protocol (SCTP) flow sessions.
15.1X49-D70	Starting in Junos OS 15.1X49-D70 and Junos OS Release 17.3R1, a new session connection (conn-tag) tag option is available to allow you to add a flow filter to further distinguish GRPS tunneling protocol, user plane (GTP-U) flow sessions and Stream Control Transmission Protocol (SCTP) flow sessions.
15.1X49-D40	Starting in Junos OS Release 15.1X49-D40 and Junos OS Release 17.3R1, the central point architecture provides enhanced support for GPRS tunneling protocol, control (GTP-C), GPRS tunneling protocol, user plane (GTP-U), and Stream Control Transmission Protocol (SCTP).
15.1X49-D30	Starting with Junos OS Release 15.1X49-D30 and Junos OS Release 17.3R1, on SRX5000 line devices, the central point architecture is enhanced to handle higher connections per second (cps).

RELATED DOCUMENTATION

Flow-Based Sessions | 45

2

CHAPTER

Flow-Based Sessions

IN THIS CHAPTER

Flow-Based Sessions

IN THIS SECTION

The Junos OS caches the session information that is triggered by the first packet of the flow. The cached session is used by subsequent packets of that same flow and the reverse flow of that session using the flow module, which is integrated into the forwarding path.

Understanding Session Characteristics for SRX Series Firewalls

IN THIS SECTION

Sessions are created, based on routing and other classification information, to store information and allocate resources for a flow. Sessions have characteristics, some of which you can change, such as when they are terminated. For example, you might want to ensure that a session table is never entirely full to protect against an attacker's attempt to flood the table and thereby prevent legitimate users from starting sessions.

Depending on the protocol and service, a session is programmed with a timeout value. For example, the default timeout for TCP is 1800 seconds. The default timeout for UDP is 60 seconds.

If no traffic uses the session before the service timeout, the session is aged out and freed to a common resource pool for reuse. You can affect the life of a session in the following ways:

- You can specify circumstances for terminating sessions by using any of the following methods:

- Age out sessions based on how full the session table is
- Set an explicit timeout for aging out TCP sessions
- Configure a TCP session to be invalidated when it receives a TCP RST (reset) message
- Configure the `fin-invalidate-session` statement to terminate sessions when either session endpoint sends a FIN(ish) message to its peer.

  When the peer endpoint receives the packet with the FIN flag set, it sends an ACK(nowlege) message. Typically, tearing down a session using this method involves transmission of a pair of FIN-ACK messages from each session.

- You can configure sessions to accommodate other systems as follows:
  - Disable TCP packet security checks
  - Change the maximum segment size

Understanding Aggressive Session Aging

The session table is a limited resource for SRX Series Firewalls. If the session table is full, any new sessions will be rejected by the device.

The aggressive session-aging mechanism accelerates the session timeout process when the number of sessions in the session table exceeds the specified high-watermark threshold. This mechanism minimizes the likelihood that the SRX Series Firewalls will reject new sessions when the session table becomes full.

Configure the following parameters to perform aggressive session aging:

- *high-watermark*–The device performs aggressive session aging when the number of sessions in the session table exceeds the *high-watermark* threshold.
- *low-watermark*–The device exits aggressive session aging and returns to normal when the number of sessions in the session table dips below the *low-watermark* threshold.
- *early-ageout* –During aggressive session aging, the sessions with an age-out time lower than the *early-ageout* threshold are marked as invalid.

Example: Controlling Session Termination for SRX Series Firewalls

IN THIS SECTION

This example shows how to terminate sessions for SRX Series Firewalls based on aging out after a certain period of time, or when the number of sessions in the session table is full or reaches a specified percentage. You specify a timeout value or the number of sessions in the session table.

Requirements

Before you begin, understand the circumstances for terminating sessions.

Overview

You can control session termination in certain situations—for example, after receiving a TCP FIN Close or receiving an RST message, when encountering ICMP errors for UDP, and when no matching traffic is received before the service timeout. When sessions are terminated, their resources are freed up for use by other sessions.

In this example, you configure the following circumstances to terminate the session:

- A timeout value of 20 seconds.

  NOTE: The minimum value you can configure for TCP session initialization is 4 seconds. The default value is 20 seconds; if required you can set the TCP session initialization value to less than 20 seconds.

- An explicit timeout value of 280 seconds, which changes the TCP session timeout during the three-way handshake.

  The command sets the initial TCP session timeout to 280 in the session table during the TCP three-way handshake. The timer is initiated when the first SYN packet is received, and reset with each packet during the three-way handshake. Once the three-way handshake is completed, the session timeout is reset to the timeout defined by the specific application. If the timer expires before the three-way handshake is complete, the session is removed from the session table.

- Any session that receives a TCP RST (reset) message is invalidated.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To control session termination for SRX Series Firewalls:

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in theCLI User Guide.

To control session termination for SRX Series Firewalls:

1. Specify an age-out value for the session.

```
[edit security flow]
user@host# set aging early-ageout 20
```

2. Configure an aging out value.

```
[edit  security flow]
user@host# set tcp-session tcp-initial-timeoout 280
```

3. Invalidate any session that receives a TCP RST message.

```
[edit  security flow]
user@host# set tcp-session rst-invalidate-session
```

4. If you are done configuring the device, commit the configuration.

```
[edit ]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security flow` command.

Clearing Sessions for SRX Series Services Gateways

IN THIS SECTION

You can use the `clear` command to terminate sessions. You can clear all sessions, including sessions of a particular application type, sessions that use a specific destination port, sessions that use a specific interface or port, sessions that use a certain IP protocol, sessions that match a source prefix, and resource manager sessions.

Terminating Sessions for SRX Series Services Gateways

You can use the following command to terminate all sessions except tunnel and resource manager sessions. The command output shows the number of sessions cleared. Be aware that this command terminates the management session through which the clear command is issued.

```
user@host> clear security flow session all
```

Terminating a Specific Session for SRX Series Services Gateways

You can use the following command to terminate the session whose session ID you specify.

```
user@host> clear security flow session session-identifier 40000381
```

Using Filters to Specify the Sessions to Be Terminated for SRX Series Services Gateways

You can terminate one or more sessions based on the filter parameter you specify for the `clear` command. The following example uses the protocol as a filter.

```
user@host> clear security flow session protocol 89
```

Configuring the Timeout Value for Multicast Flow Sessions

You can configure the timeout value for multicast flow sessions by configuring a custom application and associating the application with a policy.

Multicast flow sessions have one template session and one or more leaf sessions. Because these sessions are linked together, they can have only one timeout value. The timeout value for multicast flow sessions is determined by considering the timeout values configured in the leaf session policies and the IP protocol timeout values. The highest of these timeout values is selected as the multicast flow session timeout.

If no leaf session timeout values are configured, the IP protocol timeout value is automatically used as the timeout value for the mulicast flow session. The IP protocol timeout is the default and is not configurable.

Configuring leaf session timeouts can be especially helpful for multicast streams that have a longer packet interval than the default IP protocol timeout. For example, multicast streams with a packet interval of more than 60 seconds would experience premature aging-out of flow sessions and packet drops with the UDP timeout value, which is always 60 seconds. For such streams, you can configure a higher leaf session timeout value and prevent packet drop.

To set the leaf session timeout value, configure a custom application and associate the application with a policy:

1. Create a custom application, specify its properties, and specify bypassing the application type.

   ```
   [edit]
   user@host# edit applications application my-udp
   ```

   ```
   [edit applications application my-udp]
   user@host# set protocol udp
   user@host# set destination-port 5000
   user@host# set application-protocol ignore
   ```

2. Set the timeout value for the application protocol.

   ```
   [edit applications application my-udp]
   user@host# set inactivity-timeout 500
   ```

3. Create a policy.

   ```
   [edit]
   user@host# edit security policies from-zone vr-zone-1 to-zone junos-host policy my-policy
   ```

   ```
   [edit security policies from-zone vr-zone-1 to-zone junos-host policy my-policy]
   user@host# set match source-address 192.0.2.1
   user@host# set match destination-address any
   ```

4. Associate the custom application (with the configured timeout) to the policy.

   ```
   [edit security policies from-zone vr-zone-1 to-zone junos-host policy my-policy]
   user@host# set match application my-udp
   user@host# set then permit
   ```

5. If you are done configuring the device, commit the configuration.

   ```
   [edit]
   user@host# commit
   ```

6. To verify the updated session timeout value, enter the **show security flow session** command.

```
user@host> show security flow session destination-prefix 203.0.113.0

Session ID: 2363, Policy name: N/A, Timeout: 498, Valid
  In: 192.0.2.1/17767-->203.0.113.0/5000;udp, If: ge-0/0/1.0, Pkts:0, Bytes:0
  Out: 203.0.113.0/5000-->192.0.2.1/17767/17767;udp, If:.local..4, Pkts:0, Bytes:0



Session ID: 2364, Policy name: my-policy/4, Timeout: -1, Valid
  In: 192.0.2.1/17767-->203.0.113.0/5000;udp, If:ge-0/0/1.0, Pkts:1011, Bytes:258816
  Out: 203.0.113.0/5000-->192.0.2.1/17767;udp, If:ppe0.32769, Pkts:0, Bytes:0
Total sessions: 2
```

In this output, the session ID 2363 section displays a template session. A timeout value of 498 indicates that the template session timeout value is ticking down from the configured value of 500 seconds.

The session ID 2364 section displays a leaf session. The timeout value of -1 essentially indicates that the session will not age out unless the template session ages out.

In this example, the configured leaf session timeout value of 500 seconds is the highest timeout value and is accepted as the template session timeout value for the multicast flow session.

RELATED DOCUMENTATION

TCP Sessions

IN THIS SECTION

To send data over TCP in a network, a three-way handshake session establishment process is followed. There is a process to start a session, and there is also a process to terminate the TCP session. This topic helps you to understand the process involved in processing a TCP session.

Understanding TCP Session Checks per Policy

IN THIS SECTION

By default, the TCP SYN check and sequence check options are enabled on all TCP sessions. The Junos operating system (Junos OS) performs the following operations during TCP sessions:

- Checks for SYN flags in the first packet of a session and rejects any TCP segments with non- SYN flags that attempt to initiate a session.
- Validates the TCP sequence numbers during stateful inspection.

The TCP session check per-policy feature enables you to configure SYN and sequence checks for each policy. Currently, the TCP options flags, no-sequence-check and no-syn-check, are available at a global level to control the behavior of services gateways. To support per-policy TCP options, the following two options are available:

- sequence-check-required: The sequence-check-required value overrides the global value no-sequence-check.
- syn-check-required: The syn-check-required value overrides the global value no-syn-check.

To configure per-policy TCP options, you must turn off the respective global options; otherwise, the commit check will fail. If global TCP options are disabled and SYN flood protection permits the first packet, then the per-policy TCP options will control whether SYN and/or sequence checks are performed.

NOTE:

- The per-policy `syn-check-required` option will not override the behavior of the `set security flow tcp-session no-syn-check-in-tunnel` CLI command.
- Disabling the global SYN check reduces the effectiveness of the device In defending against packet flooding.

CAUTION: Disabling the global SYN check and enforcing the SYN check after policy search will greatly impact the number of packets that the router can process. This in turn will result in intense CPU operations. When you disable global SYN check and enable per-policy SYN check enforcement, you should be aware of this performance impact.

Disabling TCP Packet Security Checks

On an SRX Series Firewall, you can disable security checks on TCP packets to ensure interoperability with hosts and devices with faulty TCP implementations.

The `no-sequence-check` option disables TCP sequence checks. It also increases the throughput.

The `set security flow tcp-session no-sequence-check` command disables the TCP sequence checks on all TCP sessions in default or hash-based modes.

Example: Configuring TCP Packet Security Checks Per Policy

IN THIS SECTION

This example shows how to configure TCP packet security checks for each policy in the device.

Requirements

Before you begin, you must disable the tcp options, `tcp-syn-check`, and `tcp-sequence-check` that are configured at global level. .

Overview

The SYN and sequence check options are enabled by default on all TCP sessions. In environments that need to support large file transfers, or that run nonstandard applications, it might be necessary to configure sequence and sync checks differently for each policy. In this example, you configure sequence and sync check for policy `pol1`.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure TCP packet security checks at the policy level:

1. Configure the checking for the TCP SYN bit before creating a session.

```
[edit]
user@host# set security policies from-zone Zone-A to-zone Zone-B policy pol1 then permit tcp-
options syn-check-required
```

2. Configure the checking for sequence numbers in TCP segments during stateful inspection.

```
[edit]
user@host# set security policies from-zone Zone-A to-zone Zone-B policy pol1 then permit tcp-
options sequence-check-required
```

3. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify that the configuration is working properly, enter the `show security policies detail` command.

Example: Disabling TCP Packet Security Checks for SRX Series Services Gateways

IN THIS SECTION

This example shows how to disable TCP packet security checks in the device.

Requirements

Before you begin, understand the circumstances for disabling TCP packet security checks. .

Overview

Junos OS provides a mechanism for disabling security checks on TCP packets to ensure interoperability with hosts and devices with faulty TCP implementations. During no-SYN-check the Junos OS does not look for the TCP SYN packet for session creation. No-sequence check disables TCP sequence checking validation. Also, increases throughput. SYN check and sequence check are enabled by default. The set security flow command disables TCP SYN checks and TCP sequence checks on all TCP sessions thus reduces security. This may be required in scenarios with customers like big transfer files, or with applications that do not correctly work with standards.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To disable TCP packet security checks:

1. Disable the checking of the TCP SYN bit before creating a session.

```
[edit security flow]
user@host# set  tcp-session no-syn-check
```

2. Disable the checking of sequence numbers in TCP segments during stateful inspection.

```
[edit  security flow]
user@host# set tcp-session no-sequence-check
```

3. If you are done configuring the device, commit the configuration.

```
[edit ]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security flow` command.

Example: Setting the Maximum Segment Size for All TCP Sessions for SRX Series Firewalls

IN THIS SECTION

This example shows how to set the maximum segment size for all TCP sessions for SRX Series Firewalls.

Requirements

Before you begin, understand the circumstances for setting the maximum segment size.

Overview

You can terminate all TCP sessions by changing the TCP maximum segment size (TCP-MSS). To diminish the likelihood of fragmentation and to protect against packet loss, you can use the tcp-mss to specify a lower TCP MSS value. This applies to all TCP SYN packets traversing the router’s ingress interfaces whose MSS value is higher than the one you specify.

If the DF bit is set, it will not fragment the packet and Junos OS will send ICMP error type 3 code 4 packet to the application server (Destination Unreachable; Fragmentation Needed and DF set). This ICMP error message contains the correct MTU (as defined in tcp-mss) to be used by the application server, which should receive this message and adjust the packet size accordingly. This is specifically required with VPNs, as IPsec has added packet overhead; thus tcp-mss must be lowered appropriately.

NOTE: When running SRX Series Firewalls in packet mode, you use the **set system internet-options tcp-mss** to adjust the TCP-MSS value. All ports are affected by the TCP-MSS configuration; you cannot exclude a particular port. When running SRX Series Firewalls in flow mode, although you can use the **set system internet-options tcp-mss** , we recommend using only the **set security flow tcp-mss** to adjust the TCP-MSS value. If both statements are configured, the lower of the two values will take effect.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure the maximum segment size for all TCP sessions:

1. Set the TCP maximum segment size for all TCP sessions.

```
[edit security flow]
user@host# set tcp-mss all-tcp mss 1300
```

2. If you are done configuring the device, commit the configuration.

```
[edit ]
user@host# commit
```

Results

From configuration mode, confirm your configuration by entering the `show security flow` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

For brevity, this `show` command output includes only the configuration that is relevant to this example. Any other configuration on the system has been replaced with ellipses (...).

```
[edit]
user@host# show security flow
...
tcp-mss{
    all-tcp{
        mss 1300;
    }
}
...
```

Verification

To verify the configuration is working properly, enter the `show configuration security flow` command from operational mode.

```
user@host> show configuration security flow
tcp-mss{
    all-tcp{
        mss 1300;
    }
}
```

TCP Out-of-State Packet Drop Logging Overview

IN THIS SECTION

Within any packet-switched network, when demand exceeds available capacity, the packets are queued up to hold the excess packets until the queue fills, and then the packets are dropped. When TCP operates across such a network, it takes any corrective actions to maintain error-free end-to-end communications.

Flow modules already support generating RTLOG for session-based events like session creation and session close. SRX Series Firewalls now support the generation of RTLOG for packet-based events like packet drop without a session existing.

SRX Series Firewalls support logging of unsynchronized TCP out-of-state packets that are dropped by the flow module.

The TCP out-of-state packet drop logging feature avoids any packet loss and enables packet recovery by logging the out-of-sync packets for error free communication, and prevents the database servers from going out of sync. This feature is built on top of the security log (RTLOG) facility.

TCP out-of-state packet drop logging supports capturing of TCP packet drop logs under the following conditions:

- **Session ages out**—When there are cloud applications running on top of long TCP sessions, and when these applications do not refresh the TCP sessions after the session ages out, the TCP packets are dropped. This feature supports logging of these dropped TCP packets.

- **Unsynchronized first packets due to attacks or asymmetric routes**—When you deploy SRX Series Firewalls at two sites , and when routing sometimes forces asymmetric traffic, the synchronization (SYN) packet is seen at one site but the synchronization acknowledgment (SYN_ACK) packets are seen at another site.

  This means that the SRX Series Firewall sees a TCP ACK packet for which it does not have a matching state table entry. This might occur because the connection was inactive for a period of time or the connections tables were flushed (for example, because of a policy installation or restart).

  The SYN_ACK packets that are seen at another site in this case were denied by the SRX Series Firewall but were not logged. This feature supports logging of the denied SYN_ACK packets.

- **Other out-of-state conditions (like TCP sequence check fail and synchronization packet received in FIN state)**—When an SRX Series Firewall detects a sequence failure, if the device is in TCP four-way close state but receives SYN packets, or if there is a three-way handshake failure, the SRX Series Firewall drops the TCP packets and these dropped packets are logged.

NOTE: The unsynchronized TCP out-of-state packet drop log is a packet-based log, not a session-based log.
TCP out-of-state packet drop logging is designed with a throttle mechanism to protect CPU from being attacked, and within each throttle interval some logs can be dropped.

Only TCP out-of-state packets dropped by Flow module are logged. TCP packets dropped by TCP-proxy and IDP are not logged.

Understanding TCP Out-of-State Packet Drop Logging

To understand the implementation of TCP out-of-state packet drop logging, consider that you deploy SRX Series Firewalls at two sites and that routing sometimes forces asymmetrical traffic, where the SYN packet is seen at one site but the SYN_ACK packet is seen at another site. The SYN_ACK packet in this case would be denied but not logged. The TCP out-of-state packet drop logging feature provides visibility into these unsynchronized packet drops.

Consider the scenario where databases within the data center keep their TCP sockets open, with no keepalives being sent. If no data is being transmitted, the SRX Series Firewall will timeout the sessions. Although the databases will send some data through that TCP socket, when the traffic reaches the SRX Series Firewall, the session is no longer there and the packet is dropped, but not logged. These out-of-state TCP packets that are dropped are now logged by the SRX Series Firewall.

Supported TCP Out-of-State Logging Features

TCP out-of-state logging supports the following features:

- A packet filter component to filter target traffic.
- A throttle component to protect CPU from being overloaded by log messages.
- Flexibility to change the log generation rate.

Packet Filter Component

The logging filter leverages the current flow trace filter. It provides different ways to filter traffic. You must configure the filters to generate packet logs, otherwise logs will not be triggered.

This filter functionality avoids enabling logs unexpectedly. The maximum filters supported are 64.

Use the `set security flow packet-log packet-filter` *`<filter-name>`* command to enable the related filter components you want.

Throttle Component

Logging every TCP out-of-state packet can overload the device when traffic is heavy or when an attack occurs. If the CPU is idle and you want to log as many messages as possible, then this could lead to CPU overload.

The throttle mechanism allows you to configure the throttle interval from the CLI, so you can protect your CPU from being overloaded.

A hash table is introduced to map your logged data. The hash key is generated with the source-IP address, destination-IP address, source port, and destination port.

Within each throttle interval, only a limited number (more than one) of messages will be sent to RTLOG. The remaining log messages will be throttled.

The default throttle interval is 1 second. The throttle interval (at the millisecond level) needs to be configured as a power of two or zero (0, 1, 2, 4, 8, 16 ... 2^N).

When the throttle interval is configured as 0, no throttle mechanism will be involved. This is suitable for scenarios where traffic is very light and you want to record all the packet drop logs.

Configuration of the throttle interval as 2^N makes the throttle mechanism lockless and provides good log capture performance.

Flexibility for Changing the Log Generation Rate

Based on the throttle interval set, the log generation rate can be modified and managed.

This means that within each 32-millisecond (ms) interval, a limited number of logs could be generated and the remaining could be dropped. We recommend that you configure the interval as (0, 1, 2, 4, 8, 16, 32 … 2^N).

If the input value is not aligned to 2^N, it will be aligned to 2^N automatically during flow processing. For example, if you configure a 10-ms interval it will be aligned to an 8-ms interval automatically.

Understanding How Preserving Incoming Fragmentation Characteristics Can Improve Throughput

This topic covers the benefits of using the SRX Series Firewall to preserve the characteristics of incoming packet fragments.

When data is sent from one host to another, it is transmitted as a series of packets. Performance is improved and network resources are conserved when packets of the largest size can transit the path from the source node to the destination node without being fragmented at any link in the datapath. When a packet must be fragmented into smaller packets to transit a link in the path because the packet is larger than that of the maximum transmission unit (MTU) established for that link, each of the resulting fragments must contain packet header information, in addition to the payload, or data. The increased overhead can lower throughput and degrade network performance. Also, the packet fragments must be reassembled at the destination node, which consumes additional network resources.

On the other hand, network resources are wasted when a host sends packets that are much smaller than the path MTU (path maximum transmission unit), resulting in suboptimal throughput. The path MTU discovery process works to discover the optimal MTU size for fragments that transit the datapath from the source node to the destination node for a session. The optimal packet size, then, is that of the path MTU. Fragmentation occurs when the size of a packet exceeds the path MTU.

If application-layer services are configured on the SRX Series Firewall, packet fragments at the ingress interface must be reassembled before the services can be applied and the content inspected. These reassembled packet fragments must be broken down again before the data is transmitted out the egress interface. Normally, it is the MTU size of the egress interface that determines the size of fragments transmitted out the SRX Series Firewall to the next link. It could be the case that the egress MTU size on the SRX Series Firewall is larger than the path MTU, which, again, would result in packet fragmentation in the datapath, reducing performance or causing packet drop. Packet fragments must be small enough to transit every link in the path from source to destination.

By default, the SRX Series Firewall uses the MTU size configured for the egress interface to determine the size for packet fragments it transmits. However, if you enable the feature for preserving incoming fragment characteristics, the SRX Series Firewall detects and saves the size of incoming packet fragments.

To diminish the likelihood of packet fragmentation in the datapath, the SRX Series Firewall keep track of and adjust the egress MTU for that flow. It identifies the maximum size of all incoming fragments. It uses that information in conjunction with the existing MTU of the egress interface to determine the correct MTU size for fragmented packets sent out the egress interface. The SRX Series Firewall compares the two numbers. It takes the smaller number and uses it for the egress interface MTU size.

Configure the device using the `set security flow preserve-incoming-frag-size` command to enable the feature that takes into account the size of incoming packet fragments.

Table 4 on page 64 summarizes how the SRX Series egress MTU size is determined.

Table 4: How the Final Egress MTU Size for Fragments Exiting the SRX Series Firewall Is Determined

Incoming Fragment Size	Existing Egress MTU Size	Final Egress MTU Size
If the largest fragment is	*smaller* than the existing egress MTU size	largest incoming fragment size is used.
If the largest fragment is	larger than the existing egress MTU size	existing egress interface MTU is used.

NOTE: This feature is supported on SRX Series Firewalls. It supports through-traffic and traffic exiting a tunnel. It is applies to both IPv4 and IPv6 traffic.

The following two considerations affect fragment size:

- For stream-based applications, such as Content Security and ALG, the applications themselves could change or reassemble packets even if there were no fragments received. In this case, the existing egress interface MTU is used.

- When a path MTU discovery packet is delivered to a session, the path MTU for that session is reset to the value established by the path MTU packet.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D100	Configure the device using the `set security flow preserve-incoming-frag-size` command to enable the feature that takes into account the size of incoming packet fragments.

RELATED DOCUMENTATION

Flow-Based Performance | 76

Flow Distribution and Packet-Ordering | 80

ECMP Flow-Based Forwarding

IN THIS SECTION

This topic provides a brief overview of equal-cost multipath (ECMP) for forwarding and reverse side traffic on Junos OS SRX Series Firewalls and vSRX Virtual Firewall instances. For comprehensive coverage of the ECMP implementation on Junos OS SRX Series Firewalls and vSRX Virtual Firewall instances.

Understanding ECMP Flow-Based Forwarding

IN THIS SECTION

Equal-cost multipath (ECMP) is a network routing strategy that allows for traffic of the same session, or flow—that is, traffic with the same source and destination—to be transmitted across multiple paths of equal cost. It is a mechanism that allows you to load balance traffic and increase bandwidth by fully utilizing otherwise unused bandwidth on links to the same destination.

When forwarding a packet, the routing technology must decide which next-hop path to use. In making a determination, the device takes into account the packet header fields that identify a flow. When ECMP is used, next-hop paths of equal cost are identified based on routing metric calculations and hash algorithms. That is, routes of equal cost have the same preference and metric values, and the same cost to the network. The ECMP process identifies a set of routers, each of which is a legitimate equal cost next hop towards the destination. The routes that are identified are referred to as an ECMP set. Because it addresses only the next hop destination, ECMP can be used with most routing protocols.

An equal-cost multipath (ECMP) set is formed when the routing table contains multiple next-hop addresses for the same destination with equal cost. (Routes of equal cost have the same preference and metric values.) If there is an ECMP set for the active route, Junos OS uses a hash algorithm to choose *one* of the next-hop addresses in the ECMP set to install in the forwarding table.

You can configure Junos OS so that multiple next-hop entries in an ECMP set are installed in the forwarding table. On Juniper Networks devices, per-flow load balancing can be performed to spread traffic across multiple paths between routing devices. On Juniper Networks security devices, source and destination IP addresses and protocols are examined to determine individual traffic flows. Packets for the same flow are forwarded on the same interface; the interface does not change when there are additions or changes to the ECMP set. This is important for features such as source NAT, where the translation is performed only during the first path of session establishment for IDP, ALG, and route-based VPN tunnels. If a packet arrives on a given interface in an ECMP set, the security device ensures that reverse traffic is forwarded through the same interface.

ECMP flow-based forwarding on security devices applies to IPv4 and IPv6 unicast traffic flows. The ECMP flow-based forwarding of IPv6 unicast traffic is supported on all SRX Series Firewalls and vSRX Virtual Firewall instances. Multicast flow is not supported. Use Feature Explorer to confirm platform and release support for specific feature.

On Juniper Networks security devices, the maximum number of next-hop addresses in an ECMP set that can be installed in the forwarding table is 16. If there are more than 16 next-hop addresses in an ECMP set, only the first 16 addresses are used.

In a *chassis cluster* deployment, a *local* interface is an interface that is on the same node as the interface on which a packet arrives, and a *remote* interface is an interface that is on the other chassis cluster node. If an ECMP route has both local and remote interfaces in a chassis cluster, then the local interface is favored for the next hop.

If a next-hop address is no longer part of the ECMP set or if it is removed from the routing table because of a route change, a flow that uses the next hop is rerouted and the session is not affected. Rerouting of the flow also occurs if there is a configuration change that takes away the next-hop address or if an administrator takes down the next-hop interface without deleting it. If a next-hop address is removed from the routing table because the interface is deleted or the session is intentionally cleared, the session is killed without being rerouted.

NOTE: We recommend that interfaces in an ECMP set be in the same security zone. If a flow is rerouted and the rerouted flow uses an interface in a different security zone than the original route, the session is killed.

To configure ECMP flow-based forwarding on Juniper Networks security devices, first define a load-balancing routing policy by including one or more `policy-statement` configuration statements at the [`edit policy-options`] hierarchy level, with the action `load-balance per-flow`. Then apply the routing policy to routes exported from the routing table to the forwarding table. To do this, include the `forwarding-table` and `export` configuration statements at the [`edit routing-options`] hierarchy level.

ECMP Implementation for Junos OS SRX Series Firewalls and vSRX Virtual Firewall Instances

You can configure ECMP for SRX Series Firewalls and vSRX Virtual Firewall instances to implement per-flow load balancing to spread traffic across multiple paths between routing devices. Routes of equal cost have the same preference and metric values. These devices examine the source IP address, the destination IP address, and the protocol to determine individual traffic flows. Traffic with the same source IP address, destination IP address, and protocol number that is permitted by a security policy is forwarded to the same next hop. Junos OS on these devices uses the flow information in its hashing logic.

For Junos OS SRX Series Firewalls and vSRX Virtual Firewall instances, an ECMP set is formed when the routing table contains multiple next-hop addresses for the same destination with equal cost. ECMP allows for multiple next-hop entries in an ECMP set to be installed in the forwarding table. Packets for the same flow are forwarded on the same interface; the interface does not change when there are additions or changes to the ECMP set.

If there is an ECMP set for the active route, Junos OS uses a hash algorithm to choose *one* of the next-hop addresses in the ECMP set to install in the forwarding table.

ECMP for Reverse Traffic

Starting in Junos OS Release 17.3, if you enable ECMP support for reverse traffic, the SRX Series Firewall uses a hash algorithm to determine the interface to use for reverse traffic in a flow. This process is similar to asymmetric routing in which a packet traverses from a source to a destination in one path and takes a different path when it returns to the source.

If you do not enable this feature, the SRX Series Firewall selects a route in the ECMP set to the incoming interface for reverse traffic, which is the default behavior.

You use the `allow-reverse-ecmp` configuration statement in the `[edit security flow]` hierarchy to configure ECMP flow-based forwarding to use a hash algorithm in selecting a route in the ECMP set for reverse traffic transit. That is, if you enable this function, rather than selecting a route to the incoming interface, the SRX Series Firewall uses a hash algorithm to select a route in the ECMP set for reverse traffic.

Because the ECMP flow-based policy is zone-based, ECMP reverse lookup support ensures that the egress interface used for reverse traffic is in the same zone as the ingress interface used for arriving traffic.

NOTE: Interfaces in an ECMP set must be in the same security zone. If the egress interface zone is different from the ingress interface zone, a session can be created but the packets will be dropped.

CAUTION: If you decide to enable reverse ECMP, be aware of the following condition and take action to avoid it: When ECMP flow-based forwarding is used, the SRX Series Firewall could cause upstream devices to see only one-way traffic of a session. Problems might ensue for upstream devices that maintain session state, for example, for TCP-proxy and SYN-proxy. The issue is similar to asynchronous routing behavior.

Example: Configuring ECMP Flow-Based Forwarding

IN THIS SECTION

This example shows how to configure ECMP flow-based forwarding.

Requirements

No special configuration beyond device initialization is required before configuring this feature.

Overview

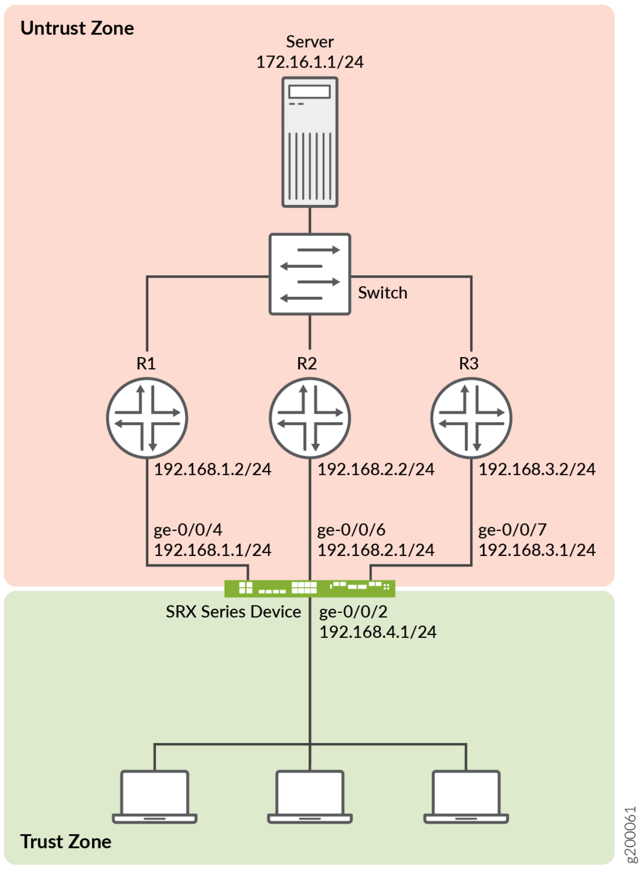
IN THIS SECTION

This example configures three static ECMP routes on an SRX Series Firewall. Each static route uses a different next-hop router to reach the destination server. The interfaces towards the routers are assigned to the untrust security zone. This example creates a load-balancing routing policy named `load-balancing-policy` and applies the policy to all routes exported from the routing table to the forwarding table.

Topology

Figure 10 on page 70 shows the topology used in this example.

Figure 10: ECMP Routes



Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

```
## Interfaces ##
set interfaces ge-0/0/2 unit 0 family inet address 192.168.4.1/24
set interfaces ge-0/0/4 unit 0 family inet address 192.168.1.1/24
set interfaces ge-0/0/6 unit 0 family inet address 192.168.2.1/24
set interfaces ge-0/0/7 unit 0 family inet address 192.168.3.1/24
## Static routes ##
set routing-options static route 172.16.1.0/24 next-hop 192.168.1.2
set routing-options static route 172.16.1.0/24 next-hop 192.168.2.2
set routing-options static route 172.16.1.0/24 next-hop 192.168.3.2
## Security zones, address book entry, and policy ##
set security zones security-zone trust interfaces ge-0/0/2
set security zones security-zone untrust interfaces ge-0/0/4
set security zones security-zone untrust interfaces ge-0/0/6
set security zones security-zone untrust interfaces ge-0/0/7
set security address-book global address FTP-servers 172.16.1.0/24
set security policies from-zone trust to-zone untrust policy permit-ftp match source-address any
set security policies from-zone trust to-zone untrust policy permit-ftp match destination-
address FTP-servers
set security policies from-zone trust to-zone untrust policy permit-ftp match application junos-
ftp
set security policies from-zone trust to-zone untrust policy permit-ftp then permit
## ECMP routing policy ##
set policy-options policy-statement load-balancing-policy then load-balance per-flow
set routing-options forwarding-table export load-balancing-policy
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy.

To configure ECMP flow-based forwarding:

1. Configure interfaces.

   ```
   [edit interaces]
   user@host# set ge-0/0/2 unit 0 family inet address 192.168.4.1/24
   user@host# set ge-0/0/4 unit 0 family inet address 192.168.1.1/24
   user@host# set ge-0/0/6 unit 0 family inet address 192.168.2.1/24
   user@host# set ge-0/0/7 unit 0 family inet address 192.168.3.1/24
   ```

2. Configure static routes.

   ```
   [edit routing-options]
   user@host# set static route 172.16.1.0/24 next-hop 192.168.1.2
   user@host# set static route 172.16.1.0/24 next-hop 192.168.2.2
   user@host# set static route 172.16.1.0/24 next-hop 192.168.3.2
   ```

3. Create the `trust` and `untrust` security zones, and include the related interfaces.

   ```
   [edit security]
   user@host# set zones security-zone trust interfaces ge-0/0/2
   user@host# set zones security-zone untrust interfaces ge-0/0/4
   user@host# set zones security-zone untrust interfaces ge-0/0/6
   user@host# set zones security-zone untrust interfaces ge-0/0/7
   ```

4. Configure an address book entry for the server subnet.

   This entry is used in the security policy.

   ```
   [edit security address-book]
   user@host# set global address FTP-servers 172.16.1.0/24
   ```

5. Configure a security policy.

   ```
   [edit security policies from-zone trust to-zone untrust]
   user@host# set policy permit-ftp match source-address any
   user@host# set policy permit-ftp match destination-address FTP-servers
   user@host# set policy permit-ftp match application junos-ftp
   user@host# set policy permit-ftp then permit
   ```

6. Create a load-balancing routing policy.

```
[edit policy-options]
user@host# set policy-statement load-balancing-policy then load-balance per-flow
```

7. Apply the routing policy to all routes exported from the routing table to the forwarding table.

```
[edit routing-options]
user@host# set forwarding-table export load-balancing-policy
```

Results

From configuration mode, confirm your configuration by issuing the `show interfaces`, `show security`, `show policy-options`, and `show routing-options` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show interfaces
ge-0/0/2 {
    unit 0 {
        family inet {
            address 192.168.4.1/24;
        }
    }
}
ge-0/0/4 {
    unit 0 {
        family inet {
            address 192.168.1.1/24;
        }
    }
}
ge-0/0/6 {
    unit 0 {
        family inet {
            address 192.168.2.1/24;
        }
    }
}
```

```
ge-0/0/7 {
    unit 0 {
        family inet {
            address 192.168.3.1/24;
        }
    }
}
user@host# show security
address-book {
    global {
        address FTP-servers 172.16.1.0/24;
    }
}
policies {
    from-zone trust to-zone untrust {
        policy permit-ftp {
            match {
                source-address any;
                destination-address FTP-servers;
                application junos-ftp;
            }
            then {
                permit;
            }
        }
    }
}
zones {
    security-zone trust {
        interfaces {
            ge-0/0/2.0;
        }
    }
    security-zone untrust {
        interfaces {
            ge-0/0/4.0;
            ge-0/0/6.0;
            ge-0/0/7.0;
        }
    }
}
user@host# show policy-options
policy-statement load-balancing-policy {
    then {
        load-balance per-flow;
    }
}
```

```
[edit]
user@host# show routing-options
static {
    route 172.16.1.0/24 next-hop [ 192.168.1.2 192.168.2.2 192.168.3.2 ];
}
forwarding-table {
    export load-balancing-policy;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Verifying the Forwarding Table

Purpose

Verify that the route information for all ECMP routes appears in the forwarding table.

Action

From operational mode, enter the **show route forwarding-table destination 172.16.1.0** command.

```
user@host> show route forwarding-table destination 172.16.1.0
Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
172.16.1.0/24      user     0                    ulst 262142     2
```

```
                                192.168.1.2        ucst   560      2 ge-0/0/4.0
                                192.168.2.2        ucst   561      2 ge-0/0/6.0
                                192.168.3.2        ucst   562      2 ge-0/0/7.0
...
```

Meaning

The output shows a next hop type of `ulst`, which means the route has multiple eligible next hops. Packets destined for the 172.16.1.0 network can use any next hop in the list.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D60	Starting with Junos OS Release 15.1X49-D60, ECMP flow-based forwarding of IPv6 unicast traffic is supported on all SRX Series Firewalls and vSRX Virtual Firewall instances. Multicast flow is not supported.
15.1X49-D60	Starting in Junos OS Release 15.1X49-D60 and Junos OS Release 17.3R1, ECMP flow-based forwarding of IPv6 unicast traffic is supported on all SRX Series Firewalls and vSRX Virtual Firewall instances. Multicast flow is not supported.

RELATED DOCUMENTATION

Flow-Based Sessions | 45
Flow Distribution and Packet-Ordering | 80

Flow-Based Performance

IN THIS SECTION

This topics explains about the performance of the session capacity. Expanding the session capacity and reverting back to the default session capacity.

Expanding Session Capacity by Device

IN THIS SECTION

To take advantage of the processing potential of a fully loaded SRX5600, SRX5800 device, or vSRX Virtual Firewall, you can expand the maximum number of concurrent sessions for these devices.

Table 5 on page 77 shows the maximum number of concurrent sessions allowed on these devices by default and with expanded capacity. Platform support depends on the Junos OS release in your installation.

Table 5: Maximum Central Point Session Increases

SRX Series Devices	Maximum Concurrent Sessions on a Fully Loaded System	
	Default	With Expanded Capacity
SRX5400	42 million	Expansion not available
SRX5600	114 million	Expansion not available
SRX5800	258 million	Expansion not available
SPC2	6 million	Expansion not available
SPC3	25 million	Expansion not available

The method used for expanding session capacity depends on the device:

- CLI optimization option on an SRX5800 device

Reverting to Default Session Capacity on an SRX5800 Device

Reverting to the default session capacity on an SRX5800 device requires a CLI configuration change.

1. Enter the following command at the CLI configuration prompt to reestablish the default session capacity value:

```
user@host# set security gprs gtp enable
```

2. Commit the configuration.

```
user@host# commit
```

3. Reboot the device to implement the new value.

Verifying the Current Session Capacity

IN THIS SECTION

Purpose

The central point session summary includes the maximum sessions setting for the device. From this value you can determine if the session capacity has been modified as you expected.

Action

To verify the current setting of the central point session capacity, enter the following CLI command.

```
user@host> show security flow cp-session summary
```

```
DCP Flow Sessions on FPC10 PIC0:

Valid sessions: 0
Pending sessions: 0
Invalidated sessions: 0
Sessions in other states: 0
Total sessions: 0

DCP Flow Sessions on FPC10 PIC1:

Valid sessions: 2
Pending sessions: 0
Invalidated sessions: 0
Sessions in other states: 0
Total sessions: 2
Maximum sessions: 7549747
Maximum inet6 sessions: 7549747

DCP Flow Sessions on FPC10 PIC2:

Valid sessions: 2
Pending sessions: 0
Invalidated sessions: 0
Sessions in other states: 0
Total sessions: 2
Maximum sessions: 7549747
Maximum inet6 sessions: 7549747

DCP Flow Sessions on FPC10 PIC3:

Valid sessions: 1
Pending sessions: 0
Invalidated sessions: 0
Sessions in other states: 0
Total sessions: 1
```

```
Maximum sessions: 7549747
Maximum inet6 sessions: 7549747
```

Meaning

The `Maximum sessions` value reflects the current session capacity on your device. A value of 14000000 means that the SRX5800 device is configured for the expanded number of central point sessions.

Flow Distribution and Packet-Ordering

IN THIS SECTION

This topic describes about the load distribution and the packet ordering on SRX5000 Line devices.

Understanding Load Distribution in SRX5000 Line Devices

IN THIS SECTION

The load distribution algorithm, which is supported on the SRX5800, SRX5600, and SRX5400 devices, is adjusted based on session capacity and processing power. (Actual platform support depends on the Junos OS release in your installation.)

Hash-based session distribution uses a hash table. The SPU session weight table is used to assign an SPU ID to each hash index in the session distribution hash table. This way, the number of sessions created on each SPU using hash-based distribution is proportional to the SPU's weight in the SPU session weight table. Each NPU also keeps an identical SPU session weight table and session distribution hash table that it uses to select an SPU to forward packets that do not match an NPU session.

In the event of a SPU failure, the Routing Engine will reset all of the cards on the dataplane including IOCs and NPCs in order to maintain hash table consistency for session distribution.

In hash-based session distribution, weights are based on session capacity. We recommend the hash session distribution mode when high session capacity is required.

Load distribution on SRX5000 line devices is always hash-based.

Insertion and removal of SPCs causes recalculation of the SPU session weight table at central point initialization time because the chassis must reboot after insertion.

The central point architecture enhancements prevent data packets from going through the central point by offloading traffic management to SPUs. The system session capacity is extended, as the session limit on the central point is removed.

Calculating SPU ID

The SPU ID for a device equipped with SRX3K-SPC-1-10-40, SRX5K-SPC-2-10-40, or SRX5K-SPC3 Services Processing Card (SPC) is calculated as follows:

```
SPU ID = (FPC ID X 4) + PIC ID
```

The SRX5K-SPC-2-10-40 and SRX5K-SPC3 contains two PICs per card, four PICs per card (FPC), and two PICs per card respectively. For example, a device contains 2 cards in slot 1 (FPC ID 0) and slot 2 (FPC ID 1), the expected SPU IDs are as follows:

- For SPC1: (0, 1) and (4, 5), total 4 SPUs in 2 cards.
- For SPC2: (0, 1, 2, 3) and (4, 5, 6, 7), total 8 SPUs in 2 cards.
- For SPC3: (0, 1) and (4, 5), total 4 SPUs in 2 cards.

For FPC1 (the second card) and PIC1 (the second PIC in the card), the SPU ID is calculated as:

```
        SPU ID = (FPC ID X 4) + PIC ID
              = (1 X 4) + 1
```

```
                    = 4 + 1
                    = 5
```

Use this convention while referring the SPU ID for CLI and SNMP.

Hash-Based Forwarding on the SRX5K-MPC, SRX5K-MPC3-40G10G (IOC3), and the SRX5K-MPC3-100G10G (IOC3)

On these SRX Series Firewalls, a packet goes through a series of events involving different components as it progresses from ingress to egress processing. With the datapath packet forwarding feature, you can obtain quick delivery of I/O traffic over the SRX 5000 line of devices.

The SRX5K-MPC, SRX5K-MPC3-40G10G (IOC3), and SRX5K-MPC3-100G10G (IOC3) are interface cards supported on the SRX5400, SRX5600, and SRX5800 devices. The Modular Port Concentrator (MPC) provides load-balancing services for Services Processing Units (SPUs) by using the hash-based forwarding method.

In hash-based forwarding, the packet might be forwarded by the MPC to a selected SPU (DCP) instead of the central point. This approach enhances session scaling and prevents overloading of the central point.

Hash value calculation involves the following steps:

- For IPv4 packets, the hash-based forwarding module generates the hash value based on Layer 3 and Layer 4 information, depending on different Layer 4 protocol types.
- For Stream Control Transmission Protocol (SCTP), TCP, UDP, Authentication Header (AH), edge service provider (ESP), and Internet Control Message Protocol (ICMP) protocols, the hash module utilizes Layer 4 information to generate the hash value. For any other protocols, only Layer 3 information is used in hash generation.
- For IPv4 fragment packets, the hash value is calculated using only the Layer 3 information. This also applies to the first fragment of the packet.
- For non-IP packets, the hash-based forwarding module uses the Layer 2 information to calculate the hash value.

Once a hash value is calculated according to the packet's Layer 2, Layer 3, or Layer 4 information, an SPU ID is assigned to each hash index in the session distribution hash table.

NOTE: The SRX5K-MPC (IOC2), SRX5K-MPC3-40G10G (IOC3), and SRX5K-MPC3-100G10G (IOC3) can only be used on SRX5400, SRX5600, and SRX5800 devices that are configured for hash-based session distribution.

When the hash-based session distribution mode is enabled, the system changes its behavior to high-session-capacity-based mode when the SRX5K-MPC, SRX5K-MPC3-40G10G (IOC3), and SRX5K-MPC3-100G10G (IOC3) are installed on the device.

NOTE: On SRX5000 line devices with an SRX5K-MPC, SRX5K-MPC3-40G10G (IOC3), or SRX5K-MPC3-100G10G (IOC3) installed, during a system or an SPU reboot, when the hash-based session distribution mode is enabled, traffic will pass only when all SPUs are up after the reboot.

The MPCs on the IOC3 provide load-balancing services for SPUs by performing hash-based datapath packet forwarding to interconnect with all existing IOCs and SPCs.

The IOC3 processes ingress and egress packets. The IOC3 parses the ingress packet and sends it to the SPU for further security processing, including flow session lookup, zone and policy check, VPN, ALG, and so on.

The IOC3 manages packet data memory and fabric queuing for packet lookup and encapsulation functions.

The IOC3 sets up a security flow table (IPv4 and IPv6) including key, result table, and packet memory.

The following functions are provided with the flow table:

- Flow lookup
- Flow insertion and deletion
- Security flow aging out
- Security flow statistics

Understanding Packet-Ordering Function on SRX5000 Line Devices

IN THIS SECTION

The packet-ordering function, which is supported on the SRX5400, SRX5600, and SRX5800, devices and vSRX Virtual Firewall, improves the performance of the device by activating the built-in packet-ordering function of the Packet Ordering Engine on the XLP processor on the application central point.

Two types of the packet ordering modes are supported: hardware and software.

If the packet-ordering function is set to *hardware*, the load-balancing thread (LBT) and the packet-ordering thread (POT) are offloaded to the packet ordering engine and resources are freed to perform packet processing. If the packet-ordering function is set to *software*, the load-balancing thread (LBT) and the packet-ordering thread (POT) are running on the SPU. By default, packet-ordering mode using the Packet Ordering Engine (hardware) is enabled on the device. You can disable it with a configuration change that requires a reboot.

The flow thread receives the packets, processes them, and sends or drops them. For packets that require no ordering, the flow thread notifies the Network Acceleration Engine (NAE) egress to send or drop the packets. For packets that require ordering, the flow thread notifies the Packet Ordering Engine to dequeue the packets from the ordering list and to send or drop the packets in order.

Changing Packet-Ordering Mode on SRX5000 Line Devices

The packet-ordering functionality using the Packet Ordering Engine is supported on SRX5400, SRX5800 and SRX5600 devices with next-generation SPCs. (Platform support depends on the Junos OS release in your installation.) By default, packet-ordering mode using the Packet Ordering Engine is enabled. To disable the packet-ordering functionality using the Packet Ordering Engine, you must update the packet-ordering mode on the device.

The following packet ordering modes are supported:

- software—Disables the packet-ordering mode using the Packet Ordering Engine.
- hardware—Enables the packet-ordering mode using the Packet Ordering Engine. This is the default option.

To disable the packet-ordering mode using the Packet Ordering Engine:

1. Enter the following command at the CLI configuration prompt to specify the packet-ordering mode.

```
[edit]
user@host#  set security forwarding-process application-services packet-ordering-mode software
```

2. Use the `show security forwarding-process` command to review your configuration.

```
[edit]
user@host# show security forwarding-process
```

```
application-services{
    packet-ordering-mode software;
}
```

3. Check your changes to the configuration before committing.

```
[edit]
user@host# commit check
```

```
warning: System packet ordering mode changed, reboot is required to take effect.
If you have deployed a cluster, be sure to reboot all nodes.
configuration check succeeds
```

4. Commit the configuration.

```
[edit]
user@host# commit
```

```
warning: System packet ordering mode changed, reboot is required to take effect.
If you have deployed a cluster, be sure to reboot all nodes.
commit complete
```

5. Reboot the device at an appropriate time.

6. Use the `show security flow status` command to verify the packet-ordering mode.

```
user@host> show security flow status

  Flow forwarding mode:
    Inet forwarding mode: flow based
    Inet6 forwarding mode: drop
    MPLS forwarding mode: drop
    ISO forwarding mode: drop
  Flow trace status
    Flow tracing status: off
  Flow session distribution
    Distribution mode: RR-based
```

```
Flow packet ordering
  Ordering mode: Software (reboot needed to change to Software)
```

Understanding Session Distribution on SRX5000 Line Devices in Adaptive Mode

Starting in Junos OS Release 15.1X49-D30 and Junos OS Release 17.3R1, adaptive mode session distribution was replaced by enhancements to the central point architecture.

Adaptive mode session distribution is implemented on the SRX5000 line devices running in mixed mode prior to Junos OS Release 15.1X49-D30 and Junos OS Release 17.1R1. Adaptive mode session distribution maximizes use of system resources by taking into account a Services Processing Unit's (SPU) capacity and its available resources. It is enabled only on SRX5000 line devices running in XLR/XLP mixed mode, that is in chassis deployments in which different types of SPUs are used in different combinations. If an SRX5800, SRX5600, or SRX5400 device contains a mix of next-generation services processing cards (SPCs) and existing SPCs, then adaptive mode session distribution is assumed as the default. For SRX5000 line devices not running in mixed mode, hash-based load balancing is the default.

A Services Processing Card (SPC) contains one or more SPUs each of which processes the packets of a flow according to the security features and other services configured for sessions distributed to it by the central point (CP). An SPU's CPU load changes from time to time. To fully utilize changing available capacity and adapt session distribution accordingly, in adaptive mode the system assigns a weight to all SPUs dynamically. It is the weight of the SPUs that determine the session distribution.

Each SPU sends its CPU usage information to the central point (CP) periodically. The central point checks these values, calculates the weight every 1 second, and distributes the sessions in such a way as to maximize overall system performance. In other words, In adaptive mode, session distribution is based on a *dynamic* weighted assignment system that is calculated in real time allowing for full capacity utilization of the CPUs of all SPUs, regardless of their type.

It is the dynamic calculation of weights that distinguishes adaptive mode session distribution from weighted round-robin (WRR) session distribution. While WRR differentiates SPUs and their CPU capacity by calculating and assigning weights to the different types of SPUs, the calculation and assignment is static, that is, it is done only once, at initialization. Adaptive mode improves on the fixed ratio session distribution process of WRR. WRR leads to underutilization of system resources because session processing limits are set based only on the type of SPU and its CPU capacity, not taking into account its available processing power.

For adaptive mode session distribution, the following formula is used to calculate the weight assigned to an SPU:

Wi = Sum(W1-n)*Ci*Si/Sum(C1-n*S1-n)

Where:

- `Wi`— weight assigned to the SPU.
- `Sum(W1-n)`— Total weight of system. This values is constant.
- `n`—total number of SPUs.
- `Ci`—available CPU computational power of the SPU.
- `Si`—available session capacity of SPU.

In adaptive mode, when the CPU usage on one SPU is high, fewer sessions are distributed to that SPU. The following examples explains the calculation.

Consider a device with two SPUs. Each SPU's session capacity is 1 million.

For a certain time:

- When SPU1 has 500,000 sessions on it, CPU usage of it is 10 percent:
  - Available CPU capacity of SPU1 (C1) = 1-10 percent = 90 (percent).
  - Available session capacity of SPU1 (S1) = 1-500,000/1M = 50 (percent).
- When SPU2 has 400,000 sessions on it, CPU usage of it is 20 percent:
  - Available capacity of SPU2 (C2)= 1-20 percent= 80 (percent).
  - Available session capacity of SPU2 (S2)= 1-400,000/1M= 60 (percent).

If the weight of the whole system is 100, the separate weight values for each SPU are:

- Weight of SPU1 (W1) = 100*90*50/(50*90+80*60) = 48
- Weight of SPU2 (W2) = 100*80*60/(50*90+80*60) = 52

For the incoming sessions, 48 percent of session are allocated to SPU1 while 52 percent of packets are allocated to SPU2.

The weighted numbers might take effect on the system within a short period before the central point checks the runtime usage information and adjusts the weights to a new value.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
17.4R1	Starting with Junos OS Release 17.4R1, traffic is hashed and distributed to different SPUs by the IOC, based on a hash-based session distribution algorithm. This enhancement provides an even hash distribution among all SPUs by using a larger fixed-length hash table. In earlier Junos OS releases, the traffic distribution was uneven among all SPUs due to a fixed-length hash table.
15.1X49-D30	Starting in Junos OS Release 15.1X49-D30, the central point architecture is enhanced to handle higher concurrent sessions and connections per second (cps) for the SRX5000 line device.
15.1X49-D30	Starting in Junos OS Release 15.1X49-D30 and Junos OS Release 17.3R1, adaptive mode session distribution was replaced by enhancements to the central point architecture.
15.1X49-D10	Starting with Junos OS Release 15.1X49-D10 and Junos OS Release 17.3R1, hash-based session distribution is the default mode for the SRX5400, SRX5600, and SRX5800 devices. Selection of hash keys depends on application protocols.

RELATED DOCUMENTATION

Flow-Based Sessions | 45

TCP Sessions | 52

ECMP Flow-Based Forwarding | 65

PowerMode

IN THIS SECTION

PowerMode

PowerMode is a new default dataplane framework that introduces an optimized fast-path allowing for higher throughput and lower latency on SRX Series Firewalls. PowerMode is able to accelerate IPsec operations and generic TCP and UDP flows in the same manner as Express Path on Trio-Based platforms.

In Junos OS Release 21.3R1, the feature has the following limitations:

- Non-IP protocol.
- IP protocols which are not TCP, UDP, ESP, SCTP and GTP.
- Multicast sessions.
- Egress Logical Tunnel (LT) interfaces and cross-lsys traffic.
- Sessions that require TCP-Proxy.
- Firewall Filters.
- Mac learning and transparent mode.
- Active/Active HA clusters when the sessions are transiting the fabric link known as Z-mode traffic.

NOTE: SRX Series Firewalls with PMI supports only flow-based CoS (Class of Service).

SEE ALSO

power-mode-disable

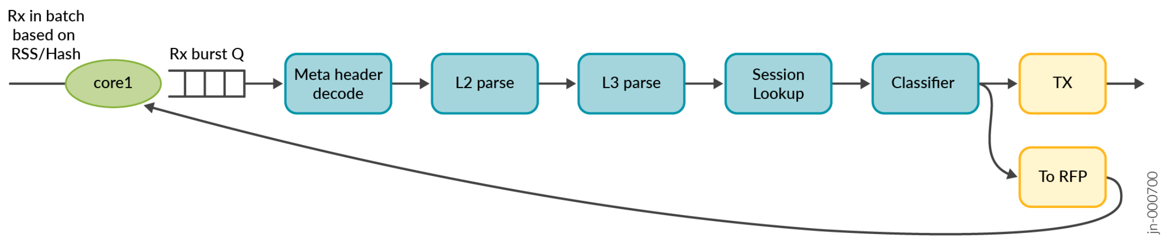
PowerMode Express

IN THIS SECTION

PowerMode Express Overview

PowerMode Express (PME) is a mode of operation that provides performance improvements using vector packet processing. PME use a small software block inside the Packet Forwarding Engine (PFE) that helps to process multiple packets in receive buffer thus better utilizing CPU cache.

Figure 11: Packet Flow in PowerMode Express



Benefits

- Improves fast path processing and UDP throughput performance.

On SRX4100, SRX4200, SRX4600, SRX5400, SRX5600, SRX5800, and vSRX devices, PowerMode Express is enabled by default from Junos OS Release 21.3R1. When you upgrade to Junos Release 21.3R1 or later, you unlock free, unparalleled next-generation firewall performance, without any additional configuration or hardware investment.

To disable the PowerMode Express globally, use the `set security flow power-mode-disable` command.

PowerMode Express supports:

- Class of Service (CoS)
- Network Address Translation (NAT)
- Screens (Anti-DDoS)
- Forwarding class
- Routing instance

PowerMode Express Limitations

PowerMode Express does not support:

- Non-IP protocol
- IP protocols which are not TCP, UDP, ESP, SCTP, and GTP
- Multicast sessions
- Egress Logical Tunnel (LT) interfaces and cross-lsys traffic
- Sessions that require policer, syslog, and counter
- Firewall filter
- Active/Active HA clusters when the sessions are transiting the fabric link known as Z-mode traffic

How does PowerMode Express Process the Traffic

When the first packet arrives at an interface, a new session is created in the PowerMode. If the new session qualifies for PowerMode Express, a PowerMode qualification check occurs.

The PowerMode Express session processes the fast-path packets in the network processor to jexec processing. In jexec layer2 forwarding stage, the cache next-hop, forwarding class, Class of Service (CoS) information allows the subsequent packets of the session for PowerMode Express.

How to Re-enable PowerMode Express

PowerMode Express is enabled by default. If you disable the PowerMode express, you can re-enable using the following command:

```
[edit]
user@host# delete security flow power-mode-disable
user@host# commit
```

Verification

To confirm that the configuration is working properly, enter the following show command.

```
user@host# show security flow status | grep "Flow power mode:"
  Flow power mode: Enabled
```

SEE ALSO

traceoptions (Security Flow)

PowerMode IPsec

PowerMode IPsec (PMI) is a new mode of operation for SRX4100, SRX4200, SRX4600, SRX5400, SRX5600, SRX5800, and vSRX Virtual Firewall instances to improve IPsec performance. Starting with Junos OS Release 19.1R1, the PMI is enhanced to handle the incoming and outgoing fragment packets using first path or fast path processing.

Enable the PMI process by using the `set security flow power-mode-ipsec` command. To verify that the packets are leveraging PMI, use the show security flow pmi statistics command.

Understanding PMI First Path and Fast Path Processing

In a PMI first path processing:

- The incoming first path packet is delivered to flow to create session.
- The incoming fragment packets are delivered to flow for reassembling.
- The incoming packets are delivered to flow for advanced security service processing.

In a PMI fast path processing, the PMI driver is used:

- To encrypt and send out the incoming clear text directly.
- To decrypt and send out the incoming ESP packets directly with session match.

Switching between PMI First Path and Fast Path Processing

The first path processing involves more features and instructions, while the PMI fast path processing provides better performance. In a PMI session, the packet processing switches between first path and fast path based on the packets flow in the session.

- The PMI session with both fragment and non-fragment packets are processed by first path.
- When the session only has non-fragmented packets, the session will switch from first path to fast past processing.

NOTE: On SRX5400, SRX5600, and SRX5800 devices, switching happens after the NP session timeout.

Fragmentation for Incoming IP Packets

To support fragmentation for incoming IP packets for PMI, following steps are used in first path:

- PMI transmits all the fragmented IP packets in a session to the flow module for processing.
- PMI transmits all the non-fragmented IP packets in the same session to the flow module for packet ordering.
- The Flow module completes reassembly of fragmented packets and transmits the packets back to PMI for encryption.

Fragmentation for Outgoing IP Packets

To support fragmentation for outgoing IP packets for PMI, following steps are used:

- PMI detects clear text packets that requires fragmentation during session lookup and delivers packets to the flow module.
- Flow module does fragmentation for outgoing packets.
- PMI encrypts the packets before transmitting them.

NP session

A PMI session or non-PMI session has no NP session installed due to limited NP session capacity then the packet ordering for this PMI session may not be available.

Use Feature Explorer to confirm platform and release support for specific features.

Review the "Platform-Specific NP Session Behavior" on page 94 section for notes related to your platform.

Platform-Specific NP Session Behavior

Use Feature Explorer to confirm platform and release support for specific features.

Use the following table to review platform-specific behavior for your platform:

Platform	Difference
SRX Series Firewall	• On SRX4100, SRX4200, and vSRX Virtual Firewall devices that support NP session, supports the fragment and non-fragment packets that are hashed to the same CPU core for processing. Hence, NP session is not supported. • On SRX5400, SRX5600, and SRX5800 devices with SPC3 that support NP session supports, fragment and non-fragment packets that are hashed to different CPU cores for processing. Hence, NP session is supported to deliver fragment or non-fragment packets to the same core for ordering.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
19.1R1	Starting with Junos OS Release 19.1R1, the PMI is enhanced to handle the incoming and outgoing fragment packets using first path or fast path processing.

Unified Policies Support for Flow

IN THIS SECTION

Starting in Junos OS Release 18.2R1, unified policies are supported on SRX Series Firewalls, allowing granular control and enforcement of dynamic Layer 7 applications within the security policy. Unified policies are the security policies that enable you to use dynamic applications as match conditions as part of the existing 5-tuple or 6-tuple (5-tuple with user firewall) match conditions to detect application changes over time.

Unified policies allow you to use dynamic application as a policy match criteria in the application. On applying Application Identification (AppID) to the traffic, the AppID checks several packets and identifies the application. After the application is identified, the final policy is applied to the session. The policy actions such as permit, deny, reject, or redirect are applied to the traffic as per the policy.

During the initial policy lookup phase, which occurs prior to a dynamic application being identified, if there are multiple policies in the potential policy list, the SRX Series Firewall applies the default security policy until a more explicit match has occurred. The policy that best matches the application is the final policy.

For more information on unified policies, See [Unified Security Policies, Application Identification Support for Unified Policies, and Understanding IDP Policy Support for Unified Policies.]

Flow First Path for Unified Policies

When the device examines the first packet of a flow, it determines the corresponding security policy, and performs a security policy lookup. During this process following cases are observed:

- If the traffic matches a legacy security policy or the final policy, the session is created.
- If there are multiple policies in the potential policy list and there is a security policy conflict, then the default security policy is applied.

- If there are multiple policies in the potential policy list, and the policy action does not permit the traffic, then the session is closed. A log message is generated to indicate the reason for the session closure. The default security policy is required during policy conflict stage, because each policy in the potential policy list has different configuration values for MSS, TCP SYN check, session timeout interval, and so on. In this case, when the default security policy is applied, all the values configured in that policy are applied. When a default security policy is matched, the policy actions are applied for the session.
    - The default security policy is system-defined policy. This policy cannot be deleted.
    - The default policy is created on every logical system level, similar to the global default policy.
    - The session timeout interval and session log values are leveraged from the default security policy and default values such as TCP-MSS and TCP SYN are leveraged from the flow configuration.
- When a default policy is applied, a potential metadata for the policy action is allocated. The potential metadata is updated according to the potential policy list.
    - Having a default security policy helps in resolving in the potential policy list.
    - There can be many sessions matching the default security policy; however, the application services defined in the policy for the permitted traffic can be different. The security flow information for each session is saved.
    - When an SRX Series Firewall is operating in chassis cluster mode, the information is synchronized from the primary node to the secondary node along with the flow session and the chassis cluster real time objects (RTO).
- When the final application is identified, the security policy matching with the final application is applied. The subsequent packets are processed according to the final policy.

Understanding Flow Fast Path

After the first packet in a flow has traversed the device and a session has been established for it, it undergoes fast path processing. When the device examines a security flow session with default policy, it performs a security policy lookup and following cases are observed:

- If the existing Application Identification requires an update, the policy lookup process in repeated. The process is repeated until an explicit policy is returned and replaced in the security flow session. If an implicit policy is returned, the traffic is denied and the session is closed.
- When the final application is identified, the final policy matching the traffic is applied. If the policy actions in the default and the final policy are similar, the final policy replaces the default policy in the

security flow session. If the policy actions in the default and the final policy are different, default policy is retained and the security flow session is closed.

NOTE: When the final and the default policy with a deny action is matched, the security flow session is closed.

- To update a session, the session timeout, log, or counter configuration in the final policy is used.

Configuring the Session Log for the Default Security Policy

The default security policy is required to manage policy conflicts in the potential policy list. You can set the session logs for the required sessions in default security policy configurations:

You can enable logging at the end of a session and at the beginning of the session with the following commands:

1. Generate a Session_Create log for policies entering the global pre-id-default-policy.

```
[edit]
user@host# set security policies pre-id-default-policy then log session-init
```

CAUTION: Configuring `session-init` logging for the `pre-id-default-policy` can generate a large amount of logs. Each session that enters the SRX that initially matches the `pre-id-default-policy` will generate an event. We recommend only using this option for troubleshooting purposes.

2. Generate a Session_Close log for policies which close without exiting the global pre-id-default-policy.

```
[edit]
user@host# set security policies pre-id-default-policy then log session-close
```

We recommend enabling session-close logging within the pre-id-default-policy. This will ensures that security logs are generated by the SRX if a flow is unable to leave the pre-id-default-policy. These events are generally a result of Juniper Networks Deep Packet Inspection (JDPI) being unable to classify traffic properly. The events might also indicate potential attempts at evading the application identification (AppID) engine.

Configuring the Session Timeout for the Default Security Policy

You can set the session timeout for the required sessions in default security policy configurations. You can specify the timeout values for UDP, TCP, ICMP, and ICMP6 sessions using the `set security policies pre-id-default-policy then session-timeout` command:

- Specify the timeout value in seconds for the TCP session:

  ```
  [edit]
  user@host# set security policies pre-id-default-policy then session-timeout tcp 1200
  ```

- Specify the timeout value in seconds for the UDP session:

  ```
  [edit]
  user@host# set security policies pre-id-default-policy then session-timeout udp 60
  ```

- Specify the timeout value in seconds for the ICMP session:

  ```
  [edit]
  user@host# set security policies pre-id-default-policy then session-timeout icmp 60
  ```

- Specify the timeout value in seconds for the ICMP6 session:

  ```
  [edit]
  user@host# set security policies pre-id-default-policy then session-timeout icmp6 120
  ```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.2R1	Starting in Junos OS Release 18.2R1, unified policies are supported on SRX Series Firewalls, allowing granular control and enforcement of dynamic Layer 7 applications within the security policy. Unified policies are the security policies that enable you to use dynamic applications as match conditions as part of the existing 5-tuple or 6-tuple (5-tuple with user firewall) match conditions to detect application changes over time.

RELATED DOCUMENTATION

Flow-Based Sessions | 45

TAP Mode for Flow Sessions

IN THIS SECTION

In TAP mode, an SRX Series Firewall will be connected to a mirror port of the switch, which provides a copy of the traffic traversing the switch. An SRX Series Firewall in TAP mode processes the incoming traffic from TAP interface and generates a security log to display the information on threats detected, application usage, and user details.

Understanding TAP Mode Support for Security Flow Sessions

Starting in Junos OS Release 18.3R1, TAP mode supports security flow sessions. The security flow session configuration remains the same as non-TAP mode. When you configure a device to operate in TAP mode, the device generates a security log information to display the information on threats detected, application usage, and user details according to the incoming traffic. TAP mode is enabled in flow status when there is a configured TAP interface.

Traffic with and without VLAN can be received by TAP interface. By default, on all devices, the `FLOW SYN-check` and `sequence-check` options are disabled at `[set security]` hierarchy level.

Starting in Junos OS Release 20.1R1, TAP mode can be used to inspect at most two levels of embedding IP-IP tunnels and one level of embedding GRE tunnel by de-encapsulating the outer and inner IP header and creating flow sessions. You can configure up to eight TAP interfaces on an SRX Series Firewall.

Example: Configuring Security Flow Sessions in TAP mode

IN THIS SECTION

This example shows how to configure security flow sessions when the SRX Series Firewall is configured in TAP mode.

Requirements

This example uses the following hardware and software components:

- An SRX Series Firewall
- Junos OS Release 19.1R1

Overview

In this example, you configure the security flow sessions when the SRX Series Firewall is configured in TAP mode. Sessions are created when a TCP SYN packet is received and permitted by the security policy.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter commit from configuration mode.

```
set security flow tcp-session no-syn-check
set security flow tcp-session no-sequence-check
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in theCLI User Guide.

To configure security flow sessions in TAP mode:

1. Configure the security flow session.

   ```
   user@host#set security flow tcp-session no-syn-check
   user@host# set security flow tcp-session no-sequence-check
   ```

Results

From configuration mode, confirm your configuration by entering the `show security flow` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show security flow
    tcp-session {
        no-syn-check;
        no-sequence-check;
    }
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform these tasks:

Verifying Security Session Configuration in TAP Mode

Purpose

Verify information about security sessions.

Action

From operational mode, enter the `show security flow session` command.

```
user@host> show security flow session
node0:
--------------------------------------------------------------------------


Flow Sessions on FPC4 PIC0:
Total sessions: 0

Flow Sessions on FPC4 PIC1:
Total sessions: 0
```

Meaning

Displays information about all currently active security sessions on the device in TAP mode.

SEE ALSO

TAP Mode Support Overview for UTM

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
20.1R1	Starting in Junos OS Release 20.1R1, TAP mode can be used to inspect at most two levels of embedding IP-IP tunnels and one level of embedding GRE tunnel by de-encapsulating the outer and inner IP header and creating flow sessions.

Flow Management in Firewalls Using Virtual Routing and Forwarding

IN THIS SECTION

Virtual Routing and Forwarding Instances in SD-WAN Deployments

Virtual routing and forwarding (VRF) instances are required to separate the routes of each tenant from the route of other tenants and from other network traffic. Firewalls use VRF instances for segmenting networks for increased security and improved manageability in SD-WAN deployments. For example, you can create distinct routing domains called tenants to segment large corporate networks and segment traffic to support multiple customer networks. Each tenant has its own routing table, which enables the support for overlapping IP subnets. VRF can be used to manage routes and to forward traffic based on independent forwarding tables in VRF for a specific tenant.

In an SD-WAN deployment, a provider edge (PE) router can be both a hub device and a spoke device that receives and forwards MPLS traffic. A customer edge (CE) router is the firewall that interacts with a PE router to transmit VPN traffic using VRF routing instances. The VRF instances forward each customer VPN traffic and each VRF instance contains one label to represent all the customer traffic that flows through that VRF.

Different sites that connect to a spoke-side of the firewall can belong to the same tenant or to the same VRF routing instance. These sites send the IP traffic that is intended to reach either public Internet or remote tenant sites.

When the traffic reaches the spoke-side of the firewall, the device identifies the VRF instance from the LAN interfaces that are connected to those sites. After security processing on this traffic, the traffic finds a route to the destination in that VRF routing table. If the destination is MPLS over next-hop-based generic routing encapsulation (GRE), the firewall adds a corresponding MPLS label and forwards the packet to the hub-side device.

At the hub-side device, after receiving MPLS over GRE tunneled traffic, the firewall associates the MPLS label to identify the corresponding VRF routing instance. After security processing of the traffic is complete, the device identifies whether the destination is on public Internet or reachable through MPLS next-hop.

If the destination is public Internet, NAT converts VRF private IP address to a public IP address and establish the session. If the destination is a type of MPLS next-hop, corresponding MPLS label is added and the packet is forwarded to the remote spoke using a GRE overlay tunnel.

At the remote spoke side, after receiving the MPLS over GRE tunnel traffic, the device identifies the corresponding VRF routing-instance using the MPLS labels. Using that VRF routing instance, the firewall finds the correct destination LAN interface in that VRF to forward the packet to the destination.

Multicast Support in SD-WAN Deployments

Starting in Junos OS Release 21.2R1, we've added support for multicast traffic on SRX Series Firewalls in Provider Edge (PE) for SD-WAN deployments. The support for multicast traffic is available when the security device is operating with forwarding option set as flow-based in the `set security forwarding-options family mpls mode` hierarchy. See forwarding-options (Security).

Limitations

- Support for multicast traffic not available when the device is operating with forwarding option set as `packet-based`.
- Support for multicast traffic available for only hub-and-spoke topology with IP-over-MPLS-over-GRE and IP-over-MPLS-over-GRE-over-IPsec
- Support for multicast traffic does not introduce any changes or address any limitations related to multicast VPN (MVPN) data forwarding functionality. See Limiting Routes to Be Advertised by an MVPN VRF Instance.

Flow Management Using VRF Routing Instance

The firewall flow creates sessions based on 5-tuple data (source IP address, destination IP address, source port number, destination port number, and protocol number) along with interface tokens of input interface and output interface of traffic. For example, the routing instance VRF-1 and the routing instance VRF-2 have the same 5-tuple traffic that can enter and exit through the same physical GRE tunnel. When these overlapping IP addresses from the same tunnel enter or exit through the firewall, then the firewall flow cannot install multiple sessions in the database because of the conflict in session information. Additional information is required for the firewall to differentiate sessions during the installation.

The firewall can use VRF information from the MPLS-tagged packets in the session key to differentiate sessions. To differentiate sessions from different VRF instances, flow uses VRF identification numbers to the existing session key to identify each VRF instance. This session key is used as one of the matching criteria during session look-up.

You can use the following matching criteria along with existing 5-tuple matching conditions in a security policy to permit or deny traffic based on given VRF:

- Source VRF—The VRF routing instance associated with the incoming interface of the packet. When an incoming MPLS packet containing a label arrives at the firewall, the device decodes the label, and maps the label to the incoming interface.

- Destination VRF—The VRF routing instance associated with the final route to the destination. During the first packet processing for a new session, flow needs a destination route for routing a packet to the next-hop device or interface. Flow searches the initial routing table from either the incoming interface or from a separate RTT table until it finds the final next-hop device or interface. Once the final route entry is found, and if that route points to an MPLS next-hop device, then the destination VRF is assigned to the routing instance in which the final route is found.

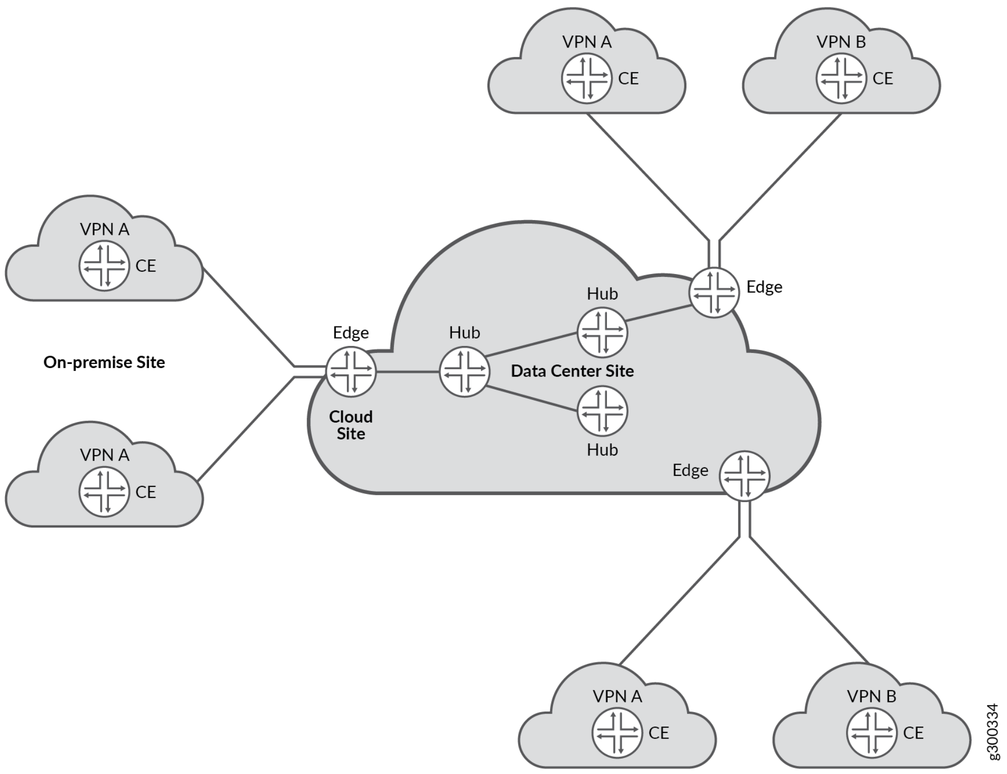
Flow Processing using Virtual Routing and Forwarding Group

IN THIS SECTION

Understanding Virtual Routing and Forwarding Groups

SD-WAN enterprise network is composed of multiple L3VPN networks as shown in No Link Title.

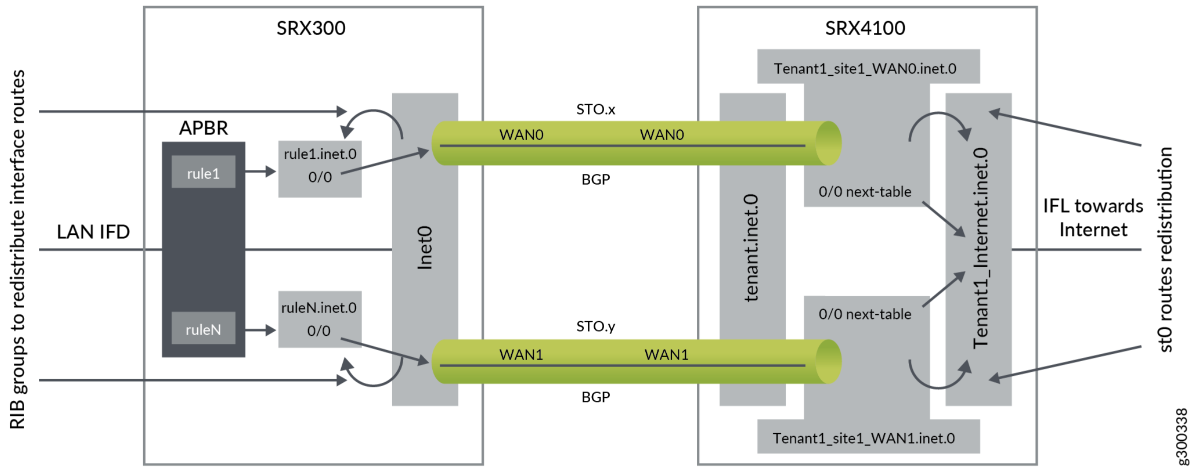
Figure 12: Multiple L3VPNs



L3VPN networks are identified at a site (CPE device) as a set of VRF instances. The VRF instances at a site belonging to a L3VPN network at a site are used for application policy based forwarding. Firewall flow session handling has enhanced to support mid-stream traffic switching between these VRF instances based on application based steering policies. The VRF instances which are logically part of a given L3VPN network can be configured as a VRF group. Existing firewall, NAT configuration commands have been enhanced to support operations on VRF group.

The No Link Title describes how traffic steering within an L3VPN is done across multiple VRFs based on APBR policies.

Figure 13:



When you configure the VRF groups using VRF instances, a VRF group-ID is generated. These VRF groups are used in the following modules to control SD-WAN L3VPN:

- **Security Policy** - For policy control.
- **Flow** - To search policies based on VRF group names, along with source or destination zone, source or destination IP address, and protocol. Hence, sessions are created using VRF groups as one of differentiator.
- **NAT** - To support NAT rules based on VRF group names.
- **ALG** - To create ALG sessions using VRF groups as one of differentiator.

The functionality of the VRF groups:

- It allows a session to switch between two MPLS VRFs.
- When the VRF instances are part of the same VRF group, security features such as flow, policy, NAT, or ALG modules treat the VRF instances similarly.
- When you configure the VRF groups using VRF instances, a VRF group-ID is generated. This group-ID is stored in session for identifying the VRF group of a particular VRF instance.

VRF group is introduced to support L3VPN MPLS based sessions in SD-WAN network. It is used to control the MPLS L3VPN traffic in policy, flow, NAT and ALG modules when there are overlapping or no overlapping IP network addresses in the MPLS L3VPN network.

If the traffic pass between non MPLS L3VPN networks, VRF groups are not configured. When VRF groups are not configured, the VRF group-ID will be zero or the policy will use the option `any` for VRF group.

The purpose of VRF groups is:

- To differentiate L3VPN sessions between MPLS L3VPN network.
- To have policy and NAT control between MPLS L3VPN network.

SEE ALSO

Flow Processing using Virtual Routing and Forwarding Group

Types of VRF groups

There are two important VRF group in L3VPN network are:

- Source-VRF group
- Destination-VRF group

To understand which VRF instances can be grouped together for Source-VRF group or Destination-VRF group, use the following information:

- **Source-VRF instances**—List of VRF instances that negotiates different MPLS paths to the same in-bound destination.
- **Destination-VRF instances**— List of VRF instances that contain the destination routes for a given L3VPN traffic.

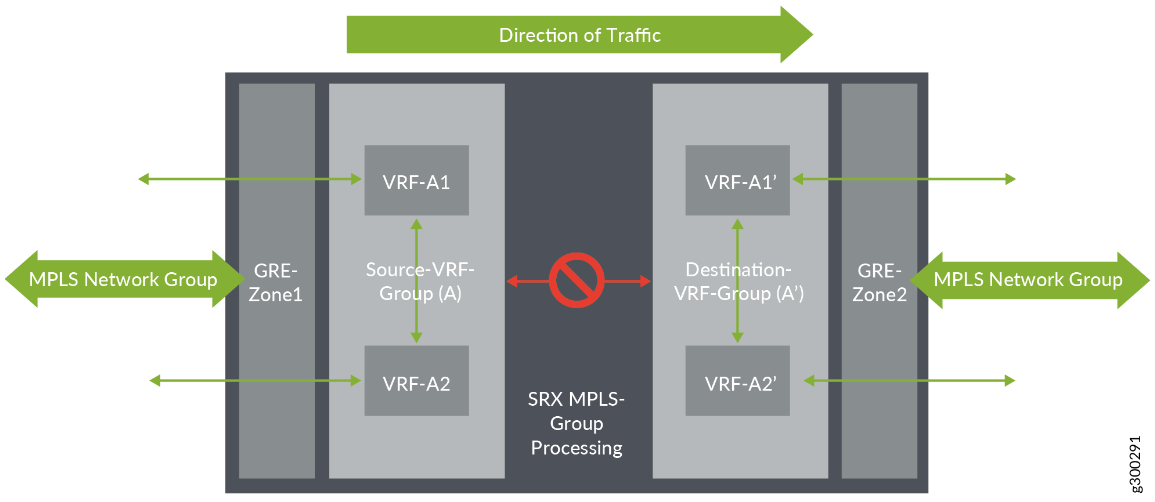
NOTE: If the traffic is initiated in the opposite direction, the VRF groups switch roles with respect to the direction of the traffic.

VRF Movement

From Figure 14 on page 109, the initial traffic flow for a session establishment is from left to right. The traffic enters GRE-Zone1, then enters Source-VRF group (A) and passes through Destination-VRF group (A’) before it exits through GRE_Zone2.

Similarly, the policy search is initiated from `GRE_Zone1->Source-VRF group(A)->Destination-VRF group->(A’)->GRE_Zone2` and the flow sessions is set-up, using Source-VRF group (A) and Destination-VRF group (A) as an additional key values in sessions. When the flow sessions are done using VRF groups, traffic can switch (re-route) from one VRF to another VRF within Group.

Figure 14: VRF Movement within VRF Group



VRF group-ID

For storing the VRF group-ID, a 16-bits number is used in a session key data structure.

Configuring VRF groups

To configure a VRF group, use the following steps:

- List the VRF instances that needs to be grouped.
- Assign a name to the VRF group.
- Apply the VRF instances and the VRF group name in the CLI command `set security l3vpn vrf-group` *`group-name`* `vrf` *`vrf1`* `vrf` *`vrf2`*

The source and destination VRF groups are configured separately based on different context.

- **Source VRF group**—The source VRF group for routing-instance is associated with MPLS packet. When the device receives a MPLS packet, the packet is decoded and mapped to LSI interface. The LSI interface contains the routing table information that helps in identifying the VRF group details.
- **Destination VRF group**—During first-path flow processing of packet for a new session, the destination route information is required to route the packet to the next-hop or interface. Flow searches the routing table to get the route information. When the received route information points to MPLS as next-hop, then the VRF of this route is used to identify the destination VRF group.

NOTE: The source and destination VRF groups are same in some cases when you prefer to control all the related VRFs in a L3VPN network.

VRF group Operations

IN THIS SECTION

When a VRF group is configured, a Group-ID is created which is unique for different VRF groups. You can perform different operations such as adding, removing, or modifying a VRF to a VRF group.

Adding VRF to a VRF group

When a VRF is added to a VRF group, the corresponding VRF group-ID is assigned to the VRF. When you add a VRF to VRF group, remember the following:

- A VRF can be added to only one VRF group. It cannot be a part of multiple VRF groups.
- A maximum of 32 VRFs are be configured in a VRF group.
- When a VRF is added, it impacts the existing session and a new session is created as per policy.
- When new sessions are created after adding a new VRF to VRF group, the sessions use the new VRF group-ID of the new VRF.

Removing VRF from a VRF group

When a VRF is removed from a VRF group, the VRF group-ID of that VRF group changes to zero but the VRF will still be available in the device. When you remove a VRF from a VRF group, it impacts the existing sessions in two ways:

- **Impacting existing sessions**—When a VRF is removed from the VRF group, the existing session is removed and a new session will be created as per policy

- **Match Traffic**—When a VRF is removed from VRF group, the VRF group-ID for that VRF changes to zero and hence will not match the session. The packet drops and a new session is created as per policy.

When a VRF is removed from the VRF group, the new session that is processed using the impacted VRF installs a new VRF group-ID. This VRF group-ID will be zero, or a new Group-ID is created if you add the VRF to a new VRF group

Modifying VRF group

Modifying a VRF group involves the following operations:

- **Changing VRF group name**: When you change the VRF group name, the policy module scans the existing sessions to verify if the new VRF group name matches the existing rules.
- **Adding VRF to VRF group**: When a VRF is added to a VRF group, the corresponding VRF group-ID is assigned to the VRF.
- **Removing VRF from VRF group**: When a VRF is removed from a VRF group, the VRF group-ID of that VRF changes to zero and still the VRF will be available in the device.

Removing VRF group

When you remove a VRF group using CLI, a session scan will be performed on the existing sessions to match the VRF group that is removed. If the session match the removed VRF group, then that session is removed from the device by setting an invalid timeout. For sessions that does not match the removed VRP-Group-ID are not impacted.

First Path Processing using VRF Group

To process a packet, the first path processing performs the following:

- **MPLS Decoder**—When flow receives a MPLS or non-MPLS packet, the packet is processed to retrieve the details of the incoming packet, interface, and routing-instance of the incoming interface.
- **FBF configuration**—When you configure FBF rules to re-direct the incoming packets to different routing-instance, the FBF rule finds the routing-instance information and pass the FBF routing-instance information instead of packet incoming interface routing-instance. This FBF VRF should be a part of VRF group to control the L3VPN network.
- **Initialize Routing-Table**—When the flow receives the packet, the initial routing-table for the packet is created. If the FBF configuration matches the firewall filters, then the routing-instance information from FBF is used for route look-up. Else, flow uses the incoming interface routing-instance information for route look-up.

- **Finding Source VRF group**—If the incoming packet is from MPLS network, then the packet is mapped to the VRF instance of source VRF group. If the incoming packet is not MPLS packet, then the source VRF group id is zero.
- **Destination NAT using VRF group**—Flow checks if the destination IP needs NAT translation. Destination NAT supports two types of match criteria for VRF:
  - NAT rule search using VRF routing-group.
  - NAT rule result using VRF routing-instance and NAT information.
- **Destination Route**—The route look-up which is done in initial route table is used to identify the outgoing interface and destination-VRF information. This information is used in policy search and session installation.
- **Final next-hop**—The first step in finding destination route is to find final the next-hop of the pointed route. Using this next-hop, flow will check if the next-hop points to MPLS network or not. If it is not pointing to MPLS network, the destination VRF group will be zero.
- **Destination VRF group**— When the destination VRF is identified, the destination VRF Group-ID is initialized. If the destination VRF is not assigned to any group, it is set to zero.
- **First Path Policy Search**—Flow performs policy search to check if the packet needs to be permitted or denied. Flow gathers the 5-tuple policy-key information and VRF information and this information is used by policy search module to find the appropriate VRF policy.
- **Source NAT using VRF group**—Flow session does source NAT using source VRF group NAT rule search. Source-NAT supports two types of NAT search criteria.
  - Source-NAT rule search using VRF group.
  - Static-NAT rule search using VRF group or VRF instance.
- **Static NAT using VRF group or VRF instance**—Static NAT supports routing-group in rule-set and routing-instance in rule with VRF type.
  - When static NAT matches as destination NAT translation for a given IP packet, the VRF routing-group will be one of the match criteria and the VRF routing-instance ill be used as destination routing table.
  - When static NAT matches as source NAT translation for a given IP packet, the VRF routing-instance will be one of the match criteria.
- **Session Installation using VRF group**—During session installation process, source VRF group-ID is stored in forward-wing indicating that the wing points MPLS network. The destination VRF group-ID that is found from route look-up is stored in reverse-wing indicating that the wing points MPLS network.

- **Re-routing using VRF group**—Once the session is established using VRF group information, re-route is initiated if the interface is down or initial route is not available. These changed routes should be part of same VRF group (Source-VRF group/Destination-VRF group), in which the session is initially established on either side. Else, traffic will not match session and future traffic of session might get dropped or create new sessions as per policy.

Fast Path Processing using VRF Group

The fast path processing performs the following steps to process a packet.

- **MPLS Decoder**—When a packet MPLS or non-MPLS packet is received, the packet undergoes MPLS processing. When the processing is complete, the flow receives the details of the incoming packet, interface, and routing-instance of the incoming interface.

- **FBF configuration**—When you configure FBF rules to re-direct the incoming packets to different routing-instance, the FBF rule finds the routing-instance information and pass the FBF routing-instance information instead of packet incoming interface routing-instance. This FBF VRF should be a part of VRF group to control the L3VPN network.

- **Session look-up using VRF Group-ID**—During session look-up process, flow checks whether to pass the VRF Group-ID in session key for look-up. If the incoming interface is MPLS, flow will pass the VRF Group-ID information of the mapped VRF routing-instance to session key along with other key tuple information. If the incoming interface is not MPLS, the VRF Group-ID will be zero.

- **Session Route change**—If the route changes for session in mid-stream, flow checks for the new VRF that belongs to this route. If the new VRF Group-ID differs from the VRF Group-ID of the session, then the route will not be processed and the future packets are dropped. Hence, for re-routing the new route should belong to a VRF that belongs to session VRF Group.

- **VRF Group policy change**—When VRF group session policy is changed due to policy attributes such as zone/interface/IP/Source-VRF group/Destination-VRF group, the policy will be re-matched for the same session by supplying policy 5-tuple along with source VRF group and destination VRF group values to check if the policy is valid or not. Upon re-match, if policy does not match the session information, then the session terminates.

- **VRF session display**—Source-VRF Group and Destination-VRF Group are displayed in session output display to differentiate different VRF group for the same tuple.

- **High Availability**—High availability is supported with no behavior change when additional VRF group-ID information is synchronized to the HA peer node for differentiating different VRF group in the session

Flow Processing with Virtual Routing and Forwarding Aware Zones

In session management, we now integrate zone IDs into the session tuple. The VRF group ID is replaced by the zone ID, which will be cached and subsequently passed down for further processing.

Zone-Based session management maintains alignment with network configurations by accurately associating sessions with the correct interface and overlay zones.When a zone configuration change occurs, such as an alteration to an interface or overlay zone, the system automatically terminates sessions linked to the affected zones, maintaining consistency and reliability across the session management framework. Only the sessions associated with modified zones are terminated, while those linked to unaffected zones continue to function seamlessly.

SEE ALSO

Configuring Security Policies for a VRF Routing Instance

show security flow session

3

CHAPTER

Flow-Based Processing for IPv6

IN THIS CHAPTER

IPv6 Flow-Based Processing

IN THIS SECTION

This topic covers information on flow processing for IPv6 traffic and IPv6 sessions.

IPv6 Advanced Flow

IN THIS SECTION

IPv6 advanced flow adds IPv6 support for firewall, NAT, NAT-PT, multicast (local link and transit), IPsec, IDP, JSF framework, TCP Proxy, and Session manager on SRX Series Firewalls. MIBs are not used in the IPv6 flow.

In order to avoid the impact on the current IPv4 environment, IPv6 security is used. If IPv6 security is enabled, extended sessions and gates are allocated. The existing address fields and gates are used to store the index of extended sessions or gates. If IPv6 security is disabled, IPv6 security-related resources are not allocated.

New logs are used for IPv6 flow traffic to prevent impact on performance in the existing IPv4 system.

The behavior and implementation of the IPv6 advanced flow are the same as those of IPv4 in most cases.

The implementations of sessions, gates, ip-actions, processing of multithread, distribution, locking, synchronization, serialization, ordering, packet queuing, asynchronous messaging, IKE traffic issues, sanity check, and queues for IPv6 are similar to IPv4 implementations.

Some of the differences are explained below:

- **Header Parse** IPv6 advanced flow stops parsing the headers and interprets the packet as the corresponding protocol packet if it encounters the following extension headers:

  - TCP/UDP
  - ESP/AH
  - ICMPv6

  IPv6 advanced flow continues parsing headers if it encounters the following extension headers:

  - Hop-by-Hop
  - Routing and Destination, Fragment

  IPv6 advanced flow interprets the packets as an unknown protocol packet if it encounters the extension header **No Next Header**

- **Sanity Checks—** IPv6 advanced flow supports the following sanity checks:

  - TCP length
  - UDP length
  - Hop-by-hop
  - IP data length error
  - Layer 3 sanity checks (for example, IP version and IP length)
  - **ICMPv6 Packets** In IPv6 advanced flow, the ICMPv6 packets share the same behavior as normal IPv6 traffic with the following exceptions:

    - Embedded ICMPv6 packet
    - Path MTU message

- **Host Inbound and Outbound Traffic** IPv6 advanced flow supports all route and management protocols running on the Routing Engine (RE), including OSPF v3, RIPng, Telnet, and SSH. Note that no flow label is used in the flow.

- **Tunnel Traffic** IPv6 advanced flow supports the following tunnel types:

  - IPv4 IP-IP

  - IPv4 GRE
  - IPv4 IPsec
  - Dual-stack lite
- **Events and Logs** The following logs are for IPv6-related flow traffic:
  - RT_FLOW_IPVX_SESSION_DENY
  - RT_FLOW_IPVX_SESSION_CREATE
  - RT_FLOW_IPVX_SESSION_CLOSE

Understanding Sessions for IPv6 Flow

This topic gives an overview of flow-based sessions.

Most packet processing occurs in the context of a flow, including management of policies, zones, and most screens. A session is created for the first packet of a flow for the following purposes:

- To store most of the security measures to be applied to the packets of the flow.
- To cache information about the state of the flow. For example, logging and counting information for a flow is cached in its session. (Also, some stateful firewall screens rely on threshold values that pertain to individual sessions or across all sessions.)
- To allocate resources required for features for the flow.
- To provide a framework for features such as Application Layer Gateways (ALGs).

Understanding IPv6 Flow Processing on SRX5400, SRX5600, and SRX5800 devices

This topic introduces the architecture for the SRX5400, SRX5600, and SRX5800 devices. Flow processing on these devices is similar to that on branch SRX Series Firewalls.

These devices include I/O cards (IOCs) and Services Processing Cards (SPCs) that each contain processing units that process a packet as it traverses the device. These processing units have different responsibilities.

- A Network Processing Unit (NPU) runs on an IOC. An IOC has one or more NPUs. An NPU processes packets discretely and performs basic flow management functions.

  When an IPv6 packet arrives at an IOC, the packet flow process begins.

- The NPU performs the following IPv6 sanity checks for the packet:
    - For the IPv6 basic header, it performs the following header checks:
        - Version. It verifies that the header specifies IPv6 for the version.
        - Payload length. It checks the payload length to ensure that the combined length of the IPv6 packet and the Layer 2 header is shorter than the Layer 2 frame length.
        - Hop limit. It checks to ensure that the hop limit does not specify 0 (zero).
        - Address checks. It checks to ensure that the source IP address does not specify ::0 or FF::00 and that the destination IP address does not specify ::0 or ::1.
    - The NPU performs IPv6 extension header checks, including the following:
        - Hop-by-hop options. It verifies that this is the first extension header to follow the IPv6 basic header.
        - Routing extension. It verifies that there is only one routing extension header.
        - Destination options. It verifies that no more than two destination options extension headers are included.
        - Fragment. It verifies that there is only one fragment header.

    NOTE: The NPU treats any other extension header as a Layer 4 header.

    - The NPU performs Layer 4 TCP, UDP, and ICMP6 protocol checks, including the following:
        - UDP. It checks to ensure that IP Payload Length packets, other than a first-fragment packet, are at least 8 bytes long.
        - TCP. It checks to ensure that IP Payload Length packets, other than a first-fragment packet, are at least 20 bytes long.
        - ICMPv6. It checks to ensure that IP Payload Length packets, other than a first-fragment packet, are at least 8 bytes long.
- If the packet specifies a TCP or a UDP protocol, the NPU creates a tuple from the packet header data using the following information:
    - Source IP address
    - Destination IP address
    - Source port

    - Destination port
    - Protocol
    - Virtual router identifier (VRID)

      The device looks up the VRID from a VRID table.

  - For Internet Control Message Protocol version 6 (ICMPv6) packets, the tuple contains the same information as used for the TCP and the UDP search key, except for the source and destination port fields. The source and destination port fields are replaced with the following information extracted from the ICMPv6 packet:
    - For ICMP error packets: The pattern "0x00010001"
    - For ICMP information packets: The type, or code, field identifier
  - For packets with an Authentication Header (AH) or an Encapsulating Security Payload (ESP) header, the search key is the same as that used for the TCP and the UDP tuple, except for the source and destination port fields. In this case, the security parameter index (SPI) field value is used instead of the source and destination ports. For Encapsulating Security Payload (ESP) header and Authentication Header (AH), before enhancements to the cenral point architecture it is hashed by the 3-tuple and the security parameter index (SPI) field, after enhancements to the cenral point architecture it is hashed by an IP pair.
  - If a session exists for the packet's flow, the NPU sends the packet to the SPU that manages the session.
  - If a matching session does not exist,
    - The NPU sends the packet information to the central point, which creates a pending session.
    - The central point selects an SPU to process the packet and create sessions for it.
    - The SPU then sends session creation messages to the central point and the ingress and egress NPUs, directing them to create a session for the packet flow.
- A central point, which can run on a dedicated SPU, or share the resources of one if there is only one SPU. A central point takes care of arbitration and allocation of resources, and it distributes sessions in an intelligent way. The central point assigns an SPU to be used for a particular session when the SPU processes the first packet of its flow.
  - For SRX5000 line devices, the central point architecture is divided into two modules—the application central point and the distributed central point (DCP). The App-CP is responsible for global resource management and loading balancing, while DCP is responsible for traffic identification (global session matching). The App-CP functionality runs on the dedicated central point SPU, while the DCP functionality is distributed to the rest of the SPUs.

- One or more SPUs that run on a Services Processing Card (SPC). All flow-based services for a packet are executed on a single SPU, within the context of a session that is set up for the packet flow.

  The SPC for SRX5000 line devices has two SPUs.

  Several SPCs can be installed in a chassis.

  Primarily, an SPU performs the following tasks:

  - It manages the session and applies security features and other services to the packet.
  - It applies packet-based stateless firewall filters, classifiers, and traffic shapers.
  - If a session does not already exist for a packet, the SPU sends a request message to the NPU that performed the search for the packet's session, to direct it to add a session for it.

These discrete, cooperating parts of the system store the information identifying whether a session exists for a stream of packets and the information against which a packet is matched to determine if it belongs to an existing session.

Enabling Flow-Based Processing for IPv6 Traffic

You have the following options for handling IPv6 traffic:

- Drop—Do not forward IPv6 packets.
- Packet-based forwarding—Do not create a session and process according to packet-based features only (includes firewall filters and class of service).
- Flow-based forwarding—Create a session and process according to packet-based features (including firewall filters and class of service) but also flow-based security features, such as screens and firewall security policy. This is the default behavior.

To enable flow-based processing for IPv6 traffic, modify the `mode` statement at the `[edit security forwarding-options family inet6]` hierarchy level:

```
security {
    forwarding-options {
        family {
            inet6 {
                mode flow-based;
            }
        }
    }
}
```

The following example shows the CLI commands you use to configure forwarding for IPv6 traffic:

```
[edit]
user@host# set security forwarding-options family inet6 mode ?
Possible completions:
  drop                    Disable forwarding
  flow-based              Enable flow-based forwarding
  packet-based            Enable packet-based forwarding

[edit]
user@host# set security forwarding-options family inet6 mode flow-based
user@host# show security forwarding-options
family {
    inet6 {
        mode flow-based;
    }
}
```

If you change the forwarding option mode for IPv6, you might need to perform a reboot to initialize the configuration change. Table 6 on page 122 summarizes device status upon configuration change.

Table 6: Device Status Upon Configuration Change

Configuration Change	Commit Warning	Reboot Required	Impact on Existing Traffic Before Reboot	Impact on New Traffic Before Reboot
Drop to flow-based	Yes	Yes	Dropped	Dropped
Drop to packet-based	No	No	Packet-based	Packet-based
Flow-based to packet-based	Yes	Yes	None	Flow sessions created
Flow-based to drop	Yes	Yes	None	Flow sessions created
Packet-based to flow-based	Yes	Yes	Packet-based	Packet-based

Table 6: Device Status Upon Configuration Change *(Continued)*

Configuration Change	Commit Warning	Reboot Required	Impact on Existing Traffic Before Reboot	Impact on New Traffic Before Reboot
Packet-based to drop	No	No	Dropped	Dropped

Flow-Based Processing for IPv6 Traffic on Security Devices

Flow-based processing mode is required for security features such as zones, screens, and firewall policies to function. By default, the SRX Series Firewall is enabled for flow-based forwarding for IPv6 traffic on all devices, apart from the SRX300 Series devices that are set to drop mode. Starting with Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, for the SRX4100, SRX4200, SRX5400, SRX5600, SRX5800 and vSRX Virtual Firewall devices, you do *not* need to reboot the device when you are switching modes between flow mode, packet mode, and drop mode. For SRX300 Series devices, you *must* reboot the device when switching between flow mode, packet mode, and drop mode.

SRX300 Series Devices

When IPv6 is configured on SRX300 Series devices, the default behavior is set to drop mode because of memory constraints. In this case, you must reboot the device after changing the processing mode from the drop mode default to flow-based processing mode or packet-based processing mode—that is, between modes on these devices.

NOTE: For drop mode processing, the traffic is dropped directly, it is not forwarded. It differs from packet-mode processing for which the traffic is handled but no security processes are applied.

To process IPv6 traffic on SRX300 Series devices, you need to configure IPv6 addresses for the transit interfaces that receive and forward the traffic. For information about the inet6 protocol family and procedures for configuring IPv6 addresses for interfaces.

Configuring an SRX Series Device as a Border Router

When an SRX Series Firewall of any type is enabled for flow-based processing or drop mode, to configure the device as a border router you must change the mode to packet-based processing for MPLS. In this case, to configure the SRX Series Firewall to packet mode for MPLS, use the `set security forwarding-options family mpls mode packet-based` statement.

Enabling Flow-Based Processing for IPv6 Traffic on SRX300 Series Devices

To enable flow-based forwarding for IPv6 traffic on SRX300 Series devices, modify the mode at the [`edit security forwarding-options family inet6`] hierarchy level:

```
security {
    forwarding-options {
        family {
            inet6 {
                mode flow-based;
            }
        }
    }
}
```

To configure forwarding for IPv6 traffic on SRX300 Series device:

1. Change the forwarding option mode for IPv6 to flow-based.

   ```
   [edit]
   user@host# security forwarding-options family inet6 mode flow-based
   ```

2. Review your configuration.

   ```
   [edit]
   user@host# show security forwarding-options
       family {
           inet6 {
               mode flow-based;
           }
       }
   ```

3. Commit the configuration.

   ```
   [edit]
   user@host# commit
   ```

4. Reboot the device.

NOTE: For SRX300 Series devices, the device discards IPv6 type 0 Routing Header (RH0) packets.

Using Filters to Display IPv6 Session and Flow Information for SRX Series Services Gateways

IN THIS SECTION

Purpose

You can display flow and session information about one or more sessions with the `show security flow session` command. IPv6 sessions are included in aggregated statistics.

You can use the following filters with the `show security flow session` command: application, destination-port, destination-prefix, family, idp, interface, nat, protocol, resource-manager, session-identifier, source-port, source-prefix, and tunnel.

NOTE: Except for the session-identifier filter, the output of all the other filters can be viewed in brief, summary, and extensive mode. Brief mode is the default mode. The output of the session-identifier filter can be viewed only in the brief mode.

You can use the same filter options with the `clear security flow session` command to terminate sessions.

Action

The following examples show how to use IPv6-related filters to display summaries and details for IPv6 sessions.

Filtered summary report based on family

```
root> show security flow session summary family ?
Possible completions:
  inet                 Show IPv4 sessions
  inet6                Show IPv6/IPv6-NATPT sessions

root> show security flow session summary family inet6
Flow Sessions on FPC10 PIC1:
```

```
Valid sessions: 2
Pending sessions: 0
Invalidated sessions: 0
Sessions in other states: 0
Total sessions: 2

Flow Sessions on FPC10 PIC2:

Valid sessions: 1
Pending sessions: 0
Invalidated sessions: 0
Sessions in other states: 0
Total sessions: 1

Flow Sessions on FPC10 PIC3:

Valid sessions: 0
Pending sessions: 0
Invalidated sessions: 1
Sessions in other states: 0
Total sessions: 1
```

Filtered detailed report based on family

```
root> show security flow session family ?
Possible completions:
  inet                 Show IPv4 sessions
  inet6                Show IPv6/IPv6-NATPT sessions

root> show security flow session family inet6
Flow Sessions on FPC10 PIC1:
Total sessions: 0

Flow Sessions on FPC10 PIC2:
Total sessions: 0

Flow Sessions on FPC10 PIC3:

Session ID: 430000026, Policy name: default-policy-00/2, Timeout: 1794, Valid
  In: 2001:db8::10/64712 -> 2001:db8::4/21;tcp If: ge-7/1/0.0, Pkts: 8, Bytes: 562, CP Session
ID: 430000025
```

```
  Out: 2001:db8::4/21  --> 2001:db8::10/64712;tcp, If: ge-7/1/1.0, Pkts: 12, Bytes: 1014, CP
Session ID: 430000025
Total sessions: 1
```

Filtered brief report based on family

```
root> show security flow session family inet brief
Flow Sessions on FPC10 PIC1:

Session ID: 410000031, Policy name: default-policy-00/2, Timeout: 48, Valid
  In: 203.0.113.8/3 --> 198.51.100.11/43053;icmp, If: ge-7/1/0.0, Pkts: 1, Bytes: 84, CP Session
ID: 410000039
  Out: 198.51.100.11/43053 --> 203.0.113.8/3;icmp, If: ge-7/1/1.0, Pkts: 0, Bytes: 0, CP Session
ID: 410000039
Total sessions: 1

Flow Sessions on FPC10 PIC2:

Session ID: 420000034, Policy name: default-policy-00/2, Timeout: 48, Valid
  In: 203.0.113.8/4 --> 198.51.100.11/43053;icmp, If: ge-7/1/0.0, Pkts: 1, Bytes: 84, CP Session
ID: 420000041
  Out: 198.51.100.11/43053  --> 203.0.113.8/4;icmp, If: ge-7/1/1.0, Pkts: 0, Bytes: 0, CP
Session ID: 420000041
Total sessions: 1

Flow Sessions on FPC10 PIC3:

Session ID: 430000042, Policy name: default-policy-00/2, Timeout: 44, Valid
  In: 203.0.113.8/2 --> 198.51.100.11/43053;icmp, If: ge-7/1/0.0, Pkts: 1, Bytes: 84, CP Session
ID: 430000041
  Out: 198.51.100.11/43053 --> 203.0.113.8/2;icmp, If: ge-7/1/1.0, Pkts: 0, Bytes: 0, CP Session
ID: 430000041
Total sessions: 1



2001:dbf8::6:2/32
```

Filtered detailed report based on an IPv6 source-prefix

```
root> show security flow session source-prefix 2001:dbf8::
Flow Sessions on FPC10 PIC1:
```

```
Session ID: 410000066, Policy name: default-policy-00/2, Timeout: 2, Valid
  In: 2001:dbf8::6:2/3 > 2001:dbf8:5::2/7214;icmp6, If: ge-7/1/0.0, Pkts: 1, Bytes: 104, CP
Session ID: 410000076
  Out: 2001:dbf8:5::2/7214 --> 2001:dbf8:5::2/323;icmp6, If: .local..0, Pkts: 1, Bytes: 104, CP
Session ID: 410000076

Session ID: 410000068, Policy name: default-policy-00/2, Timeout: 2, Valid
  In: 2001:dbf8::6:2/4 --> 2001:dbf8:5::2/7214;icmp6, If: ge-7/1/0.0, Pkts: 1, Bytes: 104, CP
Session ID: 410000077
  Out: 2001:dbf8:5::2/7214 --> 2001:dbf8::6:2/4;icmp6, If: .local..0, Pkts: 1, Bytes: 104, CP
Session ID: 410000077
Total sessions: 2

Flow Sessions on FPC10 PIC2:

Session ID: 420000067, Policy name: default-policy-00/2, Timeout: 28, Valid
  In: 2001:dbf8::6:2/4 --> 2001:dbf8:5::3/6702;icmp6, If: ge-7/1/0.0, Pkts: 1, Bytes: 104, CP
Session ID: 420000080
  Out: 2001:dbf8:5::3/6702 --> 2001:dbf8::6:2/4 ;icmp6, If: ge-7/1/1.0, Pkts: 0, Bytes: 0, CP
Session ID: 420000080
Total sessions: 1

Flow Sessions on FPC10 PIC3:

Session ID: 430000077, Policy name: default-policy-00/2, Timeout: 28, Valid
  In: 2001:dbf8::6:2/3 --> 2001:dbf8:5::3/6702;icmp6, If: ge-7/1/0.0, Pkts: 1, Bytes: 104, CP
Session ID: 430000075
  Out: 2001:dbf8:5::3/6702 --> 2001:dbf8::6:2/3;icmp6, If: ge-7/1/1.0, Pkts: 0, Bytes: 0, CP
Session ID: 430000075

Session ID: 430000078, Policy name: default-policy-00/2, Timeout: 30, Valid
  In: 2001:dbf8::6:2/5 --> 2001:dbf8:5::3/6702, If: ge-7/1/0.0, Pkts: 1, Bytes: 104, CP Session
ID: 430000076
  Out: 2001:dbf8:5::3/6702 --> 2001:dbf8::6:2/5;icmp6, If: ge-7/1/1.0, Pkts: 0, Bytes: 0, CP
Session ID: 430000076

Session ID: 430000079, Policy name: default-policy-00/2, Timeout: 2, Valid
  In: 2001:dbf8::6:2/5 --> 2001:dbf8:5::1/7214;icmp6, If: ge-7/1/0.0, Pkts: 1, Bytes: 104, CP
Session ID: 430000077
  Out: 2001:dbf8:5::1/7214 --> 2001:dbf8::6:2/5;icmp6, If: .local..0, Pkts: 1, Bytes: 104, CP
```

```
Session ID: 430000077
Total sessions: 3
```

Multiple-filtered detailed report based on family, protocol and source-prefix

```
root> show security flow session family inet protocol icmp source-prefix 2001:db8::
Flow Sessions on FPC10 PIC1:

Session ID: 410000074, Policy name: default-policy-00/2, Timeout: 2, Valid
  In: 2001:dbf8::6:2/1 --> 2001:dbf8:8::2/26935;icmp, If: ge-7/1/0.0, Pkts: 1, Bytes: 84, CP
Session ID: 410000195
  Out: 2001:dbf8:8::2 --> 2001:dbf8::6:2/1;icmp, If: ge-7/1/1.0, Pkts: 1, Bytes: 84, CP Session
ID: 410000195
Total sessions: 1

Flow Sessions on FPC10 PIC2:

Session ID: 420000075, Policy name: default-policy-00/2, Timeout: 2, Valid
  In: 2001:dbf8::6:2/3 --> 2001:dbf8::6:2/26935;icmp, If: ge-7/1/0.0, Pkts: 1, Bytes: 84, CP
Session ID: 420000159
  Out: 2001:dbf8::6:2/26935 --> 2001:dbf8::6:2/3;icmp, If: ge-7/1/1.0, Pkts: 1, Bytes: 84, CP
Session ID: 420000159
Total sessions: 1

Flow Sessions on FPC10 PIC3:

Session ID: 430000085, Policy name: default-policy-00/2, Timeout: 2, Valid
  In:  2001:dbf8::6:2/4 --> 2001:dbf8::6:2/26935;icmp, If: ge-7/1/0.0, Pkts: 1, Bytes: 84, CP
Session ID: 430000083
  Out: 2001:dbf8::6:2/26935 -->  2001:dbf8::6:2/4;icmp, If: ge-7/1/1.0, Pkts: 1, Bytes: 84, CP
Session ID: 430000083
Total sessions: 1
```

Clearing all sessions, including IPv6 sessions

```
root> clear security flow session all
This command may terminate the current session too.
Continue? [yes,no] (no) yes

0 active sessions cleared
1 active sessions cleared
```

```
1 active sessions cleared
1 active sessions cleared
```

Clearing only IPv6 sessions

```
root> clear security flow session family ?
Possible completions:
  inet                 Clear IPv4 sessions
  inet6                Clear IPv6/IPv6-NATPT sessions

root> clear security flow session family inet6
0 active sessions cleared
1 active sessions cleared
1 active sessions cleared
1 active sessions cleared
```

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D70	By default, the SRX Series Firewall is enabled for flow-based forwarding for IPv6 traffic on all devices, apart from the SRX300 Series devices that are set to drop mode. Starting with Junos OS Release 15.1X49-D70 and Junos OS Release 17.3R1, for the SRX1500 series, SRX4100, SRX4200, SRX5400, SRX5600, SRX5800 and vSRX Virtual Firewall devices, you do *not* need to reboot the device when you are switching modes between flow mode, packet mode, and drop mode. For SRX300 Series devices, you *must* reboot the device when switching between flow mode, packet mode, and drop mode.
15.1X49-D30	Starting in Junos OS Release 15.1X49-D30 and Junos OS Release 17.3R1, many of these session summaries include CP session IDs.

RELATED DOCUMENTATION

IPv6 Flow-Based Processing Overview | 132

Understanding Path MTU Messages for IPv6 Packets

This topics describes path maximum transmission unit (MTU) and explains how the flow module for SRX Series Firewalls processes and uses path MTU messages.

Every link has an MTU size that specifies the size of the largest packet the link can transmit. A larger MTU size means that fewer packets are required to transmit a certain amount of data. To achieve the best data transmission performance, IPv6 data packets sent from one node (the source) to another node (the destination) should be the largest possible size that can traverse the path between the nodes. (Larger and fewer packets constrain the cost of packet header processing and routing processes that can affect transmission performance.)

However, for a packet to successfully traverse the path from the source node to the destination node, the MTU size of the source node interface must be no larger than that of the smallest MTU size of all nodes on the path between the source and destination. This value is referred to as the path maximum transmission unit (path MTU). If a packet is larger than a link's MTU size, it is likely that the link will drop it. For IPv6, an intermediate node cannot fragment a packet.

IPv6 defines a standard mechanism called path MTU discovery that a source node can use to learn the path MTU of a path that a packet is likely to traverse. If any of the packets sent on that path are too large to be forwarded by a node along the path, that node discards the packet and returns an ICMPv6 Packet Too Big message. The source node can then adjust the MTU size to be smaller than that of the node that dropped it and sent the ICMPv6 message, and then retransmit the packet. A source node might receive Packet Too Big messages repeatedly until its packet traverses all nodes along the path successfully.

NOTE: On all SRX Series Firewalls, the Routing Engine cannot detect the path MTU of an IPv6 multicast address (with a large size packet).

After the path MTU size is determined and the appropriate MTU size is set, an outgoing packet might be routed along a different path with a node whose link MTU size is smaller than the path MTU size determined previously. In this case, the flow module engages the path MTU discovery process again.

When the flow module receives an ICMP Packet Too Big message with a destination address that belongs to it, it:

- If the packet is a transit one, the flow module searches for a session that matches the packet's embedded 5-tuple data. It is finds a matching session, it delivers the packet to it. If there is no matching session, it drops the packet.

When the flow module receives a packet, before it transmits it to the egress interface, it checks to determine if the MTU size of the egress interface is greater than the packet length.

- If the MTU size is greater than the packet length, it continues to process the packet.
- If the MTU size is less than the packet length, it drops the packet and sends an ICMPv6 Packet Too Big message to the source node.

NOTE: When *chassis cluster* is configured and the path MTU updates the MTU of the tunnel interface, the flow module does not synchronize the new MTU to peer nodes. The MTU size might be updated again by a larger packet on a peer node, which has no impact on packet transmission.

IPv6 Flow-Based Processing Overview

IN THIS SECTION

Learn about how SRX Series Firewalls process IPv6 packets, IPv6 extension headers, and ICMPv6 packets.

The IPv6 Packet Header and SRX Series Overview

Every IPv6 packet has a minimum 40 bytes (320 bits) long basic packet header. The IPv6 packet header optionally have extension headers, which contains the supplementary information about the network devices.

For IPv6 packets, flow processing parses the extension headers and transport layer headers in the following way:

- If the software encounters a TCP, a UDP, an ESP, an AH, or an ICMPv6 header, it parses the header and assumes that the packet payload corresponds to the specified protocol type.
- If the software encounters a hop-by-hop header, a routing and destination header, or a fragment header, it continues to parse the next extension header.
- If it encounters the no-next-header extension header, the software detects that the packet is that of an unknown protocol (protocol equals 0).
- For other extension headers, the software parses the header and identifies the packet as belonging to the protocol indicated by the extension header.

Understanding IPv6 Packet Header Extensions

IPv6 extension headers contains supplementary information used by network devices (such as routers, switches, and endpoint hosts) to decide how to direct or process an IPv6 packet. The length of each extension header is an integer multiple of 8 octets. This allows subsequent extension headers to use 8-octet structures.

Any header followed by an extension header contains a Next Header value that identifies the extension header type. Extension headers may be placed between the IPv6 header and the upper-layer header in a packet. Figure 15 on page 133 shows an IPv6 packet with the hop-by-hop options header. Similarly, an IPv6 header can carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header. Extension headers always follow the basic IPv6 header in the order as shown in Table 7 on page 134:

Figure 15: IPv6 Extension Header

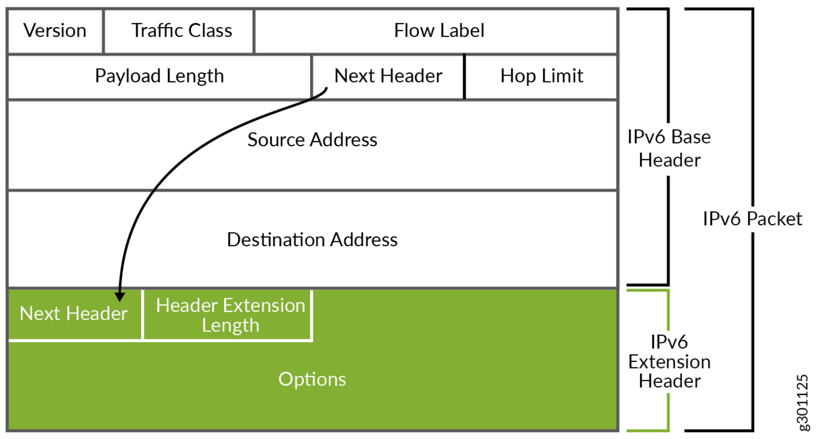


Table 7: IPv6 Extension Headers

Header Name	Purpose	Next Header Value
Hop-by-Hop Options	Specifies delivery parameters at each hop on the path to the destination host. A hop-by-hop option can appear only following the IPv6 basic header. If it is used, it should be the first extension header. It cannot appear after another extension header.	0
Destination Options	Specifies packet delivery parameters for either intermediate destination devices or the final destination host. When a packet uses this header.	60
Routing	Defines strict source routing and loose source routing for the packet. (With strict source routing, each intermediate destination device must be a single hop away. With loose source routing, intermediate destination devices can be one or more hops away)	43
Fragment	Specifies how to perform IPv6 fragmentation and reassembly services. A source node uses the fragment extension header to tell the destination node the size of the packet that was fragmented so that the destination node can reassemble the packet.	44
Authenticatio n	Provides authentication, data integrity, and anti-replay protection.	51
Encapsulating Security Payload	Provides data confidentiality, data authentication, and anti-replay protection for Encapsulated Security Payload (ESP) packets.	50

Table 7: IPv6 Extension Headers *(Continued)*

Header Name	Purpose	Next Header Value
Destination IP Address	Identifies the host device, or interface on a node, to which the IPv6 packet is to be sent. The destination address may appear twice, the first instance after the hop limit following the source IP address and the second instance after the final extension header.	60

For information on IPv6, refer to RFC2460.

SEE ALSO

IPv6 Overview

Understanding How SRX Series Firewalls Handle ICMPv6 Packets

This topic explains Internet Control Message Protocol (ICMP), ICMP messages, and how Junos OS for SRX Series Firewalls uses them.

ICMP provides a framework for reporting packet processing errors, for diagnostic purposes, and for implementation-specific functions. ICMP error messages make it possible for one node to inform another node that something has gone wrong during the course of data transfer. When IP version 6 (IPv6) was defined, the differences between IP version 4 (IPv4) and it were significant enough to require a new version of ICMP.

Every ICMPv6 message is preceded by an IPv6 header and zero or more IPv6 extension headers. The ICMPv6 header is identified by a Next Header value of 58 in the immediately preceding header. This is different from the value used to identify ICMP for IPv4. All ICMPv6 error messages have 32 bits of type-specific data to help the packet recipient locate the embedded invoking packet.

Most ICMPv6 packets have the same characteristics and behavior as normal IPv6 packets, and the Junos OS flow module processes them through first path and fast-path processing in the same way that it does normal IPv6 packets. Table 8 on page 136 shows the ICMPv6 embedded packet types that the flow module handles differently from normal ICMPv6 packets.

For these packets, the flow module uses a tuple that it creates from the embedded ICMPv6 packet to search for a matching session. It continues to process the packet without modifying the maximum transmission unit (MTU) until it finds a matching session, unless it receives an ICMPv6 Packet Too Big message for the interface. In this case, it modifies the MTU size for that interface. If the flow module does not find a matching session or if it cannot obtain a valid IPv6 header from the embedded payload, it drops the packet.

NOTE: A Packet Too Big message is the only kind of ICMPv6 packet that will cause the flow module to modify an interface.

Table 8: ICMPv6 Packets That Junos OS Handles Differently from Other ICMPv6 Packets

Message	Meaning
01-Destination Unreachable	When a packet cannot be delivered because of a problem with the way it is being sent, it is useful to have a feedback mechanism that can tell the source about the problem, including the reason why delivery of the packet failed. For IPv6, the Destination Unreachable message serves this purpose. Each message includes a code that indicates the nature of the problem that caused the packet delivery to fail. It also includes all or part of the packet that could not be delivered, to help the source device resolve the problem. When the flow module encounters a Destination Unreachable ICMP packet whose embedded packet header data matches the 5-tuple data for a session, the software terminates the session.

Table 8: ICMPv6 Packets That Junos OS Handles Differently from Other ICMPv6 Packets *(Continued)*

Message	Meaning
02-Packet Too Big	When the flow module receives an ICMPv6 Packet Too Big message intended for it, the flow module sends the packet to the ICMP protocol stack on the Routing Engine to engage the path maximum transmission unit (path MTU) discovery process. If the Packet Too Big message does not pertain to the device but rather is a transit packet, the device attempts to match the embedded 5-tuple data with a session. • If a matching session exists, the device delivers it to the source node. • If a matching session does not exist, the device drops the packet **NOTE**: A Packet Too Big message is the only kind of ICMPv6 packet that will cause the flow module to modify an interface.
03-Time Exceeded	When the flow module receives a packet that cannot be delivered because it has exceeded the hop count specified in the basic header hop-by-hop field, it sends this message to inform the packet’s source node that the packet was discarded for this reason.
04-Parameter Problem	When the device finds a problem with a field in the IPv6 header or extension headers that makes it impossible for it to process the packet, the software discards it and sends this ICMPv6 message to the packet’s source node, indicating the type and location of the problem.

4

CHAPTER

Monitoring Flow-Based Sessions and Establishing Parameters for Error Handling

IN THIS CHAPTER

Monitoring Security Flow Sessions

IN THIS SECTION

This topic covers information for monitoring, displaying and verifying of flow sessions using operational mode commands. Thus, you can debug without having to commit or modify your running configuration.

Monitoring Security Flow Sessions Overview

Junos OS allows you to configure and start the monitoring of flow sessions using operational mode commands. Thus, you can debug without having to commit or modify your running configuration. This approach can be especially useful when you do not want to change the state of your device by committing the configuration to turn on trace options.

To configure flow session monitoring, you must define flow filters, specify the output file, and start monitoring. Flow session monitoring does not start unless a filter (at least one) and an output file are specified. Also, defining the filters themselves does not trigger monitoring. You have to explicitly use the `monitor security flow start` and `monitor security flow stop` commands to enable and disable monitoring, respectively.

- Define flow filters—Define the flow sessions that you want to monitor using combinations of match criteria, such as source address, destination address, source port, destination port, IP protocol number, name of the incoming or outgoing interface, and the logical system name. You can delete filters using the `clear monitor security flow filter` command.

NOTE: Unlike filters defined in the configuration mode, filters defined using operational mode commands are cleared when you reboot your system.

- Specify the output file—Create an output file in which the security flow monitoring information is to be saved. This file is saved in the `/var/log/` directory. You can view the contents of this file by using the `show log` *`filename`* command. Use the `monitor security flow file` command to specify output file characteristics, such as its maximum size, maximum number, and type.

- Start monitoring—Use the `monitor security flow start` command to start monitoring. Once monitoring starts, any traffic that matches the filters is saved in the specified output file in the `/var/log/` directory. The basic-datapath flag is the default flag and turns on as monitoring starts.

  Use the `monitor security flow stop` command to stop monitoring. Once monitoring stops, the basic-datapath flag is cleared.

- Display monitoring flow information—Use the `show monitoring security flow` command to display details about the monitoring operation.

NOTE: You can configure flow session monitoring and debugging by using the monitoring operational mode commands and flow traceoptions configuration statements. These two operations cannot run in parallel. When you turn on security flow monitoring, the flow traceoption session is blocked and when the flow traceoption session is running, monitoring of the flow session is blocked.

Understanding How to Obtain Session Information for SRX Series Firewalls

You can obtain information about the sessions and packet flows active on your device, including detailed information about specific sessions. (The SRX Series Firewall also displays information about failed sessions.) You can display this information to observe activity and for debugging purposes. For example, you can use the show security flow session command:

- To display a list of incoming and outgoing IP flows, including services

- To show the security attributes associated with a flow, for example, the policies that apply to traffic belonging to that flow

- To display the session timeout value, when the session became active, for how long it has been active, and if there is active traffic on the session

NOTE: If an interface NAT is configured and sessions are set up with the NAT using that interface IP address, whenever the interface IP address changes, the sessions set up with NAT get refreshed and new sessions will be setup with new IP address. This you can verify using `show security flow session` CLI command.

Session information can also be logged if a related policy configuration includes the logging option. For the flow session log on all SRX Series Firewalls, policy configuration has been enhanced. Information on the packet incoming interface parameter in the session log for session-init and session-close and when a session is denied by a policy or by the application firewall is provided to meet Common Criteria (CC) Medium Robustness Protection Profiles (MRPP) compliance:

Policy configuration—To configure the policy for the session for which you want to log matches as log **session-init** or **session-close** and to record sessions in syslog:

- `set security policies from-zone untrustZone to-zone trust zone policy policy13 match source-address extHost1`
- `set security policies from-zone untrustZone to-zone trustZone policy policy13 match application junos-ping`
- `set security policies from-zone untrustZone to-zone trustZone policy policy13 then permit`
- `set security policies from-zone untrustZone to-zone trustZone policy policy13 then log session-init`
- `set security policies from-zone untrustZone to-zone trustZone policy policy13 then log session-close`

Example : Flow match policy13 will record the following information in the log:

<14>1 2010-09-30T14:55:04.323+08:00 mrpp-srx550-dut01 RT_FLOW - RT_FLOW_SESSION_CREATE [junos@2626.192.0.2.1.40 source-address="192.0.2.1" source-port="1" destination-address="198.51.100.12" destination-port="46384" service-name="icmp" nat-source-address="192.0.2.1" nat-source-port="1" nat-destination-address="198.51.100.12" nat-destination-port="46384" src-nat-rule-name="None" dst-nat-rule-name="None" protocol-id="1" policy-name="policy1" source-zone-name="trustZone" destination-zone-name="untrustZone" session-id-32="41" packet-incoming-interface="ge-0/0/1.0"] session created 192.0.2.1/1-->198.51.100.12/46384 icmp 192.0.2.1/1-->198.51.100.12/46384 None None 1 policy1 trustZone untrustZone 41 ge-0/0/1.0

<14>1 2010-09-30T14:55:07.188+08:00 mrpp-srx550-dut01 RT_FLOW - RT_FLOW_SESSION_CLOSE [junos@2626.192.0.2.1.40 reason="response received" source-address="192.0.2.1" source-port="1" destination-address="198.51.100.12" destination-port="46384" service-name="icmp" nat-source-address="192.0.2.1" nat-source-port="1" nat-destination-address=”198.51.100.12" nat-destination-port="46384" src-nat-rule-name="None" dst-nat-rule-name="None" protocol-id="1" policy-name="policy1" source-zone-name="trustZone" destination-zone-name="untrustZone" session-id-32="41" packets-from-client="1" bytes-from-client="84" packets-from-server="1" bytes-from-server="84" elapsed-time="0" packet-incoming-interface="ge-0/0/1.0"] session closed response

received: 192.0.2.1/1-->198.51.100.12/46384 icmp 192.0.2.1/1-->198.51.100.12/46384 None None 1 policy1 trustZone untrustZone 41 1(84) 1(84) 0 ge-0/0/1.0

Displaying Global Session Parameters for All SRX Series Services Gateways

IN THIS SECTION

Purpose

Obtain information about configured parameters that apply to all flows or sessions.

Action

To view session information in the CLI, enter the following command:

```
user@host# show security flow
```

Meaning

The `show security flow` configuration command displays the following information:

- `allow-dns-reply`—Identifies if unmatched incoming Domain Name System (DNS) reply packets are allowed.
- `route-change-timeout`—If enabled, displays the session timeout value to be used on a route change to a nonexistent route.
- `tcp-mss`—Shows the current configuration for the TCP maximum segment size value to be used for all TCP packets for network traffic.
- `tcp-session`—Displays all configured parameters that control session parameters.

- `syn-flood-protection-mode`—Displays the SYN Proxy mode.

Displaying a Summary of Sessions for SRX Series Services Gateways

IN THIS SECTION

Purpose

Determine the kinds of sessions on your device, how many of each kind there are—for example, the number of unicast sessions and multicast sessions—the number of failed sessions, the number of sessions that are currently used and the maximum number of sessions that the device supports. This command also displays the details of the sessions that are currently used. For example, valid sessions, pending sessions, invalidated sessions and sessions in other states.

Action

To view session summary information in the CLI, enter the following CLI command:

```
user@host> show security flow session summary
```

Displaying Session and Flow Information About Sessions for SRX Series Services Gateways

IN THIS SECTION

Purpose

Display information about all sessions on your device, including the session ID, the virtual system the session belongs to, the Network Address Translation (NAT) source pool (if source NAT is used), the configured timeout value for the session and its standard timeout, and the session start time and how long the session has been active. The display also shows all standard flow information, including the direction of the flow, the source address and port, the destination address and port, the IP protocol, and the interface used for the session.

Action

To view session flow information in the CLI, enter the following command:

```
user@host> show security flow session
```

Displaying Session and Flow Information About a Specific Session for SRX Series Services Gateways

IN THIS SECTION

Purpose

When you know the session identifier, you can display all session and flow information for a specific session rather than for all sessions.

Action

To view information about a specific session in the CLI, enter the following command:

```
user@host> show security flow session session-identifier 40000381
```

Using Filters to Display Session and Flow Information for SRX Series Services Gateways

IN THIS SECTION

Purpose

You can display flow and session information about one or more sessions by specifying a filter as an argument to the `show security flow session` command. You can use the following filters: application, destination-port, destination-prefix, family, idp, interface, nat, protocol, resource-manager, session-identifier, source-port, source-prefix and tunnel. The device displays the information for each session followed by a line specifying the number of sessions reported on. Here is an example of the command using the source-prefix filter.

Action

To view information about selected sessions using filters in the CLI, enter the following command:

```
user@host> show security flow session source-prefix 10/8
```

Information Provided in Session Log Entries for SRX Series Services Gateways

Session log entries are tied to policy configuration. Each main session event—create, close, and deny—will create a log entry if the controlling policy has enabled logging.

Different fields are logged for session create, session close, and session deny events as shown in Table 9 on page 146, Table 10 on page 147, and Table 11 on page 151. The same field name under each type indicates that the same information is logged, but each table is a full list of all data recorded for that type of session log.

The following table defines the fields displayed in session log entries.

Table 9: Session Create Log Fields

Field	Description
`source-address`	Source IP address of the packet that created the session.
`source-port`	Source port of the packet that created the session.
`destination-address`	Destination IP address of the packet that created the session.
`destination-port`	Destination port of the packet that created the session.
`service-name`	Application that the packet traversed (for example, “junos-telnet” for Telnet traffic during the session allowed by a policy that permits native Telnet).
`nat-source-address`	The translated NAT source address if NAT was applied; otherwise, the source address as above.
`nat-source-port`	The translated NAT source port if NAT was applied; otherwise, the source port as above.
`nat-destination-address`	The translated NAT destination address if NAT was applied; otherwise, the destination address as above.
`nat-destination-port`	The translated NAT destination port if NAT was applied; otherwise, the destination port as above.
`src-nat-rule-name`	The source NAT rule that was applied to the session (if any). If static NAT is also configured and applied to the session and if source address translation takes place, then this field shows the static NAT rule name.*
`dst-nat-rule-name`	The destination NAT rule that was applied to the session (if any). If static NAT is also configured and applied to the session and if destination address translation takes place, then this field shows the static NAT rule name.*
`protocol-id`	The protocol ID of the packet that created the session.

Table 9: Session Create Log Fields *(Continued)*

Field	Description
`policy-name`	The name of the policy that permitted the session creation.
`session-id-32`	The 32-bit session ID.
** Note that some sessions might have both destination and source NAT applied and the information logged.*	

Starting with Junos OS Release 12.1X47-D20 and Junos OS Release 17.3R1, the system log includes information about NAT rule type. Two new src-nat-rule-type and dst-nat-rule-type fileds are introduced in the NAT rule session.

Table 10: Session Close Log Fields

Field	Description
`reason`	The reason the session was closed.
`source-address`	Source IP address of the packet that created the session.
`source-port`	Source port of the packet that created the session.
`destination-address`	Destination IP address of the packet that created the session.
`destination-port`	Destination port of the packet that created the session.
`service-name`	Application that the packet traversed (for example, “junos-telnet” for Telnet traffic during the session allowed by a policy that permits native Telnet).
`nat-source-address`	The translated NAT source address if NAT was applied; otherwise, the source address as above.

Table 10: Session Close Log Fields *(Continued)*

Field	Description
`nat-source-port`	The translated NAT source port if NAT was applied; otherwise, the source port as above.
`nat-destination-address`	The translated NAT destination address if NAT was applied; otherwise, the destination address as above.
`nat-destination-port`	The translated NAT destination port if NAT was applied; otherwise, the destination port as above.
`src-nat-rule-name`	The source NAT rule that was applied to the session (if any). If static NAT is also configured and applied to the session and if source address translation takes place, then this field shows the static NAT rule name.*
`dst-nat-rule-name`	The destination NAT rule that was applied to the session (if any). If static NAT is also configured and applied to the session and if destination address translation takes place, then this field shows the static NAT rule name.*
`protocol-id`	The protocol ID of the packet that created the session.
`policy-name`	The name of the policy that permitted the session creation.
`session-id-32`	The 32-bit session ID.
`packets-from-client`	The number of packets sent by the client related to this session.
`bytes-from-client`	The number of data bytes sent by the client related to this session.

Table 10: Session Close Log Fields *(Continued)*

Field	Description
`packets-from-server`	The number of packets sent by the server related to this session.
`bytes-from-server`	The number of data bytes sent by the server related to this session.
`elapsed-time`	The total session elapsed time from permit to close, given in seconds.
`unset`	During the session creation, you can set the session close reason as `unset`. The session closes with the reason `unset` if the session installation on the control point is not successful. The reason for session installation varies, for example, nonavailability of memory for nonmanagement session installation.
`TCP CLIENT RST`	The session was closed by a TCP reset packet sent to it from the client.
`TCP SERVER RST`	The session was closed by a TCP reset packet sent to it from the server.
`TCP FIN`	FIN received from either end.
`response received`	Response received for a packet request (for example, ICMP req-reply).
`ICMP error`	ICMP error received.
`aged out`	Session aged out was reached.

Table 10: Session Close Log Fields *(Continued)*

Field	Description
`ALG`	ALG errors closed the session (for example, remote access server (RAS) maximum limit reached).
`HA`	HA message closed the session.
`idle Timeout`	There was no traffic for the session before the configured age-out time was reached.
`auth`	Authentication failed.
`IDP`	IDP closed the session because of security module (SM) internal error.
`synproxy failure`	SYN proxy failure closed the session.
`synproxy limit`	Reason for failure in allocating minor session, need to free original session.
`parent closed`	Parent session closed.
`CLI`	Session cleared by a CLI .
`CP NACK`	CP NACK response received.
`CP delete`	CP ACK deletion closed the session.
`policy delete`	Corresponding policy marked for deletion.
`fwd session`	Session closed because of forwarding session deletion.
`multicast route change`	Session closed because multicast route changed.

Table 10: Session Close Log Fields *(Continued)*

Field	Description
`first path reroute, session recreated`	The first path is rerouted and session is re-created.
`source NAT allocation failure`	SPU received ACK message from the central point but failed to receive the DIP resource. Therefore this packet is dropped and the session is closed.
`other`	Session closed because of all other reasons (for example, the pim reg tun needed refreshing).
`error create IKE pass-through template`	IKE pass-through template creation errors.
`IKE pass-through child session ageout`	Session is deleted because the IKE pass through template session has no child.
`sess timeout on pending state`	Pending session closed because time out timer reached the pending state.
`unknown`	Session closed because of unknown reasons.

** Note that some sessions might have both destination and source NAT applied and the information logged.*

Table 11: Session Deny Log Fields

Field	Description
`source-address`	Source IP address of the packet that attempted to create the session.
`source-port`	Source port of the packet that attempted to create the session.
`destination-address`	Destination IP address of the packet that attempted to create the session.
`destination-port`	Destination port of the packet that attempted to create the session.

Table 11: Session Deny Log Fields *(Continued)*

Field	Description
`service-name`	Application that the packet attempted to traverse.
`protocol-id`	The protocol ID of the packet that attempted to create the session.
`icmp-type`	The ICMP type if the denied packet was ICMP configured; otherwise, this field will be 0.
`policy-name`	The name of the policy that denied the session creation.

Error Handling Extensions

IN THIS SECTION

Understanding Chassis Manager FPC Fault Detection and Error Handling Enhancements

IN THIS SECTION

The Junos OS Routing Engine and microkernel error detection and management feature on the SRX5400, SRX5600, and SRX5800 devices enables the Routing Engine and the ukernel to accumulate and store the history of all reported error activity and counters for various severity levels. You can configure how errors are handled and specify the severity levels and the actions to perform when an error is detected and a threshold is reached. You can generate and display reports for encountered errors based on stored information.

Starting with Junos OS Release 15.1X49-D30 and Junos OS Release 17.3R1, error detection enhancements are provided that detect additional errors on IOCs and SPCs and provide enhanced error management. This implementation extends the error detection and management covered in the `show chassis fpc error` topic.

NOTE: This feature is not supported on Routing Engine version 1.

Error Handling on IOCs and SPCs

Starting with Junos OS Release 15.1-X49-D50 and Junos OS Release 17.3R1, the error management enhancements are supported on IOC2 and IOC3 I/O cards (IOCs) and SPC2 Services Processing Cards (SPCs). Some enhancement functions are particular to either the IOC2 and the IOC3 or the SPC2 FPCs, and the differences are called out in this topic.

Error Detection and Management

Error management entails:

- Detecting an error.

  Junos OS monitors the chassis component state to detect a set of error conditions. A detected error can belong to one of the preconfigured error severity levels:

  - Fatal
  - Major
  - Minor

- Identifying the action to take.

  When an error occurs, the system identifies the action to take based on the severity level of the error and the thresholds set and met.

  An FPC maintains a set of error counters for each error severity level. An error counter set consists of a counter that is cumulative across all errors and counters for individual errors and types. It is this information that is stored in the Routing Engine. Each occurrence counter is associated with an error occurrence threshold. There are two threshold levels: one based on the type and the other on severity.

- Executing the action.

  For these enhancements, the preconfigured actions that you can direct the device to take when the Routing Engine's error occurrence count for a given security level reaches the configured threshold are:

  - Reset
  - Offline
  - Alarm
  - Get-state
  - Log

CAUTION: Take care when setting the fault handling actions for SPC2 cards on the SRX5000 line of devices. Consider that if you set the fault handling action on an SPC2 card to offline or reset, when the card is either taken offline or the reboot occurs, the chassis daemon (chassisd) will reboot all of its FPC cards, both SPCs and IOCs—that is, the entire chassis will be rebooted.

Error Detection Processes

With these enhancements, the following error detection processes are enabled and supported:

- Error management on the Routing Engine version 2.
- Error management on ukernel modules on SPC2 cards.
- Error management on the IOC2 and IOC3 cards.
- Driver checks for datapath error detection of wedge conditions.

  NOTE: Wedge condition detection for the Trinity Offload Engine driver is supported only on SPC2 cards. That is, it is not supported on the IOC2 and IOC3 cards.

- Wedge detection for host loopback.

NOTE: Wedge condition detection for host loopback is supported only on SPC2 cards. That is, it is not supported on the IOC2 and IOC3 cards.

- Chassis Manager fabric error detection.
- Control path error detections on IOC2 and IOC3 cards.

Integration with Chassis Cluster

In a chassis cluster environment, when an alarm is raised for the first time because of a major or a fatal error, a Redundancy Group 1 (RG1) switchover is triggered. This is the standard behavior on SRX Series Firewalls, and it remains unchanged. However, with these enhancements, the alarm is added to the default fault handling action list for a fatal error. Adding an alarm to the default fault handling list allows the chassis alarm to trigger the RG1 switchover as soon as the fatal error is detected.

Wedge Detection, Reporting, and Management

A wedge condition is caused by an error that blocks network traffic.

This feature detects several types of wedge conditions. It:

- Determines if the wedge is transient or irreversible.
- Records the wedge conditions in statistics and syslogs.
- Alerts network administrators to irreversible wedges by raising a chassis alarm on the Routing Engine.
- Verifies that the following datapath error detections are enabled for the IOC2, IOC3, and SPC2 cards:
  - Wedge detection for XM driver
  - Wedge detection for LU driver
  - Wedge detection for XL driver
  - Wedge detection for TOE driver (SPC2 only)
  - Wedge detection for host loopback (SPC2 only)

All datapath wedge conditions are detected and reported within 5 seconds. Each error detecting module records and reports the state and history of its identifiable wedge conditions.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D50	Starting with Junos OS Release 15.1-X49-D50 and Junos OS Release 17.3R1, the error management enhancements are supported on IOC2 and IOC3 I/O cards (IOCs) and SPC2 Services Processing Cards (SPCs).
15.1X49-D30	Starting with Junos OS Release 15.1X49-D30 and Junos OS Release 17.3R1, error detection enhancements are provided that detect additional errors on IOCs and SPCs and provide enhanced error management.
12.1X47-D20	Starting with Junos OS Release 12.1X47-D20 and Junos OS Release 17.3R1, the system log includes information about NAT rule type.

RELATED DOCUMENTATION

Monitoring X2 Traffic | 156

Monitoring X2 Traffic

IN THIS SECTION

This topic covers X2 traffic monitoring on SRX Series Firewalls.

Understanding X2 Traffic Monitoring

IN THIS SECTION

In an LTE mobile network, SRX Series Firewalls act as secure gateways connecting Evolved Node Bs (eNodeBs) for signal handover, monitoring, and radio coverage. SRX Series Firewalls use IPsec tunnels to connect eNodeBs. The user plane and control plane traffic that flows from one eNodeB to the other eNodeB is called the X2 traffic.

X2 Traffic Monitoring Overview

The X2 traffic passing through IPsec tunnels is encrypted. Because of this, mobile network operators need a way to monitor X2 traffic so that they can debug handover issues across eNodeBs. The Junos OS implementation allows monitoring of the X2 traffic by snooping into the cleartext X2 traffic as it flows through the SRX Series Firewall coming out of one IPsec tunnel and going into the other IPsec tunnel—after traffic is decrypted and before it is encrypted again.

Figure 16 on page 158 shows the flow of X2 traffic within the SRX Series Firewall. As the traffic reaches the SRX Series Firewall on one st0.x interface, it gets decrypted. Then it is encrypted and forwarded to the destination eNodeB through its dedicated st0.y interface. Snooping is performed on the decrypted X2 traffic on the SRX Series Firewall.

Figure 16: SRX Series Firewall in an LTE Mobile Network

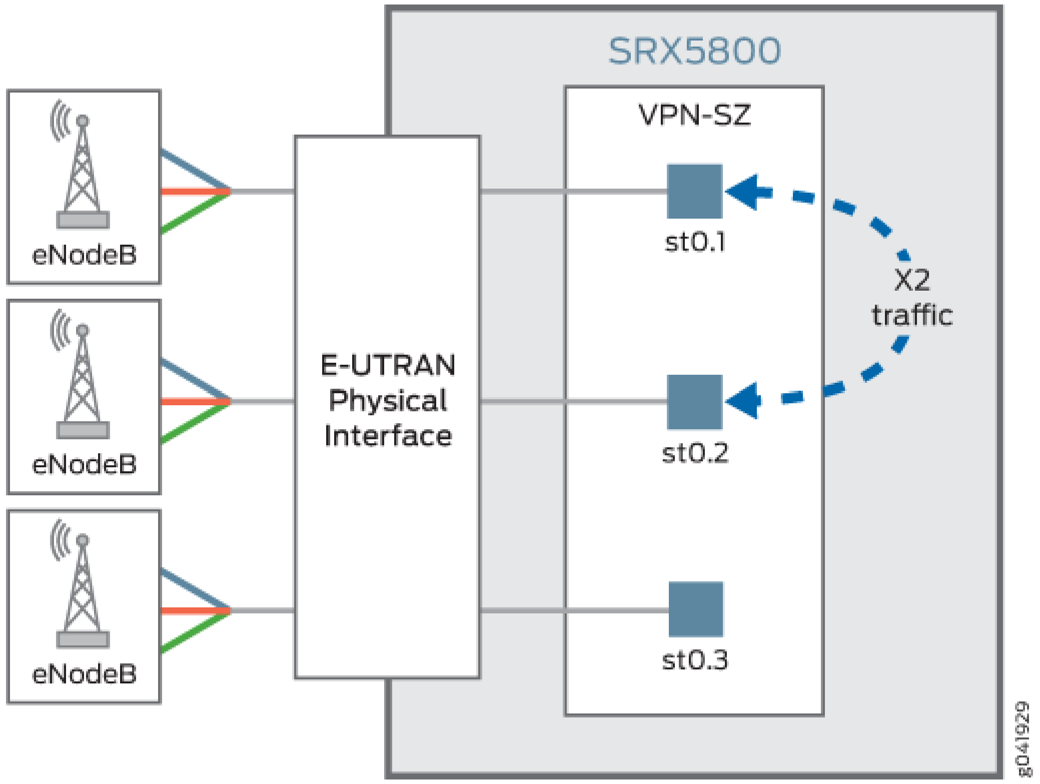
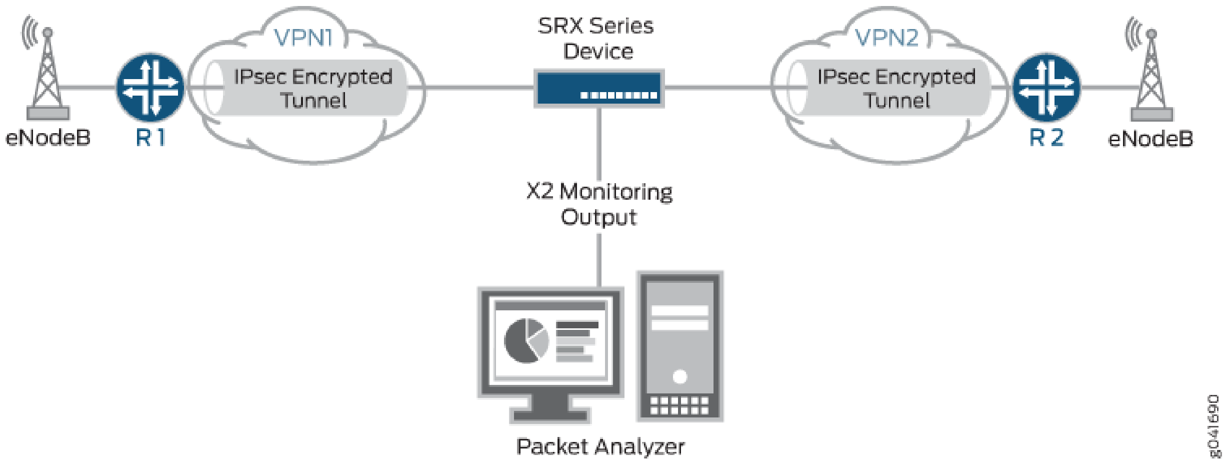


Figure 17 on page 159 shows a mobile operators network with an SRX Series Firewall providing IPsec tunnel connection between the two eNodeBs. The SRX Series Firewall is connected to a packet analyzer (also called a *sniffing* device) that is used for collecting and monitoring the X2 traffic. The IPsec tunnel from each eNodeB terminates on a dedicated secure tunnel interface on the SRX Series Firewall. Inbound traffic coming out of the IPsec tunnel is decrypted while outbound traffic leaving the device is encrypted.

Figure 17: Monitoring X2 Traffic



To monitor the X2 traffic, you can configure up to 15 different mirror filters that specify unique sets of parameters against which traffic is matched. The filtered packets are duplicated and sent to a physical interface. To allow the packet analyzer to capture the filtered packets, you specify the output interface on the SRX Series Firewall and the MAC address of the packet analyzer. Because the output interface is connected to the same Layer 2 network as the packet analyzer, once mirror filtering is turned on, the packet analyzer can collect and analyze the X2 traffic.

The SRX Series mirror filter feature is bidirectional, much like a session. X2 traffic flowing through an IPSec VPN that matches a mirror filter is mirrored and analyzed; traffic returning from those devices is also mirrored and analyzed.

Starting in Junos OS Release 18.4R1, if the output X2 interface of a mirror filter is configured for an st0 interface to filter traffic that you want to analyze, the packet is duplicated and encrypted by the IPsec tunnel bound to the st0 interface. This enhancement supports the SRX Series Firewalls to send traffic mirrored from a port on an IPsec tunnel. Mirrored traffic includes unmodified Layer 3 headers.

NOTE: Although there is no minimum required number of parameters for a mirror filter, please be mindful that if you specify too few criteria or accidentally commit an incomplete filter, an over-proportional amount of traffic flow through the system could be mirrored.

Limitations of X2 Traffic Monitoring

- For X2 traffic in a chassis cluster setup, mirrored packets cannot traverse through the data link (fabric interface).

- Support for X2 traffic mirroring is not available with PowerMode IPsec (PMI) enabled. If PMI is enabled in one direction but disabled in the other, you can capture X2 traffic mirroring only for the direction where PMI is not active.

X2 Traffic Terminology

Table 12 on page 160 lists some X2 traffic related terms and their descriptions.

Table 12: X2 Traffic Terminology

Term	Description
Evolved packet core (EPC)	Main component of System Architecture Evolution (SAE) and is also known as the SAE core. The EPC supports the IP network and serves as the equivalent of a General Packet Radio Service (GPRS) network, using the mobility management entity (MME), Serving Gateway (SGW), and Packet Data Network Gateway (PGW) subcomponents.
Evolved Universal Terrestrial Radio Access Network (E-UTRAN)	A radio access network standard. E-UTRAN is a new air interface system. It provides higher data rates and lower latency and is optimized for packet data. It uses Orthogonal Frequency-Division Multiple Access (OFDMA) for the downlink and Single-carrier Frequency Division Multiple Access for the uplink.
Evolved Node B (eNodeB)	A device connected to the mobile phone network that communicates directly with mobile handsets, like a base transceiver station in Global System for Mobile Communications (GSM) networks. An eNodeB is controlled by a radio network controller (RNC).
Long Term Evolution (LTE)	A standard for wireless communication of high-speed data for mobile phones and data terminals. It increases the capacity and speed using a different radio interface and makes core network improvements.
X2 interface	A point-to-point logical interface between two eNodeBs with the E-UTRAN. It supports the exchange of signaling information between two eNodeBs and supports the forwarding of protocol data units (PDUs) to the respective tunnel endpoints.

Table 12: X2 Traffic Terminology *(Continued)*

Term	Description
X2 Application Protocol (X2AP)	Protocol used by the X2 interface. It is used for handling the user equipment mobility within the E-UTRAN and provides the following functions: • Manages mobility and load • Reports general error situations • Sets and resets the X2 interface • Updates the eNodeB configuration

Example: Configuring a Mirror Filter for X2 Traffic Monitoring

IN THIS SECTION

This example shows how to configure a mirror filter to monitor X2 traffic between two eNodeBs in an LTE mobile network.

Requirements

Before you begin:

- Understand X2 traffic monitoring. .
- Configure the interfaces, security zones, security policies, and the route-based VPN tunnels to allow data to be securely transferred between the SRX Series Firewall and the two eNodeBs.

Overview

As a network operator, you need a way to monitor the X2 traffic to debug any handover issues across eNodeBs. The mirror filter feature allows you to do that. Traffic coming out of an IPsec tunnel is decrypted, mirrored and analyzed, and then encrypted again to go into the outbound IPsec tunnel.

More specifically, traffic that matches a mirror filter is mirrored and sent to an output interface that is connected to a packet analyzer (also called a *sniffing* device). The packet analyzer analyzes the X2 traffic, allowing you to monitor it. Then the traffic is encrypted again before it is sent to the outbound IPsec tunnel.

The SRX Series mirror filter feature is bidirectional, much like a session. X2 traffic flowing through an IPSec VPN that matches a mirror filter is mirrored and analyzed; traffic returning from those devices is also mirrored and analyzed.

To use the mirror filter feature to monitor X2 traffic, you configure mirror filters. You can configure up to 15 different mirror filters to be used concurrently to filter for various kinds of traffic. Each mirror filter contains a set of parameters and their values against which traffic is matched.

NOTE: Although there is no minimum required number of parameters for a mirror filter, please be mindful that if you specify too few criteria or accidentally commit an incomplete filter, an over-proportional amount of traffic flow through the system could be mirrored.

A mirror filter can contain some or all of the following parameters to filter traffic:

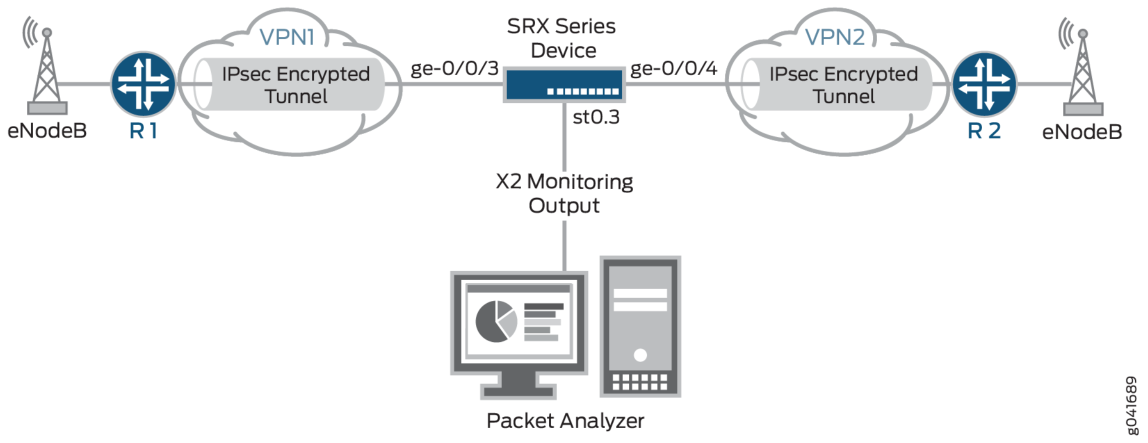
- destination IP address prefix
- destination port
- IP protocol
- source IP address prefix
- source port
- incoming and outgoing interfaces

You also specify the output interface and the MAC address of the packet analyzer as part of the configuration.

In this example, an SRX Series Firewall uses IPsec tunnels to connect two eNodeBs in an LTE mobile network. The example configures a mirror filter called traffic-https.

Figure 18 on page 163 shows the SRX Series Firewall connecting to the eNodeBs using IPsec tunnels. The SRX Series Firewall is also connected to a packet analyzer.

Figure 18: Configuring Mirror Filters for X2 Traffic Monitoring



In this example, all HTTPS traffic is analyzed whose destination is to devices with IP addresses that have the prefix 203.0.113.0/24 and for which the destination port 443 is used, the default port for HTTPS traffic. Packets that match the traffic-https filter are *mirrored* and sent through the output interface ge-0/0/5 to the packet analyzer with the MAC address 00:50:56:87:20:5E. Returning traffic from these devices is also monitored.

NOTE: The output interface for mirror filter is that of the packet analyzer, which is why the HTTP protocol is used.

The output interface for the packet analyzer uses the HTTP protocol.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security forwarding-options mirror-filter traffic-https
set security forwarding-options mirror-filter traffic-https destination-port 443
set security forwarding-options mirror-filter traffic-https destination-prefix 203.0.113.0/24
set security forwarding-options mirror-filter traffic-https protocol 6
set security forwarding-options mirror-filter traffic-http output interface ge-0/0/5
set security forwarding-options mirror-filter traffic-http output destination-mac
00:50:56:87:20:5E
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure a mirror filter for monitoring X2 traffic:

1. Create a mirror filter called traffic-https.

   ```
   [edit]
   user@host# edit security forwarding-options mirror-filter traffic-https
   ```

2. Specify the mirror filter parameters against which traffic is matched.

   ```
   [edit security forwarding-options mirror-filter traffic-https]
   user@host# set destination-port 443
   user@host# set destination-prefix 203.0.113.0/24
   user@host# set protocol 6
   ```

3. Specify the output interface for the mirrored packets to be sent to the packet analyzer.

```
[edit security forwarding-options mirror-filter traffic-https]
user@host# set output interface ge-0/0/5
```

4. Specify the MAC address of the packet analyzer as a destination for all mirrored packets, that is, those packets that match the mirror filters.

```
[edit security forwarding-options mirror-filter traffic-https]
user@host# set output destination-mac 00:50:56:87:20:5E
```

Results

From configuration mode, confirm your configuration by entering the `show security forwarding-options` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
user@host# show security forwarding-options
    mirror-filter traffic-https {
        protocol 6;
        destination-port 443;
        destination-prefix 203.0.113.0/24;
        output {
            interface ge-0/0/5;
            destination-mac 00:50:56:87:20:5E;
        }
    }
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying the Status of Mirror Filter

Purpose

Verify that mirror filter is active or not.

Action

From operational mode, enter the `show security forward-options mirror-filter` command for the specific mirror filter.

```
user@host> show security forward-options mirror-filter traffic-https
Security mirror status

    mirror-filter-name: traffic-https
    protocol: 6
         destination-port: 443
    destination-prefix 203.0.113.0/24
    filter-counters: 2
    output-counters: 2
```

Meaning

The output provides the mirror filter status. It shows that a mirror filter called traffic-https is active. The traffic-https mirror filter specifies the protocol, destination prefix, and destination port that traffic must match in order for it to be mirrored and analyzed.

This output shows that two packets were mirrored.

5

CHAPTER

Packet Based Forwarding

IN THIS CHAPTER

Packet-Based Forwarding

IN THIS SECTION

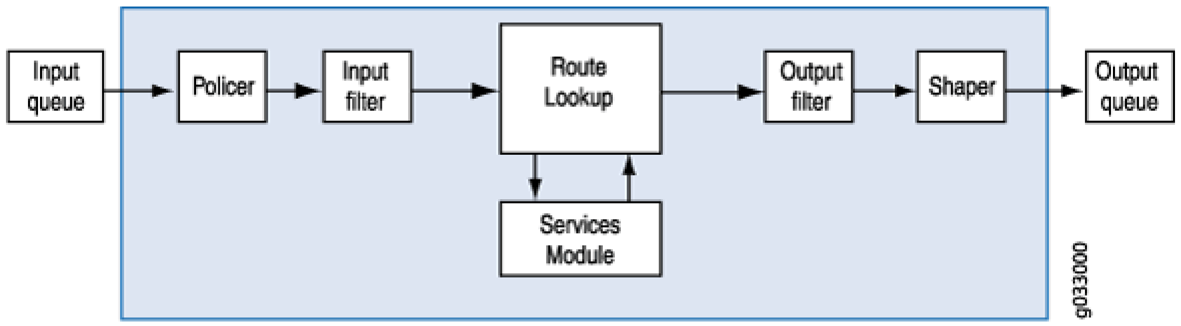
An SRX Series Firewall operate in two different modes: packet mode and flow mode. In flow mode, SRX processes all traffic by analyzing the state or session of traffic. This is also called stateful processing of traffic. In packet mode, SRX processes the traffic on a per-packet basis. This is also known as stateless processing of traffic.

Understanding Packet-Based Processing

Packets that enter and exit a Juniper Networks device running Junos OS can undergo packet-based processing. Packet-based, or stateless, packet processing treats packets discretely. Each packet is assessed individually for treatment. Stateless packet-based forwarding is performed on a packet-by-packet basis without regard to flow or state information. Each packet is assessed individually for treatment.

Figure 19 on page 169 shows the traffic flow for packet-based forwarding.

Figure 19: Traffic Flow for Packet-Based Forwarding



As packets enter the device, classifiers, filters and policers are applied to it. Next, the egress interface for the packet is determined through a route lookup. Once the egress interface for the packet is found, filters are applied and the packet is sent to the egress interface where it is queued and scheduled for transmission.

Packet-based forwarding does not require any information about either previous or subsequent packets that belong to a given connection, and any decision to allow or deny traffic is packet specific. This architecture has the benefit of massive scaling because it forwards packets without keeping track of individual flows or state.

Understanding Selective Stateless Packet-Based Services

Selective stateless packet-based services allow you to use both flow-based and packet-based forwarding simultaneously on a system. You can selectively direct traffic that requires packet-based, stateless forwarding to avoid stateful flow-based forwarding by using stateless firewall filters, also known as access control lists (ACLs). The traffic not so directed follows the default flow-based forwarding path. Bypassing flow-based forwarding can be useful for traffic for which you explicitly want to avoid flow session-scaling constraints.

By default, Juniper Networks Security devices running Junos OS use flow-based forwarding. Selective stateless packet-based services allows you to configure the device to provide only packet-based processing for selected traffic based on input filter terms. Other traffic is processed for flow-based forwarding. Bypassing flow-based forwarding is useful for deployments where you want to avoid session-scaling constraints and session creation and maintenance costs.

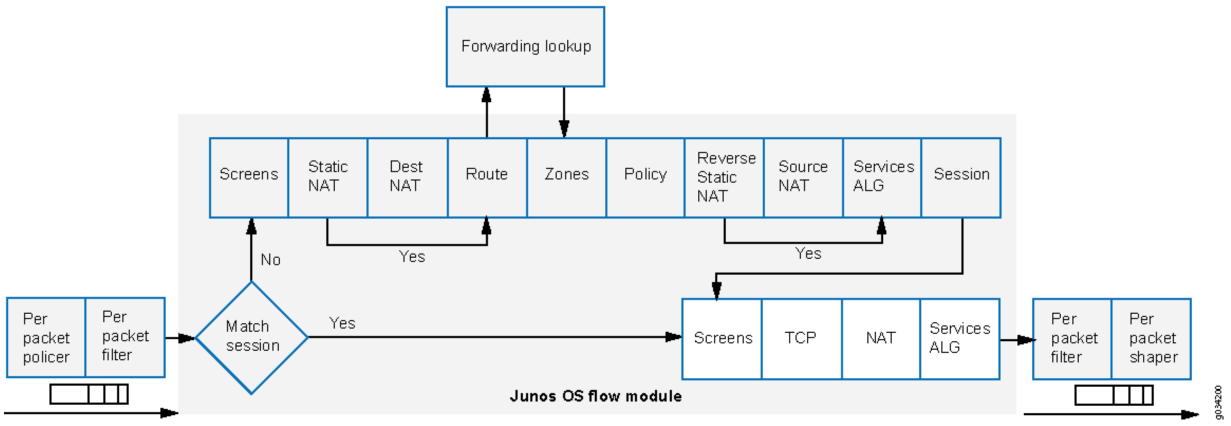
When you configure the device for selective stateless packet-based processing, packets entering the system are treated differently depending on certain conditions:

- If a packet satisfies matching conditions specified in input filter terms, it is marked for packet mode and all configured packet mode features are applied to it. No flow-based security features are applied. It bypasses them.

- If a packet has not been flagged for packet-mode, it undergoes normal processing. All services except for MPLS can be applied to this traffic.

Figure 20 on page 170 shows traffic flow with selective stateless packet-based services bypassing flow-based processing.

Figure 20: Traffic Flow with Selective Stateless Packet-Based Services



When the packet comes in on an interface, the input packet filters configured on the interface are applied.

- If the packet matches the conditions specified in the *firewall filter*, a `packet-mode` action modifier is set to the packet. The packet-mode action modifier updates a bit field in the packet key buffer—this bit field is used to determine if the flow-based forwarding needs to be bypassed. As a result, the packet with the packet-mode action modifier bypasses the flow-based forwarding completely. The egress interface for the packet is determined through a route lookup. Once the egress interface for the packet is found, filters are applied and the packet is sent to the egress interface where it is queued and scheduled for transmission.

- If the packet does not match the conditions specified in this filter term, it is evaluated against other terms configured in the filter. If, after all terms are evaluated, a packet matches no terms in a filter, the packet is silently discarded. To prevent packets from being discarded, you configure a term in the filter specifying an action to accept all packets.

A defined set of stateless services is available with selective stateless packet-based services:

- IPv4/IPv6 routing (unicast and multicast protocols)

- *Class of service* (CoS)

- Link fragmentation and interleaving (LFI)

- Generic routing encapsulation (GRE)

- Layer 2 switching
- Multiprotocol Label Switching (MPLS)
- Stateless firewall filters
- Compressed Real-Time Transport Protocol (CRTP)

Although traffic requiring MPLS services must be processed in packet mode, under some circumstances it might be necessary to concurrently apply certain services to this traffic that can only be provided in flow mode, such as stateful inspection, NAT, and IPsec. To direct the system to process traffic in both flow and packet modes, you must configure multiple routing instances connected through a tunnel interface. One routing instance must be configured to process the packets in flow mode and the other routing instance must be configured to process the packets in packet mode. When you use a tunnel interface to connect routing instances, traffic between those routing instances is injected again into the forwarding path and it can then be reprocessed using a different forwarding method.

Selective Stateless Packet-Based Services Configuration Overview

This feature is supported on SRX300, SRX320, SRX340, SRX345, and vSRX Virtual Firewall devices. You configure selective stateless packet-based services using the stateless firewall filters, also known as access control lists (ACLs). You classify traffic for packet-based forwarding by specifying match conditions in the firewall filters and configure a `packet-mode` action modifier to specify the action. Once match conditions and actions are defined, firewall filters are applied to relevant interfaces.

To configure a firewall filter:

1. Define the address family—First define the address family of the packets that a firewall filter matches. To define the family name, specify `inet` to filter IPv4/IPv6 packets. Specify `mpls` to filter MPLS packets. Specify `ccc` to filter Layer 2 switching cross-connects.
2. Define terms—Define one or more terms that specify the filtering criteria and the action to take if a match occurs. Each term consists of two components—match conditions and actions.
    - Match conditions—Specify certain characteristics that the packet must match for the action to be performed. You can define various match conditions, such as the IP source address field, IP destination address field, and IP protocol field.
    - Action—Specify what is to be done with the packet if it matches the match conditions. Possible actions are to accept, discard, or reject a packet; go to the next term; or take no action.

      You can specify only one `action` (or omit it) in a term, but you can specify any combination of action modifiers with it. Action modifiers include a default `accept` action. For example, if you specify an action modifier and do not specify an action, the specified action modifier is implemented and the packet is accepted.

The `packet-mode` action modifier specifies traffic to bypass flow-based forwarding. Like other action modifiers, you can configure the `packet-mode` action modifier along with other actions, such as `accept` or `count`.

3. Apply firewall filters to interfaces—Apply the firewall filter to the interface to have the firewall filter take effect.

When the packet comes in on an interface, the input packet filters configured on the interface are applied. If the packet matches the specified conditions and `packet-mode` action is configured, the packet bypasses the flow-based forwarding completely.

When configuring filters, be mindful of the order of the terms within the firewall filter. Packets are tested against each term in the order in which it is listed in the configuration. When the first matching conditions are found, the action associated with that term is applied to the packet and the evaluation of the firewall filter ends, unless the `next term` action modifier is included. If the `next term` action is included, the matching packet is then evaluated against the next term in the firewall filter; otherwise, the matching packet is not evaluated against subsequent terms in the firewall filter.

When configuring firewall filters for selective stateless packet-based services:

- Accurately identify traffic that needs to bypass flow to avoid unnecessary packet drops.
- Make sure to apply the firewall filter with packet-mode action on all interfaces involved in the packet-based flow path.
- Make sure to configure host-bound TCP traffic to use flow-based forwarding—exclude this traffic when specifying match conditions for the firewall filter term containing the `packet-mode` action modifier. Any host-bound TCP traffic configured to bypass flow is dropped. Asynchronous flow-mode processing is not supported with selective stateless packet-based services.
- Configure input packet filters (not output) with the `packet-mode` action modifier.

NOTE: Nested firewall filters (configuring a filter within the term of another filter) are not supported with selective stateless packet-based services.

Some typical deployment scenarios where you can configure selective stateless packet-based services are as follows:

- Traffic flow between private LAN and WAN interfaces, such as for Intranet traffic, where end-to-end forwarding is packet-based
- Traffic flow between private LAN and not-so-secure WAN interfaces, where traffic uses packet-based and flow-based forwarding for secure and not so secure traffic respectively
- Traffic flow between the private LAN and WAN interface with failover to flow-based IPsec WAN when the private WAN link is down

- Traffic flow from flow-based LAN to packet-based MPLS WAN

Example: Configuring Selective Stateless Packet-Based Services for End-to-End Packet-Based Forwarding

IN THIS SECTION

This example shows how to configure selective stateless packet-based services for end-to-end packet-based forwarding. This feature is supported on the SRX300, SRX320, SRX340, SRX345, and vSRX Virtual Firewall devices

Requirements

Before you begin:

- Understand how to configure stateless firewall filters.
- Establish basic connectivity. .

Overview

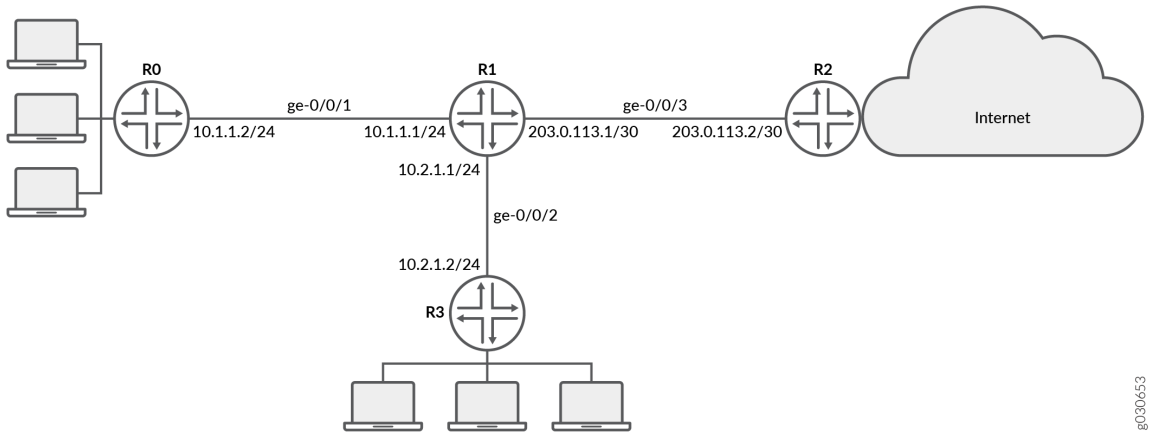
In this example, you configure the IP addresses for the interfaces on each of the devices. For R0 it is 10.1.1.2/24 ; for R1 they are 10.1.1.1/24, 10.2.1.1/24, and 203.0.113.1/30; for R2 it is 203.0.113.2/30; and for R3 it is 10.2.1.2/24. You create static routes and associate next-hop addresses for the devices as follows: R0 is 10.1.1.2, R1 is 198.51.100.2, R2 is 203.0.113.1, and R3 is 10.2.1.1.

Then on device R1 you configure a zone called untrust and assign it to interface ge-0/0/3. You also create a zone called trust and assign interfaces ge-0/0/1 and ge-0/0/2 to it. You configure trust and untrust zones to allow all supported application services as inbound services. You allow traffic from any source address, destination address, and application to pass between the zones.

You then create the firewall filter bypass-flow-filter and define the terms bypass-flow-term-1 and bypass-flow-term-2 that match the traffic between internal interfaces ge-0/0/1 and ge-0/0/2 and that contain the packet-mode action modifier. You define the term accept-rest to accept all remaining traffic. Finally, you apply the firewall filter bypass-flow-filter to internal interfaces ge-0/0/1 and ge-0/0/2 (not on the external interface). As a result, all internal traffic bypasses flow-based forwarding and the traffic to and from the Internet does not bypass flow-based forwarding.

Figure 21 on page 174 shows the network topology used in this example.

Figure 21: Intranet Traffic Using End-to-End Packet-Based Services



Your company’s branch offices are connected to each other through a private WAN. For this internal traffic, packet forwarding is required because security is not an issue. Hence for this traffic, you decide to configure selective stateless packet-based services to bypass flow-based forwarding. The remaining traffic, to and from the Internet, uses flow-based forwarding.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
{device R0}
[edit]
set interfaces ge-0/0/1 description "Internal 1" unit 0 family inet address 10.1.1.2/24
set routing-options static route 0.0.0.0/0 next-hop 10.1.1.1
```

```
{device R1}
set interfaces ge-0/0/1 description "Internal 1" unit 0 family inet address 10.1.1.1/24
set interfaces ge-0/0/2 description "Internal 2" unit 0 family inet address 10.2.1.1/24
set interfaces ge-0/0/3 description "Internet" unit 0 family inet address 203.0.113.1/30
set routing-options static route 0.0.0.0/0 next-hop 203.0.113.2
set security zones security-zone untrust interfaces ge-0/0/3
set security zones security-zone trust interfaces ge-0/0/1
set security zones security-zone trust interfaces ge-0/0/2
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic system-services all
set security policies from-zone trust to-zone untrust policy Internet-traffic match source-
address any destination-address any application any
set security policies from-zone trust to-zone untrust policy Internet-traffic then permit
set security policies from-zone untrust to-zone trust policy Incoming-traffic match source-
address any destination-address any application any
set security policies from-zone untrust to-zone trust policy Incoming-traffic then permit
set security policies from-zone trust to-zone trust policy Intrazone-traffic match source-
address any destination-address any application any
set security policies from-zone trust to-zone trust policy Intrazone-traffic then permit
set firewall family inet filter bypass-flow-filter term bypass-flow-term-1 from source-address
10.1.1.0/24
set firewall family inet filter bypass-flow-filter term bypass-flow-term-1 from destination-
address 10.2.1.0/24
set firewall family inet filter bypass-flow-filter term bypass-flow-term-1 then packet-mode
set firewall family inet filter bypass-flow-filter term bypass-flow-term-2 from source-address
10.2.1.0/24
set firewall family inet filter bypass-flow-filter term bypass-flow-term-2 from destination-
address 10.1.1.0/24
```

```
set firewall family inet filter bypass-flow-filter term bypass-flow-term-2 then packet-mode
set firewall family inet filter bypass-flow-filter term accept-rest then accept
set interfaces ge-0/0/1 description "Internal 1" unit 0 family inet filter input bypass-flow-
filer
set interfaces ge-0/0/2 description "Internal 2" unit 0 family inet filter input bypass-flow-
filer
```

```
{device R2}
set interfaces ge-0/0/3 description "Internet" unit 0 family inet address 10.1.1.2/30
set routing-options static route 0.0.0.0/0 next-hop 10.1.1.1
```

```
{device R3}
[edit]
set interfaces ge-0/0/2 description "Internal 2" unit 0 family inet address 10.2.1.2/24
set routing-options static route 0.0.0.0/0 next-hop 10.2.1.1
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure selective stateless packet-based services for end-to-end packet-based forwarding:

1. Configure the IP addresses for the interfaces on devices R0, R1, R2, and R3.

    ```
    {device R0}
    [edit]
    user@host# set interfaces ge-0/0/1 description "Internal 1" unit 0 family inet address 10.1.1.2/24
    ```

    ```
    {device R1}
    [edit]
    user@host# set interfaces ge-0/0/1 description "Internal 1" unit 0 family inet address 10.1.1.1/24
    ```

```
user@host# set interfaces ge-0/0/2 description "Internal 2" unit 0 family inet address 10.2.1.1/24
user@host#   set interfaces ge-0/0/3 description "Internet" unit 0 family inet address 203.0.113.1/30
```

```
{device R2}
[edit]
user@host# set interfaces ge-0/0/3 description "Internet" unit 0 family inet address 203.0.113.1/30
```

```
{device R3}
[edit]
user@host# set interfaces ge-0/0/2 description "Internal 2" unit 0 family inet address 10.2.1.2/24
```

2. Create static routes and associate the appropriate next-hop addresses for devices R0, R1, R2, and R3.

```
{device R0}
[edit]
user@host# set routing-options static route 0.0.0.0/0 next-hop 10.1.1.1
```

```
{device R1}
[edit]
user@host# set routing-options static route 0.0.0.0/0 next-hop 203.0.113.1
```

```
{device R2}
[edit]
user@host# set routing-options static route 0.0.0.0/0 next-hop 203.0.113.2
```

```
{device R3}
[edit]
user@host# set routing-options static route 0.0.0.0/0 next-hop 10.2.1.1
```

3. Configure security zones and assign interfaces.

```
{device R1}
[edit]
user@host# set security zones security-zone untrust interfaces ge-0/0/3
```

```
user@host# set security zones security-zone trust interfaces ge-0/0/1
user@host# set security zones security-zone trust interfaces ge-0/0/2
```

4. Configure application services for zones.

```
{device R1}
[edit]
user@host# set security zones security-zone trust host-inbound-traffic system-services all
user@host# set security zones security-zone untrust host-inbound-traffic system-services all
```

5. Configure a security policy

```
{device R1}
[edit]
user@host# set security policies from-zone trust to-zone untrust policy Internet-traffic match source-
address any destination-address any application any
user@host# set security policies from-zone trust to-zone untrust policy Internet-traffic then permit
user@host# set security policies from-zone untrust to-zone trust policy Incoming-traffic match source-
address any destination-address any application any
user@host# set security policies from-zone untrust to-zone trust policy Incoming-traffic then permit
user@host# set security policies from-zone trust to-zone trust policy Intrazone-traffic match source-
address any destination-address any application any
user@host# set security policies from-zone trust to-zone trust policy Intrazone-traffic then permit
```

6. Create a firewall filter and define terms for all the packet-based forwarding traffic.

```
{device R1}
[edit]
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term-1 from source-
address 10.1.1.0/24
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term-1 from destination-
address 10.2.1.0/24
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term-1 then packet-mode
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term-2 from source-
address 10.2.1.0/24
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term-2 from destination-
address 10.1.1.0/24
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term-2 then packet-mode
```

7. Specify another term for the remaining traffic.

```
{device R1}
[edit]
user@host# set firewall family inet filter bypass-flow-filter term accept-rest then accept
```

8. Apply the firewall filter to relevant interfaces.

```
{device R1}
[edit]
user@host# set interfaces ge-0/0/1 description "Internal 1" unit 0 family inet filter input bypass-flow-
filer
user@host# set interfaces ge-0/0/2 description "Internal 2" unit 0 family inet filter input bypass-flow-
filer
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show routing-options`, and `show firewall` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
{device R0}
[edit]
user@host# show interfaces
ge-0/0/1 {
    description “Internal 1”
    unit 0 {
        family inet {
            address 10.1.1.2/24
        }
    }
}
```

```
{device R0}
[edit]
user@host# show routing-options
static {
    route 0.0.0.0/0 next-hop 10.1.1.1;
}
```

```
{device R2}
[edit]
user@host# show interfaces
ge-0/0/3 {
    description “Internet”
    unit 0 {
        family inet {
            address 203.0.113.2/30;
        }
    }
}
```

```
{device R2}
[edit]
user@host# show routing-options
static {
    route 0.0.0.0/0 next-hop 203.0.113.1;
}
```

```
{device R3}
[edit]
user@host# show interfaces
ge-0/0/2 {
    description “Internal 2”
    unit 0 {
        family inet {
            address 10.2.1.2/24;
        }
    }
}
```

```
{device R3}
user@host# show routing-options
static {
    route 0.0.0.0/0 next-hop 10.2.1.1;
}
```

```
{device R1}
[edit]
user@host# show interfaces
ge-0/0/1 {
    description “internal 1”
    unit 0 {
        family inet {
            filter {
                input bypass-flow-filter;
                }
            address 10.1.1.1/24;
        }
    }
}
ge-0/0/2 {
    description “Internal 2”
    unit 0 {
        family inet {
            filter {
                input bypass-flow-filter;
                }
            address 10.2.1.1/24;
        }
    }
}
ge-0/0/3 {
    description “Internet”
    unit 0 {
        family inet {
            address 203.0.113.1/30;
        }
    }
}
{device R1}
[edit]
user@host# show routing-options
static {
    route 0.0.0.0/0 next-hop 203.0.113.1;
}
{device R1}
[edit]
user@host# show firewall
family inet {
    filter bypass-flow-filter {
        term bypass-flow-term-1 {
            from {
                source-address {
                    10.1.1.0/24;
                }
                destination-address {
                    10.2.1.0/24;
                }
            }
            then packet-mode;
        }
        term bypass-flow-term-2 {
            from {
                source-address {
                    10.2.1.0/24;
                }
                destination-address {
                    10.1.1.0/24;
                }
            }
            then packet-mode;
        }
        term accept-rest {
            then accept;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying the End-to-End Packet-Based Configuration

Purpose

Verify that the selective stateless packet-based services are configured.

Action

From configuration mode, enter the `show interfaces`, `show routing-options`, `show security zones`, `show security policies`, and `show firewall` commands.

Verify that the output shows the intended configuration of the firewall filter, interfaces, and policies.

Verify that the terms are listed in the order in which you want the packets to be tested. You can move terms within a firewall filter by using the `insert` command.

Verifying Session Establishment on Intranet Traffic

Purpose

Verify that sessions are established when traffic is transmitted to interfaces within the Intranet.

Action

To verify that sessions are established, perform the following tasks:

1. On device `R1`, enter the operational mode `clear security flow session all` command to clear all existing security flow sessions.

2. On device `R0`, enter the operational mode `ping` command to transmit traffic to device `R3`.

3. On device R1, with traffic transmitting from devices R0 to R3 through R1, enter the operational mode `show security flow session` command.

```
Flow Sessions on FPC10 PIC1:
Total sessions: 0

Flow Sessions on FPC10 PIC2:
Total sessions: 0

Flow Sessions on FPC10 PIC3:
Total sessions: 0
```

NOTE: To verify established sessions, make sure to enter the `show security flow session` command while the `ping` command is sending and receiving packets.

```
{device R0}
user@host> ping 203.0.113.6
```

```
PING 203.0.113.6 (203.0.113.6): 56 data bytes
64 bytes from 203.0.113.6: icmp_seq=0 ttl=63 time=2.326 ms
64 bytes from 203.0.113.6: icmp_seq=1 ttl=63 time=2.569 ms
64 bytes from 203.0.113.6: icmp_seq=2 ttl=63 time=2.565 ms
64 bytes from 203.0.113.6: icmp_seq=3 ttl=63 time=2.563 ms
64 bytes from 203.0.113.6: icmp_seq=4 ttl=63 time=2.306 ms
64 bytes from 203.0.113.6: icmp_seq=5 ttl=63 time=2.560 ms
64 bytes from 203.0.113.6: icmp_seq=6 ttl=63 time=4.130 ms
64 bytes from 203.0.113.6: icmp_seq=7 ttl=63 time=2.316 ms
...
```

```
{device R1}
user@host> show security flow session
```

```
Flow Sessions on FPC10 PIC1:
Total sessions: 0

Flow Sessions on FPC10 PIC2:
```

```
Total sessions: 0

Flow Sessions on FPC10 PIC3:
Total sessions: 0
```

The output shows traffic transmitting from `R0` to `R3` and no sessions are established. In this example, you applied the `bypass-flow-filter` with the `packet-mode` action modifier on interfaces `Internal 1` and `Internal 2` for your company's Intranet traffic. This output verifies that the traffic between the two interfaces is correctly bypassing flow-based forwarding and hence no sessions are established.

Verifying Session Establishment on Internet Traffic

Purpose

Verify that sessions are established when traffic is transmitted to the Internet.

Action

To verify that traffic to the Internet is using flow-based forwarding and sessions are established, perform the following tasks:

1. On device `R1`, enter the operational mode `clear security flow session all` command to clear all existing security flow sessions.
2. On device `R0`, enter the operational mode `ping` command to transmit traffic to device `R2`.
3. On device `R1`, with traffic transmitting from `R0` to `R2` through `R1`, enter the operational mode `show security flow session` command.

NOTE: To verify established sessions, make sure to enter the `show security flow session` command while the `ping` command is sending and receiving packets.

```
{device R0}
user@host> ping 10.2.1.2  -c 10
```

```


PING 10.2.1.2 (10.2.1.2) 56(84) bytes of data.
64 bytes from 10.2.1.2: icmp_seq=1 ttl=63 time=6.07 ms
64 bytes from 10.2.1.2: icmp_seq=2 ttl=63 time=4.24 ms
64 bytes from 10.2.1.2: icmp_seq=3 ttl=63 time=2.85 ms
```

```
64 bytes from 10.2.1.2: icmp_seq=4 ttl=63 time=6.14 ms
...
```

```
{device R1}
user@host>show security flow session
```

```
Flow Sessions on FPC10 PIC1:

Session ID: 410000077, Policy name: Internet-traffic/5, Timeout: 2, Valid
  In: 10.1.1.2/3 --> 10.2.1.2/32055;icmp, If: ge-0/0/1.0, Pkts: 1, Bytes: 84, CP Session ID: 
410000198
  Out: 10.2.1.2/32055 --> 10.1.1.2/3;icmp, If: ge-0/0/2.0, Pkts: 1, Bytes: 84, CP Session ID: 
410000198
Total sessions: 1

Flow Sessions on FPC10 PIC2:

Session ID: 420000079, Policy name: Internet-traffic/5, Timeout: 2, Valid
  In: 10.1.1.2/5 --> 10.2.1.2/32055;icmp, If: ge-0/0/1.0, Pkts: 1, Bytes: 84, CP Session ID: 
420000163
  Out: 10.2.1.2/32055 --> 10.1.1.2/5;icmp, If: ge-0/0/2.0, Pkts: 1, Bytes: 84, CP Session ID: 
420000163
Total sessions: 1

Flow Sessions on FPC10 PIC3:

Session ID: 430000090, Policy name: Internet-traffic/5, Timeout: 4, Valid
  In:10.1.1.2/7 --> 10.2.1.2/32055;icmp, If: ge-0/0/1.0, Pkts: 1, Bytes: 84, CP Session ID: 
430000088
  Out: 10.2.1.2/32055 --> 10.1.1.2/7;icmp, If: ge-0/0/2.0, Pkts: 1, Bytes: 84, CP Session ID: 
430000088
Total sessions: 1
```

The output shows traffic transmitting from devices `R0` to `R1` and established sessions. In this example, you did not apply the `bypass-flow-filter` with the `packet-mode` action modifier on interface `Internet` for your company's Internet traffic. This output verifies that the traffic to the Internet is correctly using flow-based forwarding and hence sessions are established.

Transmit traffic from device `R3` to `R2` and use the commands in this section to verify established sessions.

Example: Configuring Selective Stateless Packet-Based Services for Packet-Based to Flow-Based Forwarding

IN THIS SECTION

This example shows how to configure selective stateless packet-based services for packet-based to flow-based forwarding. This feature is supported on SRX300, SRX320, SRX340, SRX345, and vSRX Virtual Firewall devices.

Requirements

Before you begin:

- Understand how to configure stateless firewall filters.
- Establish basic connectivity. .

Overview

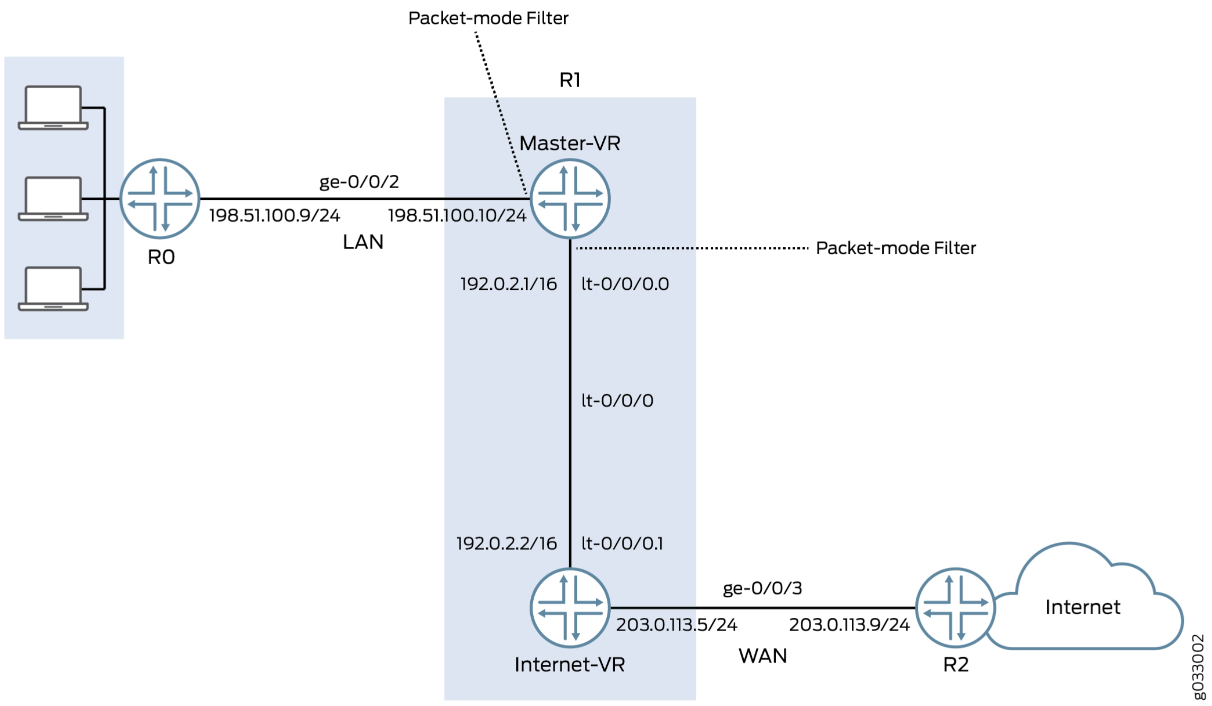
In this example, you configure the IP addresses for the interfaces on each of the devices. For device R0 as 198.51.100.9/24; for R1 the are198.51.100.10/24 and 203.0.113.5/24; and for R2 it is 203.0.113.9/24. On device R1, you set an internal service interface lt-0/0/0 between routing instances and configure a peer relationship between two virtual devices. You then create two security zones, Primary-VR-zone and Internet-VR-zone, assign related interfaces to them, and configure them to allow all supported applications and protocols.

Then you configure policies and specify that all packets are permitted. You configure a virtual device routing instance Internet-VR and assign interfaces for flow-based forwarding. You enable OSPF on devices R0, R1, and R2. On Device R2, you configure the filter bypass-flow-filter with the term bypass-flow-term that contains the packet-mode action modifier. Because you have not specified any match conditions, this filter applies to all traffic that traverses the interfaces on which it is applied.

Finally, on device R1 you apply the firewall filter bypass-flow-filter to internal interfaces ge-0/0/2.0 and lt-0/0/0.0. You do not apply the filter to the interfaces associated with the Internet-VR routing instance. As a result, all traffic that traverses the LAN interfaces associated with the primary routing instance uses packet-based forwarding and all traffic that traverses the Internet-VR routing instance uses flow-based forwarding.

Figure 22 on page 188 shows the network topology used in this example.

Figure 22: Selective Stateless Packet-Based Services for Packet-Based Forwarding



The interface facing the private LAN does not need any security services, but the interface facing the WAN needs security. In this example, you decide to configure both packet-based and flow-based forwarding for secure and not so secure traffic by configuring two routing instances—one handling the packet-based forwarding and the other handling the flow-based forwarding.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
{device R0}
set interfaces description “Connect to Primary VR” ge-0/0/2 unit 0 family inet address
198.51.100.9/24
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
```

```
{device R1}
set interfaces description “Connect to R0” ge-0/0/2 unit 0 family inet address 198.51.100.10/24
set interfaces description “Connect to R2” ge-0/0/3 unit 0 family inet address 203.0.113.5/24
set interfaces lt-0/0/0 unit 0 encapsulation frame-relay dlci 100 peer-unit 1 family inet
address 192.0.2.1/16
set interfaces lt-0/0/0 unit 1 encapsulation frame-relay dlci 100 peer-unit 0 family inet
address 192.0.2.2/16
set security zones security-zone Primary-VR-zone host-inbound-traffic system-services all
set security zones security-zone Primary-VR-zone host-inbound-traffic protocols all
set security zones security-zone Primary-VR-zone interfaces ge-0/0/2.0
set security zones security-zone Primary-VR-zone interfaces lt-0/0/0.0
set security zones security-zone Internet-VR-zone host-inbound-traffic system-services all
set security zones security-zone Internet-VR-zone host-inbound-traffic protocols all
set security zones security-zone Internet-VR-zone interfaces ge-0/0/3.0
set security zones security-zone Internet-VR-zone interfaces lt-0/0/0.1
set security policies default-policy permit-all
set routing-instances Internet-VR instance-type virtual-router interface lt-0/0/0.1
set routing-instances Internet-VR instance-type virtual-router interface ge-0/0/3.0
set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
set protocols ospf area 0.0.0.0 interface lt-0/0/0.0
set routing-instances Internet-VR protocols ospf area 0.0.0.0 interface lt-0/0/0.1
set routing-instances Internet-VR protocols ospf area 0.0.0.0 interface ge-0/0/3.0
set firewall family inet filter bypass-flow-filter term bypass-flow-term then accept
set firewall family inet filter bypass-flow-filter term bypass-flow-term then packet-mode
set interfaces ge-0/0/2 unit 0 family inet filter input bypass-flow-filter
set interfaces lt-0/0/0 unit 0 family inet filter input bypass-flow-filter
```

```
{device R2}
set interfaces description “Connect to Internet-VR” ge-0/0/3 unit 0 family inet address
203.0.113.9/24
set protocols ospf area 0.0.0.0 interface ge-0/0/3
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure selective stateless packet-based services for end-to-end packet-based forwarding:

1. Configure the IP addresses for the interfaces.

    ```
    {device R0}
    [edit]
    user@host# set interfaces description “Connect to Primary VR” ge-0/0/2 unit 0 family inet address
    198.51.100.9/24
    ```

    ```
    {device R1}
    [edit]
    user@host# set interfaces description “Connect to R0” ge-0/0/2 unit 0 family inet address
    198.51.100.10/24
    user@host# set interfaces description “Connect to R2” ge-0/0/3 unit 0 family inet address
    203.0.113.5/24
    ```

    ```
    {device R2}
    [edit]
    user@host# set interfaces description “Connect to Internet-VR” ge-0/0/3 unit 0 family inet address
    203.0.113.9/24
    ```

2. Set an internal service interface between routing instances.

```
{device R1}
[edit]
user@host# set interfaces lt-0/0/0 unit 0 encapsulation frame-relay dlci 100 peer-unit 1 family inet
address 192.0.2.1/16
user@host# set interfaces lt-0/0/0 unit 1 encapsulation frame-relay dlci 100 peer-unit 0 family inet
address 192.0.2.2/16
```

3. Configure security zones.

```
{device R1}
[edit]
user@host# set security zones security-zone Primary-VR-zone host-inbound-traffic system-services all
user@host# set security zones security-zone Primary-VR-zone host-inbound-traffic protocols all
user@host# set security zones security-zone Primary-VR-zone interfaces ge-0/0/2.0
user@host# set security zones security-zone Primary-VR-zone interfaces lt-0/0/0.0
user@host# set security zones security-zone Internet-VR-zone host-inbound-traffic system-services all
user@host# set security zones security-zone Internet-VR-zone host-inbound-traffic protocols all
user@host# set security zones security-zone Internet-VR-zone interfaces ge-0/0/3.0
user@host# set security zones security-zone Internet-VR-zone interfaces lt-0/0/0.1
```

4. Configure policies.

```
{device R1}
[edit]
user@host# set security policies default-policy permit-all
```

5. Configure a virtual device routing instance.

```
{device R1}
[edit]
user@host# set routing-instances Internet-VR instance-type virtual-router interface lt-0/0/0.1
user@host# set routing-instances Internet-VR instance-type virtual-router interface ge-0/0/3.0
```

6. Enable OSPF on all interfaces in the network.

```
{device R0}
[edit]
user@host# set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
```

```
{device R1 for Primary-VR}
[edit]
user@host# set protocols ospf area 0.0.0.0 interface ge-0/0/2.0
user@host# set protocols ospf area 0.0.0.0 interface lt-0/0/0.0
```

```
{device R1 for Internet-VR}
[edit]
user@host#  set routing-instances Internet-VR protocols ospf area 0.0.0.0 interface lt-0/0/0.1
user@host#  set routing-instances Internet-VR protocols ospf area 0.0.0.0 interface ge-0/0/3.0
```

```
{device R2}
[edit]
user@host# set protocols ospf area 0.0.0.0 interface ge-0/0/3
```

7. Create a firewall filter and define a term for packet-based forwarding traffic.

```
{device R1}
[edit]
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term then accept
user@host# set firewall family inet filter bypass-flow-filter term bypass-flow-term then packet-mode
```

8. Apply the firewall filter to relevant interfaces.

```
{device R1}
[edit]
user@host# set interfaces ge-0/0/2 unit 0 family inet filter input bypass-flow-filter
user@host# set interfaces lt-0/0/0 unit 0 family inet filter input bypass-flow-filter
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces`, `show protocols`, `show security`, `show routing-instances`, and `show firewall` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
{device R0}
[edit]
user@host# show interfaces
ge-0/0/2 {
    description “Connect to Primary-VR”
    unit 0 {
        family inet {
            address 198.51.100.9/24
        }
    }
}
```

```
{device R0}
[edit]
user@host# show protocols
ospf {
    area 0.0.0.0/0 {
        interface ge-0/0/2.0;
    }
}
```

```
{device R2}
[edit]
user@host# show interfaces
ge-0/0/3 {
    description “Connect to Internet-VR”
    unit 0 {
        family inet {
            address 203.0.113.9/24;
        }
    }
}
```

```
{device R2}
[edit]
user@host# show protocols
ospf {
    area 0.0.0.0/0 {
        interface ge-0/0/3.0;
    }
}
```

```
{device R1}
[edit]
user@host# show interfaces
ge-0/0/2 {
    description “Connect to R0”
    unit 0 {
        family inet {
            filter {
                input bypass-flow-filter;
                }
            address 198.51.100.10/24;
        }
    }
}
lt-0/0/0 {
    unit 0 {
        encapsulation frame-relay;
        dlci 100;
        peer-unit 1;
        family inet {
            filter {
                input bypass-flow-filter
            }
            address 192.0.2.1/16;
        }
    }
    unit 1{
        encapsulation frame-relay;
        dlci 100;
        peer-unit 0;
        family inet {
            address 192.0.2.2/16 ;
        }
    }
}
{device R1}
[edit]
user@host# show protocols
ospf {
    area 0.0.0.0/0 {
        interface ge-0/0/2.0;
        interface lt-0/0/0.0;
    }
}
{device R1}
[edit]
user@host# show firewall
filter bypass-flow-filter {
    term bypass-flow-term {
        then {
            packet-mode;
            accept;
        }
    }
}
```

```
{device R1}
[edit]
user@host# show routing-instances
Internet-VR {
    instance-type virtual-router;
    interface lt-0/0/0.1;
    interface ge-0/0/3.0;
    protocols {
        ospf {
            area 0.0.0.0 {
                interface ge-0/0/3.0;
                lt-0/0/0.1;
            }
        }
    }
}
```

```
{device R1}
[edit]
user@host# show security
security zone Primary-VR-zone {
    host-inbound-traffic {
        system-services {
            all;
            {
            protocols {
            all;
        {
    {
    intefaces {
    ge-0/0/2.0;
    lt-0/0/0.0;
    {
{
security zone Internet-VR-zone {
    host-inbound-traffic {
        system-services {
            all;
        {
        protocols {
            all;
        }
    }
    intefaces {
    ge-0/0/3.0;
    lt-0/0/0.1;
    {
{
policies {
    default-policy {
        permit-all;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying the Packet-Based to Flow-Based Configuration

Purpose

Verify that the selective stateless packet-based services are configured for packet-based to flow-based forwarding.

Action

From configuration mode, enter the `show interfaces`, `show protocols`, `show security`, `show routing-instances`, and `show firewall` commands.

Verify that the output shows the intended configuration of the firewall filter, routing instances, interfaces, and policies.

Verify that the terms are listed in the order in which you want the packets to be tested. You can move terms within a firewall filter by using the `insert` command.

Verifying Session Establishment on LAN Traffic

Purpose

Verify that the sessions are established when traffic is transmitted on interfaces within the LAN.

Action

To verify that sessions are established, perform the following tasks:

1. On device R1, from operational mode enter the `clear security flow session all` command to clear all existing security flow sessions.
2. On device R0, from operational mode enter the `ping` command to transmit traffic to device Primary-VR.
3. On device R1, with traffic transmitting from devices R0 through R1, from operational mode enter the `show security flow session` command.

NOTE: To verify established sessions, ensure that you enter the `show security flow session` command while the `ping` command is sending and receiving packets.

```
{device R0}
user@host> ping 192.0.2.1
```

```
PING 192.0.2.1 (192.0.2.1): 56 data bytes
64 bytes from 192.0.2.1: icmp_seq=0 ttl=63 time=2.208 ms
64 bytes from 192.0.2.1: icmp_seq=1 ttl=63 time=2.568 ms
64 bytes from 192.0.2.1: icmp_seq=2 ttl=63 time=2.573 ms
64 bytes from 192.0.2.1: icmp_seq=3 ttl=63 time=2.310 ms
64 bytes from 192.0.2.1: icmp_seq=4 ttl=63 time=1.566 ms
64 bytes from 192.0.2.1: icmp_seq=5 ttl=63 time=1.569 ms
...
```

```
{device R1}
user@host> show security flow session
```

```
0 sessions displayed
```

The output shows traffic transmitting from R0 to Primary-VR and no sessions are established. In this example, you applied the `bypass-flow-filter` with the `packet-mode` action modifier on interfaces `ge-0/0/0` and `lt-0/0/0.0` for your company's LAN traffic. This output verifies that the traffic between the two interfaces is correctly bypassing flow-based forwarding and hence no sessions are established.

Verifying Session Establishment on Internet Traffic

Purpose

Verify that sessions are established when traffic is transmitted to the Internet.

Action

To verify that traffic to the Internet is using flow-based forwarding and sessions are established, perform the following tasks:

1. On device `R1`, from operational mode enter the `clear security flow session all` command to clear all existing security flow sessions.
2. On device `R0`, from operational mode enter the `ping` command to transmit traffic to device `R2`.
3. On device `R1`, with traffic transmitting from `R0` to `R2` through `R1`, from operational mode enter the `show security flow session` command.

```
root@host> show security flow session
Flow Sessions on FPC10 PIC1:
Total sessions: 0

Flow Sessions on FPC10 PIC2:
Total sessions: 0

Flow Sessions on FPC10 PIC3:
Total sessions: 0
```

NOTE: To verify established sessions, ensure that you enter the `show security flow session` command while the `ping` command is sending and receiving packets.

```
{device R0}
user@host> ping 192.0.2.1 -c 10
```

```
PING 60.0.0.1 (60.0.0.1) 56(84) bytes of data.
64 bytes from 192.0.2.1: icmp_seq=1 ttl=64 time=1.98 ms
64 bytes from 192.0.2.1: icmp_seq=2 ttl=64 time=1.94 ms
64 bytes from 192.0.2.1: icmp_seq=3 ttl=64 time=1.92 ms
```

```
64 bytes from 192.0.2.1: icmp_seq=4 ttl=64 time=1.89 ms

...
```

```
{device R1}
user@host> show security flow session
```

```

Session ID: 189900, Policy name: default-policy/2, Timeout: 2
  In: 198.51.100.9/0 --> 192.0.2.1/5924;icmp, If: lt-0/0/0.1
  Out: 192.0.2.1/5924 --> 198.51.100.9/0;icmp, If: ge-0/0/3.0

Session ID: 189901, Policy name: default-policy/2, Timeout: 2
  In: 198.51.100.9/1 --> 192.0.2.1/5924;icmp, If: lt-0/0/0.1
  Out: 192.0.2.1/5924 --> 198.51.100.9/1;icmp, If: ge-0/0/3.0

Session ID: 189902, Policy name: default-policy/2, Timeout: 4
  In: 198.51.100.9/2 --> 192.0.2.1/5924;icmp, If: lt-0/0/0.1
  Out: 192.0.2.1/5924 --> 198.51.100.9/2;icmp, If: ge-0/0/3.0

 3 sessions displayed
```

The output shows traffic transmitting from devices `R0` to `R2` and established sessions. In this example, you did not apply the `bypass-flow-filter` with the `packet-mode` action modifier on routing instance `Internet-VR` for your company's Internet traffic. This output verifies that the traffic to the Internet is correctly using flow-based forwarding and hence sessions are established.

Note that sessions are established only when traffic is flowing between `lt-0/0/0.1` and `ge-0/0/3` and not when traffic is flowing between `ge-0/0/2` and `lt-0/0/0.0`.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D30	Starting in Junos OS Release 15.1X49-D30 and Junos OS Release 17.3R1, the session flow summaries include CP session IDs.

RELATED DOCUMENTATION

Express Path Overview | 204

Reverse Route Packet Mode using Virtual Router

IN THIS SECTION

During flow processing, when the traffic route between the server and client is changed, the traffic reroutes using the virtual router (VR). The VR used in rerouting is available in the interface or the filter-based forwarding (FBF). The behavior of the reroute is monitored using the `set security flow advanced-options reverse-route-packet-mode-vr` command.

NOTE: The `reverse-route-packet-mode-vr` command works on root logical system and is enabled globally.

When the reverse route option is enabled, there is no change in the packet flow. When the reverse route option is disabled, the route lookup uses the VR from the packet incoming interface. If the VR in the route is incorrectly configured, then the traffic between the server and client is dropped.

NOTE: The resolve reserve route in the flow first path is not configured as the VR information from the client to the server packet is not available.

For example, Figure 23 on page 202 shows the behavior of the packet flow when the `reverse-route-packet-mode-vr` command is not configured. The client to server traffic uses the routing instance VR2 of incoming interface ge-0/0/0.0 to route the traffic. The server to client traffic also uses the routing instance VR2 of incoming interface ge-0/0/0.0 to route the traffic.

Figure 23: Reverse Route Disabled

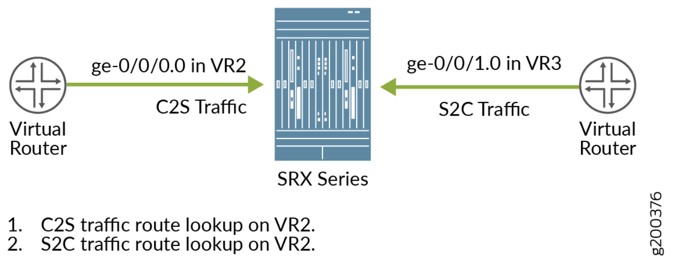


Figure 24 on page 202 shows the behavior of the packet flow when the `reverse-route-packet-mode-vr` command is configured using interface. The client to server traffic uses the routing instance VR2 of incoming interface ge-0/0/0.0 to route the traffic. The server to client traffic uses the routing instance VR3 of interface ge-0/0/1.0 to route the traffic.

Figure 24: Reverse Route Enabled with Interface

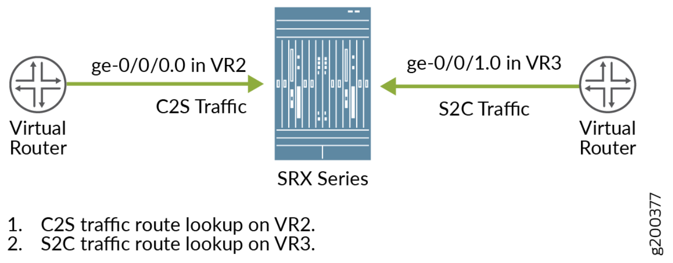
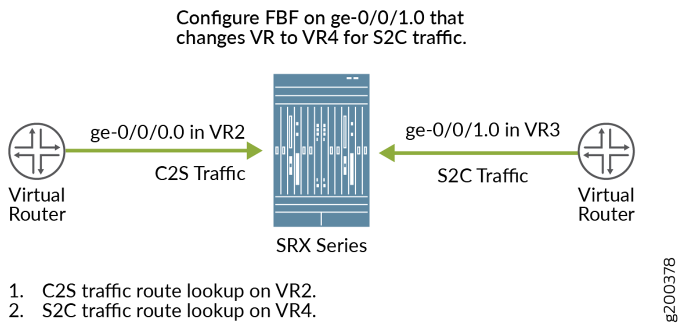


Figure 25 on page 203 shows the behavior of the packet flow when the `reverse-route-packet-mode-vr` command is configured using FBF. The client to server traffic uses the packet incoming interface ge-0/0/0.0 in VR2 to route the traffic. Configuring FBF on the interface ge-0/0/1.0 changes VR3 to VR4. The server to client traffic uses VR4 to route the traffic.

Figure 25: Reverse Route Enabled with FBF

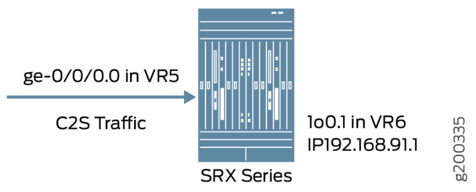

Configure FBF on ge-0/0/1.0 that changes VR to VR4 for S2C traffic.
ge-0/0/0.0 in VR2
C2S Traffic
ge-0/0/1.0 in VR3
S2C Traffic
Virtual Router
Virtual Router
SRX Series
1. C2S traffic route lookup on VR2.
2. S2C traffic route lookup on VR4.
g200378

Understanding To-host Traffic on Virtual Router

On a SRX Series Firewall, all the traffic that passes the firewall filter is referred as to-host traffic. The traffic from the firewall to the device is referred as the from-host traffic. The to-host traffic uses an egress interface and the from-host traffic uses an ingress interface. If both the interfaces are not in the same routing instance, there will be a session mismatched. To overcome this issue, the to-host and the from-host traffic choose interfaces that are available in the same routing instance.

Figure 26 on page 203 shows the to-host traffic using the routing instance VR5 of interface ge-0/0/0.0 and the routing instance VR6 of destination interface lo0.1.

Figure 26: To-host Traffic on VR



For example, if the to-host traffic uses a local interface (such as local....X) that is in routing instance 5 (VR5), and the from-host traffic uses the interface in routing instance 6 (VR6). The session output displaying the interface information of the to-host traffic is:

```
Session ID: 10000179, Policy name: pol1/4, Timeout: 2, Valid
  In: 192.168.90.1/4 --> 192.168.91.1/19050;icmp, Conn Tag: 0x0, If: xe-9/0/3.0, Pkts: 1, Bytes:
84, CP Session ID: 10000178
```

```
  Out: 192.168.91.1/19050 --> 192.168.90.1/4;icmp, Conn Tag: 0x0, If: .local..5, Pkts: 1, Bytes:
84, CP Session ID: 10000178
```

NOTE: The session output displays the local interface of the to-host traffic as local....5.

To synchronize the to-host and from-host traffics, the to-host traffic uses traffic destination IP interface (lo0.1) that is available in VR6. As the from-host traffic is using the interface available in VR6, the session matches. The session output displaying the interface information of the to-host traffic is:

```
Session ID: 10000179, Policy name: pol1/4, Timeout: 2, Valid
  In: 192.168.90.1/4 --> 192.168.91.1/19050;icmp, Conn Tag: 0x0, If: xe-9/0/3.0, Pkts: 1, Bytes:
84, CP Session ID: 10000178
  Out: 192.168.91.1/19050 --> 192.168.90.1/4;icmp, Conn Tag: 0x0, If: .local..6, Pkts: 1, Bytes:
84, CP Session ID: 10000178
```

NOTE: The session output displays the local interface of the to-host traffic as local....6.

RELATED DOCUMENTATION

Packet-Based Forwarding | 168

Express Path Overview

IN THIS SECTION

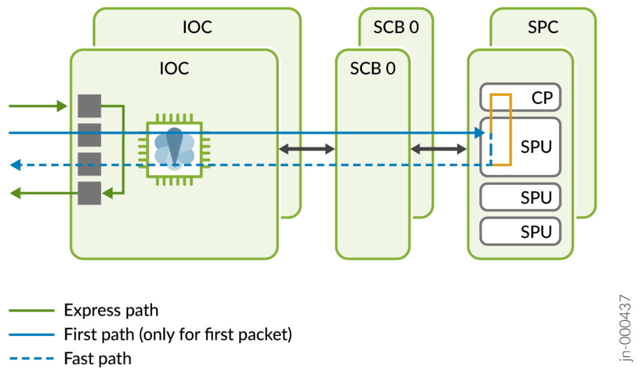
Express Path (formerly known as *services offloading*) is a mechanism for processing fast-path packets in the network processor instead of in the Services Processing Unit (SPU). Express Path increases the performance by offloading certain traffic from SPU to network processors.

When you create an Express Path session on the network processor, subsequent packets of the flow match the session on the network processors. The network processor then processes and forwards the packet. You cannot configure asymmetric routing between nodes because Express Path does not support HA forward for sessions between nodes.

The network processor also manages additional processing such as TCP sequence check, time-to-live (TTL) processing, Network Address Translation (NAT), and Layer 2 header translation. The flow table on the IOC3 is managed by the SPU of the flow module. The SPU inserts and deletes flow entries in the flow table based on policy matching results. Express Path supports IPv6.

The figure shows the packet flow in Express Path.

Figure 27: Packet flow and Express Path



Benefits of Express Path

- Significantly improves single-flow and chassis-level performance.
- Reduces SPU utilization and latency.

Express Path Limitations

Express Path does not support:

- Features
  - Transparent Mode
  - Multicast session with more than one fan-out

    - Fragmented packets
    - IPsec VPN
    - Different MTU size values
    - J-Flow
    - Flexible VLAN tagging
- Application Layer Gateway (ALG) data traffic:
    - DNS
    - IKE and ESP
    - PPTP
    - SQL-NET
- IPv6
    - NAT
    - Transparent mode
    - Different MTU size values
    - Class of Service (CoS) on egress interfaces

Express path and packet offloading doesn't work when you use firewall filters to direct traffic into a virtual router,

If you enable Express Path on a device operating in chassis cluster mode:

- You cannot configure asymmetric I/O cards (IOC).
- If a child link from the LACP-enabled reth interface goes down, all traffic on this link is distributed to other active child links of the interface. If the child link comes up and rejoins the reth interface, then the existing traffic or sessions are not redistributed over this newly rejoined active child link. New sessions traverse through this link.
- If a new child link is added to the LACP-enabled reth interface, then the existing traffic or sessions are not redistributed over this new child link. New sessions traverse this link.

Automated Express Path

Automated Express Path is by default enabled from Junos OS Release 21.2R1. When you upgrade to Junos Release 21.2R1 or later, you unlock free, unparalleled next-generation firewall performance, without any additional configuration or hardware investment. By default, an automated Express Path is enabled.

In Junos OS Release 21.2R1 to disable the Express Path per rule, use `set security policies from-zone [untrust] to-zone ptrust] policy [services-offload-pol1] then permit no-services-offload` command.

To revert to previous behavior by enabling services-offload per rule, use the `set security forwarding-options services-offload disable` command.

Automated Express Path supports the following features:

- Stateful Firewall
- Network Address Translation (NAT)
- Unified-Policies (with Dynamic-Applications and URL-Categories)
- User-Firewall
- Security Intelligence
- Intrusion Detection and Prevention (IDP)
- Enhanced Web-Filtering
- Application Layer Gateways (ALG)
- Screens (Anti-DDoS)

How does Express Path Process the Traffic?

When the first packet arrives at an interface, the network processor forwards it to the central point (CP). The central point in turn forwards the packet to the SPU. The SPU then creates a session on the network processor and verifies if the traffic qualifies for the Express Path session or a normal session.

If the traffic qualifies for Express Path processing, an Express Path session for the traffic is created in the SPU. The Express Path session processes the fast-path packets in the network processor, and the packets exit from the network processor.

If the traffic doesn't qualify for Express Path processing, the SPU creates a normal session. The normal session forwards packets from the network processor to the SPU for fast-path processing,

Express Path Network Processor

IN THIS SECTION

On SRX firewall with network processor, when all the plugins including packet plugins and stream plugins ignore a session, we service offload the session and then install the session on the network processor. When the packet plugin ignores the session, we mark the ignore flags. When the streaming plugin ignores the session, we mark the ignore flags and short-circuit the TCP-T and TCP-I. We then install the session on the network processor to offload the session.

The I/O card (IOC) network processor processes the fast-path packets without going through the switch fabric or the SPU. This reduces the packet-processing latency.

Each flow entry has a per-wing counter in the Express Path network processor. The counter captures the number of bytes that the network processor sends out over the wing.

The behavior of the network processor in different scenarios is as follows:

- **First-path flow**—The first-path flow is the same as the current network processor flow process. When the first packet arrives at the network processor, the network processor parses the TCP or the UDP packet to extract a 5-tuple key and then performs session lookup in the flow table. The network processor then forwards the first packet to the central point. The central point cannot find a match at this time because this is the first packet. The central point and the SPU create a session and match it against user-configured policies to determine if the session is a normal session or a services-offload session.

  If you specify the session to be managed with Express Path the SPU creates a session entry in the network processor flow table. This enables the Express Path flag in the session entry table; otherwise, the SPU creates a normal session entry in the network processor without the Express Path flag.

- **Fast-path flow**—After you create the session entry in the network processor, subsequent packets of the session will match the session entry table.

  1. If the Express Path flag is not set, then the network processor forwards the packet to the SPU s specified in the session entry table. The packet goes through the normal flow process.

  2. If the network processor finds the services-offload flag in the session entry table, it will process the packet locally and send the packet out directly.

3. The fast-forwarding function on the network processor supports one-fanout multicast sessions. The egress port in the session must also be associated with the same network processor as the ingress port. All other multicast cases need to be managed as normal sessions.

- **NAT process**—The SPU is responsible for mapping between the internal IP address or port and the external IP address or port. When the first packet of the session arrives, the SPU allocates the IP address or port mapping and stores the information in the network processor session entry. If the NAT flag is set, the network processor modifies the packet.
- **Session age-out**—To improve traffic throughput for services-offload sessions, a copy of a packet is sent to the SPU at every predefined time period to reduce the packet processing demand on the SPU. To limit the number of packet copies sent to the SPU, a timestamp is implemented for each service-offload session. The network processor calculates the elapsed time since the last session match. If the elapsed time is greater than the predefined time period, then the network processor sends a copy of the packet to the SPU and updates the session timestamp.
- **Session termination and deletion**—If the network processor receives an IP packet with a FIN (finished data) or an RST (reset connection) flag, it forwards the packet to the SPU. The SPU then deletes the session cache on the network processor. The network processor continues to receive and forward any packets to the SPU during state transition.

Wing Statistics Counter

In Express Path, the network processor provides the option for each flow entry to keep a per-wing bytes counter. The counter captures the number of bytes that the network processor sends out over the wing.

When you enable the counter, the network processor searches its flow entry (a session wing) for every ingress packet. If the packet belongs to an established flow entry, the network processor increases the byte counter of the flow entry in the packet. The network processor periodically copies a packet (copy-packet) of each flow entry to its associated SPU, allowing the SPU to maintain the session. The network processor sends flow-byte counter values in the header of copy-packet packets. The SPU accumulates and keeps per-wing statistics counters.

You cannot change the statistics configuration during the life cycle of a live session. Disabling or enabling the per-wing statistics configuration while a session is alive at the network processor invalidates the session statistics on the current session. The new session statistics can be valid only after the configuration changes are committed. Network processor per-wing counters cannot be cleared.

Sessions per Wing Statistics

The network processor has a larger static RAM (SRAM) to accommodate session resources, thus hosting more sessions per PIC. No Link Title displays the total number of session wings, including both Express Path and non-Express Path.

Use Feature Explorer to confirm platform and release support for specific features. Additional platforms may be supported.

See the "No Link Title" on page 210 section for more information.

Additional Platform Information

Use Feature Explorer to confirm platform and release support for specific features. Additional platforms may be supported.

Table 13: Total Number of Sessions per Wing in Network Processor Express Path Configuration Mode

Total Number of Wings		Number of Express Path UDP Wings		Number of Express Path TCP Wings	
Cards and SRX Series Firewall	**Non-Express Path Mode Sessions**	**Without Statistics**	**With Statistics**	**Without Statistics**	**With Statistics**
SRX5000 line device SRX5K-MPC (IOC2)	1.8 million	1.8 million	1.8 million	1.8 million	1.8 million
SRX5000 line device SRX5K-MPC3 (IOC3)	20 million	20 million	20 million	20 million	20 million
SRX5000 line device SRX5K IOC4	10 million	10 million	10 million	10 million	10 million
SRX4600	20 million	20 million	20 million	20 million	20 million

Express Path Packet Processing on IOC cards

Express Path on the IOC cards is based on processing fast-path packets through the network processor chipset instead of in the SPU to offload some basic firewall functions to the IOC card.

If you've enabled the Express Path feature, then the IOC card provides lower latency and also supports higher throughput by removing the overload on the SPU. The IOC card supports both intra-card traffic flow and inter-card traffic flow. To achieve the best latency results, both the ingress port and egress port of a traffic flow needs to be on the same XM chip of the IOC card.

The IOC card supports 240Gbps FPC and uses third generation Network Processing (NP) line of chipsets. This latest lookup and queuing chip is optimized for higher capacity. IOC card is compatible with SCB2 and SCB3, the earlier SCB is not supported.

You cannot power on all four PICs in the IOC card simultaneously because of power and thermal constrain. Power on a maximum of two PICs either in even or odd order. You can use the `set chassis fpc <slot> pic <pic> power off` command to choose the PIC to power on.

The system log messages are:

- XMCHIP_CMERROR_DDRIF_INT_REG_CHKSUM_ERR_MINOR
- XMCHIP_CMERROR_DDRIF_INT_REG_CHKSUM_ERR_MAJOR

The error messages indicate that the XM chip on a Flexible PIC Concentrator (FPC) has detected a checksum error, which is causing packet drops. The following error threshold values classify the error as a major error or a minor error:

- Minor error —> 5 errors per second
- Major error —> 255 errors per second (maximum count)

In the data plane, the IOC card parses packets and looks them up in the flow table. If the IOC card finds a match in the flow table, then it forwards packets based on the instructions given in the flow table. The IOC card can perform NAT, encapsulate the Layer 2 (L2) header, and forward the packets out of the egress interface. The egress interface can be located on the same IOC card (intra-card case) or another IOC card (inter-card case).

When the IOC card receives the first packet, it does not match any existing fast-forward session. The default hash-based forwarding is performed to send the first packet to the SPU. The SPU then creates the security session. If the SPU finds that the traffic is qualified for fast forwarding, and the related IOC card supports fast forwarding, it will install fast-forward session to the IOC card. If fast forwarding cannot be applied to the traffic, no session message is sent, and the IOC card uses the default hash-based forwarding to forward the packets to the SPU.

In fast-forward IOC card processing, if a fast-forward session is matched, the packet can be directly forwarded according to the session flow result. The IOC card takes all the necessary actions, for example, forwarding the packet, TTL checking and decreasing NAT translation, and Layer 2 header encapsulation.

In addition, the XL chip sends one copy of the forwarding packet to the SPU at a predefined time. This copy is used to refresh the SPU session, detect the current XL chip state, and so on. The SPU consumes this packet and does not forward it, because the real packet has been processed and transmitted.

Figure 28: IOC3 Intra-PFE Express Path

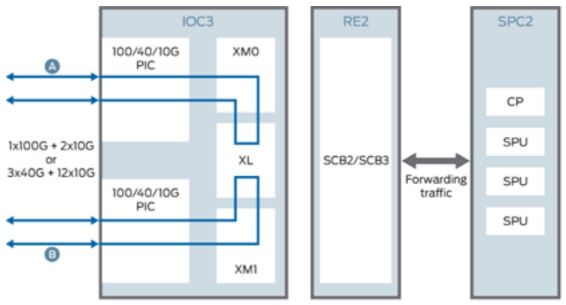


Figure 29: IOC3 Inter-PFE Express Path

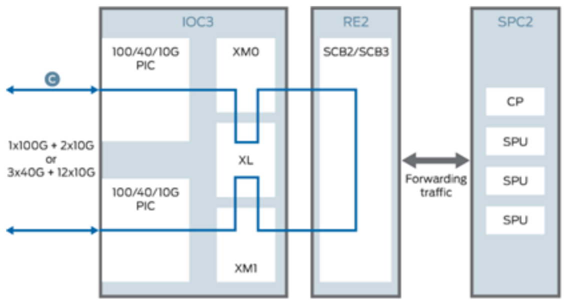
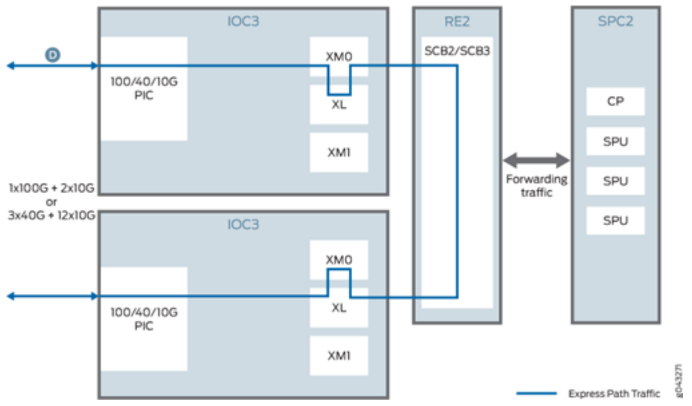


Figure 30: Inter-IOC3 Express Path



6

CHAPTER

Configuration Statements and Operational Commands

IN THIS CHAPTER

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Read this guide to learn about the syntax and options that make up the statements and commands. Also understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- Junos CLI Reference

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- Configuration Statements
- Operational Commands