NOKIA

Nokia Service Router Linux

PRODUCT OVERVIEW
RELEASE 21.11

3HE 17899 AAAA TQZZA
Issue 1

December 2021

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

This document includes Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

This document is intended for use by Nokia's customers ("You"/"Your") in connection with a product purchased or licensed from any company within Nokia Group of Companies. Use this document as agreed. You agree to notify Nokia of any errors you may find in this document; however, should you elect to use this document for any purpose(s) for which it is not intended, You understand and warrant that any determinations You may make or actions You may take will be based upon Your independent judgment and analysis of the content of this document.

Nokia reserves the right to make changes to this document without notice. At all times, the controlling version is the one available on Nokia's site.

No part of this document may be modified.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

Table of contents

1 About this guide

This document provides a basic overview of the Nokia Service Router Linux (SR Linux). This document is intended for marketing personnel, network technicians, administrators, operators, service providers, and others who need a basic understanding of the SR Linux.

Note:

This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Release Notes* for information about features supported in each load.

1.1 What's new

Topic	Location
Define new customer documents	SR Linux documentation
Define new hardware	7220 IXR-DL series
Define new supported protocols	Standards compliance and protocol support
Role-based access control	Securing access
DHCP relay for non-local IP-VRFs	DHCP relay
MPLS and LDP	MPLS feature support

1.2 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

ℹ **Tip:** Tip provides suggestions for use or best practices.

1.3 Conventions

SR Linux documentation uses the following command conventions.

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- An open right angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface). Example: **start** > **connect to**.
- Angle brackets (< >) indicate an item that is not used verbatim. For example, for the command **show ethernet <***name***>**, *name* should be replaced with the name of the interface.
- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

2 About SR Linux

SR Linux is a key component of a Nokia data center focused networking solution that delivers scalability, flexibility, and ease of operations for data centers and across hybrid- and multi-cloud environments. The SR Linux delivers:

- An open and extensible system that is fully programmable and scalable
- Model-driven management for simplified operations, integrations, and visibility
- Plug-and-play hardware integration
- Superior support for integrating community and customer-driven applications
- Customizable Command Line Interface (CLI) and on-demand CLI commands

2.1 What is SR Linux?

SR Linux is a Network Operating System (NOS) that leverages its routing protocol stack from Nokia SR OS, while also using Linux as its underlying operating system, allowing operators to use debugging and configuration tools that they are familiar with, even if they have no experience with SR OS.

Routing functions on SR Linux run as modular, lightweight applications, configurable via external APIs. These applications use gRPC and APIs to communicate with each other and external systems over TCP. The Nokia-supplied applications can be augmented by third-party developed applications, which plug into the SR Linux framework. Application-based functions allow for modular upgrades and easy fault isolation. Figure 1: SR Linux management framework shows the SR Linux management framework.

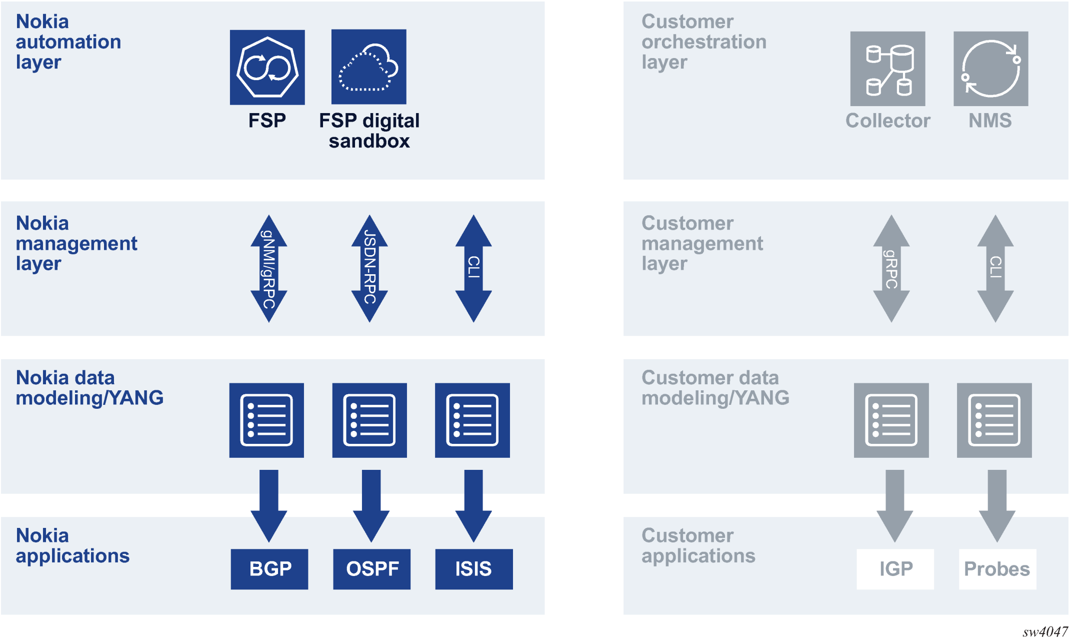

Figure 1: SR Linux management framework

2.2 Features overview

SR Linux supports a robust set of features. The sections that follow highlight major functionality.

2.2.1 Modular network applications

As a Linux-based NOS, SR Linux uses modular applications that are isolated in their own failure domains. A central application manager is responsible for the lifecycle of each application and provides full control of the protocols running on the system.

On SR Linux, each protocol (BGP, IS-IS, and so on) runs as its own application. These applications may be configured using external APIs, including CLI, gNMI, and JSON-RPC. The applications run like any others do in Linux. As well, users can integrate their own applications into SR Linux.

SR Linux uses an unmodified Linux kernel as its foundation to build a suite of network applications. This provides benefits such as reliability, portability, and ease of application development. Using an unmodified kernel also speeds the availability of non-Nokia applications (for example, OpenSSH) and security patches for operating system components.

2.2.2 Model-driven architecture

SR Linux makes extensive use of structured data models. Each application has a YANG model that defines its configuration and state. SR Linux exposes the YANG models to the supported management APIs. For example, the command tree in the CLI is derived from the SR Linux YANG models loaded into the system, and a gNMI client can use *Set* RPCs to configure an application based on its YANG model. When a configuration is committed, the SR Linux management server validates the YANG models and translates them into protocol buffers for the impart database (IDB).

See the SR Linux architecture overview chapter for more information about the relationship between IDB and SR Linux components).

2.2.3 IDB publish/subscribe model for messaging

IDB is a lightweight database that controls messaging between SR Linux applications, using a publish/subscribe (pub/sub) model. To do this, the IDB database is split up into topics. Each application owns a set of topics, to which it publishes information, and can subscribe to topics published by other applications. Applications subscribe to topics when they open a session to the IDB, and publish messages to their own topics for other applications to consume.

2.2.4 Protocol buffers and gRPC for inter-process communication

IDB stores data as protocol buffers (protobufs). Protobufs are a language-neutral, platform-neutral mechanism for serializing structured data. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs.

SR Linux uses gRPC for inter-process communication. gRPC is a client application that can directly call methods on a server application on a different machine as if it was a local object. The supported external APIs (CLI, gNMI, and JSON-RPC) communicate with the SR Linux and retrieve state information using gRPC.

SR Linux applications share state details with each other using the pub/sub model (see Figure 2: SR Linux infrastructure).

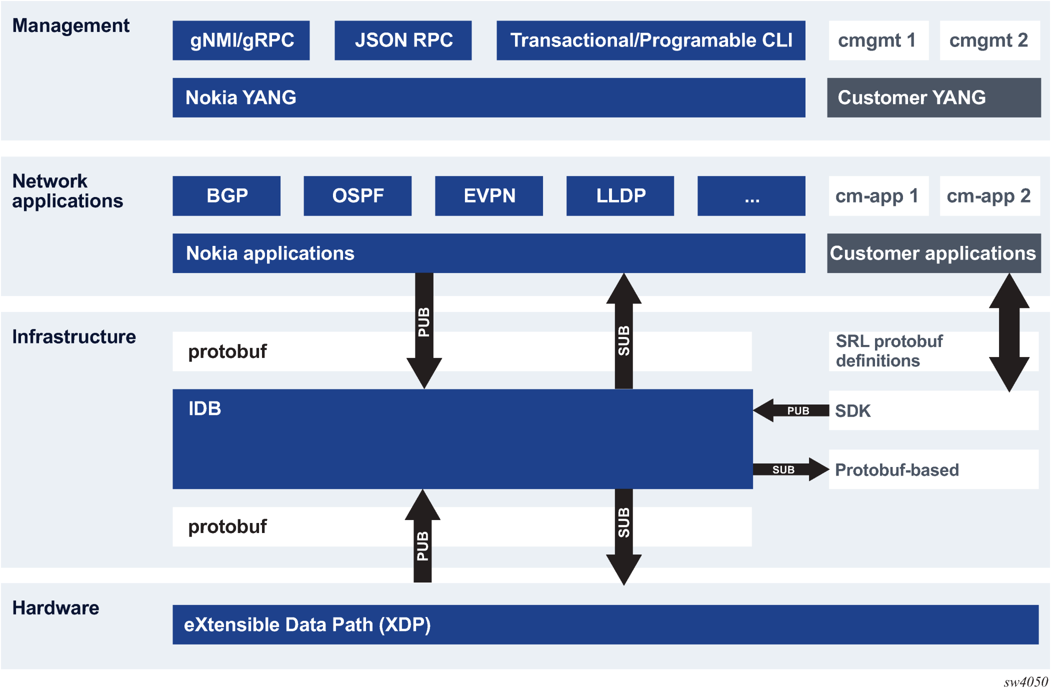

Figure 2: SR Linux infrastructure

2.2.5 Third-party application support

Third-party applications can be fully integrated into the SR Linux with the same functionality as Nokia applications. This includes configuration using YANG, telemetry support, and life-cycle management. Because third-party applications are not managed independently, it allows a reduction in operational overhead.

2.2.6 CLI plug-ins

The SR Linux CLI is itself an application that can load dynamic plugins from other applications. You can develop custom **show** commands and run them from the SR Linux CLI. The CLI plugins allow for integration with remote systems, supporting retrieval of state information.

2.2.7 Hardware extensibility

SR Linux supports a variety of network chipsets through the Nokia eXtensible Data Path (XDP). XDP serves as a hardware abstraction layer that facilitates adoption of new or non-Nokia network chipsets. It provides a common set of software instructions that northbound applications use so that they are not directly dependent on ASIC vendor SDKs. XDP borrows from the development experience for high-performance VNFs and makes use of user space acceleration for traffic destined for the control plane and any non-ASIC interfaces.

2.2.8 Software extensibility

Every SR Linux application, including third-party applications, supports its own YANG model, which can be loaded into the system. Operators can see and define the syntax and semantics of their application in a simple and standardized form. With this design, the YANG data model is defined first, then the CLI, APIs, and **show** output formats are derived from it.

SR Linux handles management and operations using the gRPC Network Management Interface (gNMI). Because SR Linux is natively model-driven, it can stream telemetry without requiring any translation layers. Telemetry is supported using POLL, ON_CHANGE, and ONCE streaming.

Third-party applications have access to the full streaming telemetry framework. This allows these applications to operate and be monitored, configured, and debugged the same as any other application on the system (see Figure 3: SR Linux software extensibility).

In addition to the gNMI interface, SR Linux includes a CLI and a JSON-RPC API for management. The CLI provides a framework for accessing the underlying data models of the system. The JSON-RPC API interface supports requests against the data models, as well as allowing a programmable interface access to the extensible plugin framework in the CLI.

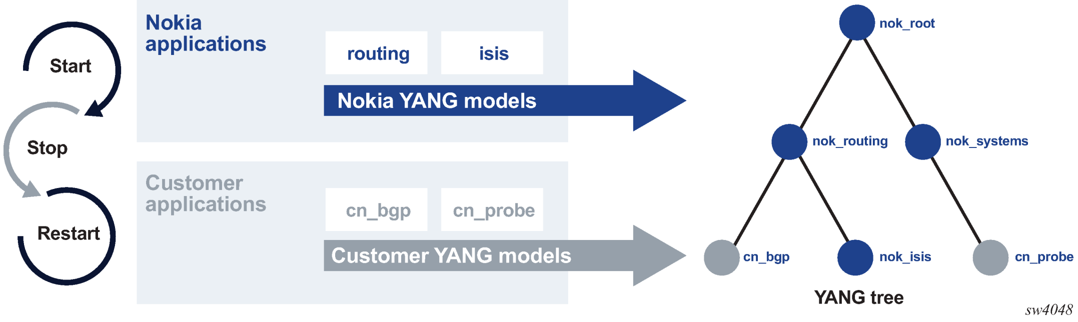

Figure 3: SR Linux software extensibility

2.3 SR Linux NDK

SR Linux provides the NetOps Development Kit (NDK), a software development kit with a suite of libraries to assist operators with developing alongside SR Linux applications. The NDK is provided in the form of header files written in C++. This allows the operator to add SR Linux functionality to their own applications, by using these header files and the methods they provide to interact directly with the SR Linux the IDB server.

See the *SR Linux NDK API Guide* for reference information about the gPRC APIs used with the NDK.

2.4 SR Linux documentation

The SR Linux documentation set consists of the documents listed in Table 1: SR Linux documentation set . These documents are available in PDF and HTML formats.

Table 1: SR Linux documentation set

Document	Description
SR Linux Product Overview	High-level description of SR Linux functionality, including key components, and where it fits into the network.
SR Linux 7250 IXR-6 and 7250 IXR-10 Hardware Installation Guide *SR Linux 7220 IXR-D Chassis Installation Guide* *SR Linux 7220 IXR-H Chassis Installation Guide* *SR Linux 7220 IXR-DL Chassis Installation Guide*	SR Linux supports the following hardware platforms. • 7250 IXR-6 chassis and the 7250 IXR-10 chassis • 7220 IXR-D chassis • 7220 IXR-H chassis • 7220 IXR-DL chassis Each document describes site preparation, chassis and component installation procedures, and hardware component configuration procedures.
SR Linux Software Installation Guide	Basic concepts behind Linux kernel operation on SR Linux, and provides procedures for upgrading the software and provisioning SR Linux using Zero Touch Provisioning (ZTP).
SR Linux Configuration Basics Guide	Basic configuration concepts for the SR Linux, including accessing and using the CLI, and how to manage the system. Descriptions and examples of how to configure key features are provided.
SR Linux EVPN-VXLAN Guide	Basic configuration concepts for EVPN Layer 2 (L2) and Layer 3 ((L3) functionality. Provides examples to configure and implement various protocols and services.
SR Linux Quality of Service Guide	Basic configuration concepts for Quality of Service (QoS) functionality. Provides examples to configure QoS features and classifier policies.
SR Linux Segment Routing Guide	Basic configuration concepts for segment routing functionality. Provides examples to configure segment routing and use of tools that provide operational information about segment routing.
SR Linux MPLS Guide	Basic configuration concepts for the Multiprotocol Label Switching (MPLS) protocol. Provides examples to configure MPLS and LDP.

Document	Description
SR Linux Data Model Reference	Descriptions of the configuration and state data models available for the SR Linux.
SR Linux System Management Guide	Descriptions of the interfaces used with the SR Linux, which include the CLI, gNMI, and JSON. This document also provides an overview to CLI plug-ins and describes Nokia-defined general, operation, and show commands.
SR Linux CLI Quick Reference	Commonly used CLI commands for displaying information, viewing logs, and troubleshooting.
SR Linux Troubleshooting Toolkit	How to use and configure diagnostic tools for SR Linux, including sFlow, interactive traffic monitoring, and packet tracing.
SR Linux CLI Plug-in Guide	How to create custom show routines for SR Linux, as well as how to install, modify, and remove them.
SR Linux NDK API Guide	Reference information for gPRC APIs used with the SR Linux NDK. The NDK provides a way to program high-performance, integrated agents to run alongside SR Linux.
SR Linux Log Event Guide	Contents of the log messages generated by the SR Linux.
SR Linux Advanced Solutions Guide	Scenarios for configuring complex network-level configurations where additional guidance and more detailed procedures may be required.
SR Linux Release Notes	The most up-to-date information about supported features, supported hardware, known limitations, and resolved issues.

3 Hardware overview

The SR Linux software supports nine hardware platforms:

- 7250 IXR-6
- 7250 IXR-10
- 7220 IXR-D1
- 7220 IXR-D2
- 7220 IXR-D3
- 7220 IXR-D2L
- 7220 IXR-D3L
- 7220 IXR-H2
- 7220 IXR-H3

The following figures show these platforms:

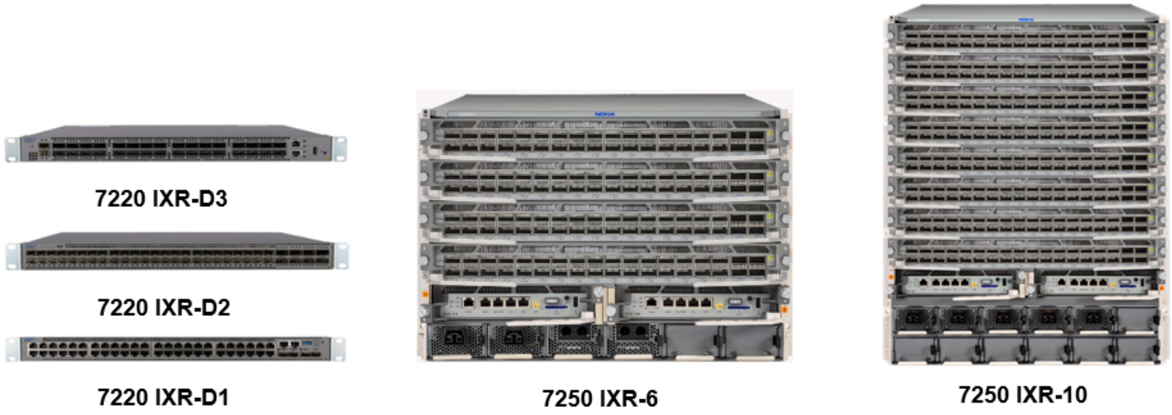

Figure 4: SR Linux 7250 IXR-6, 7250 IXR-10, and 7220 IXR-D platforms

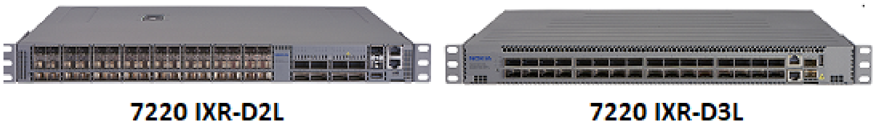

Figure 5: SR Linux 7220 IXR-DL platforms

7220 IXR-H2

7220 IXR-H3

Figure 6: SR Linux 7220 IXR-H platforms

3.1 7250 IXR series

The SR Linux is supported on the 7250 IXR-6 and 7250 IXR-10 hardware platforms. The following table summarizes the specifications of both products.

Table 2: 7250 IXR-6 and 7250 IXR-10 specifications

Parameter	7250 IXR-6	7250 IXR-10
Height	7 RU	13 RU
Depth	81.28 cm	81.28 cm
Number of IMM slots	4	8
Slot capacity	9.6T full duplex (FD)	9.6T FD
Maximum system capacity per chassis	76.1T half duplex (HD) or 115.2T HD with local switching	153.6T HD or 230.4T HD with local switching

3.1.1 Architecture

The 7250 IXR-6 and 7250 IXR-10 platforms deliver capabilities that include IP routing, Layer 2 Ethernet, QoS, router security, scalable telemetry and model-driven programmability. Flexible traffic management includes big buffering, and per-port queuing, shaping and policing.

Each chassis uses an orthogonal direct cross-connect architecture, with Integrated Media Modules (IMMs) connecting in front and switch fabrics and fans connecting at the rear. The lack of a backplane, midplane, or midplane connector system provides a compact chassis design, optimal cooling, and easy capacity upgrades.

3.1.2 Chassis components

The 7250 IXR-6 and 7250 IXR-10 chassis include fan trays, Power Supply Units (PSUs), Control Processing Modules (CPMs), Switch Fabric Modules (SFMs), and IMMs. The two chassis vary in system capacity, height, and number of IMM slots. Table 2: 7250 IXR-6 and 7250 IXR-10 specifications summarizes the differences between the 7250 IXR-6 and 7250 IXR-10.

The system uses a complete Faraday Cage design to ensure EMI containment, a critical requirement for platform evolution that will support next-generation Application-Specific Integrated Circuits (ASICs). The routers are high-density, high-performance modular devices that are designed for data spine deployments. They provide hardware support for 400GE, 100GE, 40GE, 25GE, and 10GE interfaces for intra-fabric and server connectivity.

3.1.3 Power and cooling

The 7250 IXR-6 and 7250 IXR-10 can be AC or DC powered. The DC supply has hot-swappable and load-sharing PSUs. The PSUs are N+M power redundant and each PSU is cooled by a fan independent of the chassis fans. The IXR-6 has a maximum of 12 PSUs and the IXR-10 has a maximum of 12 PSUs.

The chassis have dual fans that support front-to-back airflow. The fan design uses a stainless-steel orthogonal mesh honeycomb placed in front of the line card for air intake. The perforation rate in the honeycomb is approximately 90%, which allows a large surface area that is exposed to cool air causing a reduction in power consumption.

3.2 7220 IXR-D series

SR Linux is supported by the 7220 IXR-D1, 7220 IXR-D2 and 7220 IXR-D3 platforms. The following table summarizes the features of these routers.

Table 3: 7220 IXR-D1, 7220 IXR-D2, and 7220 IXR-D3 specifications

Parameter	7220 IXR-D1	7220 IXR-D2	7220 IXR-D3
Height	1 RU	1 RU	1 RU
Depth	41 cm	46 cm	46 cm
System Capacity (FD)	88G	2T	3.2T

3.2.1 Architecture

The 7220 IXR-D series routers are high-performance, high-density, fixed configuration devices designed for data center leaf-spine deployments. The 7220 IXR-D series platforms deliver capabilities including IP routing, Layer 2 switching, QoS, router security, scalable telemetry, and model-driven management.

3.2.2 Chassis components

The 7220 IXR-D chassis vary in system capacity and height. Table 3: 7220 IXR-D1, 7220 IXR-D2, and 7220 IXR-D3 specifications compares the differences between 7220 IXR-D1, 7220 IXR-D2, and 7220 IXR-D3. They provide high-density QSFP28, QSFP+, SFP28, SFP+, SFP, and RJ-45 ports. The router supports native 100GE, 50GE, 40GE, 25GE, 10GE, and 1GE port options.

3.2.3 Power and cooling

The 7220 IXR-D series chassis can be powered by either dual removable AC PSUs or DC PSUs. The chassis support two PSUs with 1+1 redundancy.

The chassis are equipped with N+1 hot-swappable fans with front-to-back or back-to-front airflow for cooling. The IXR-D1 has three fan modules, the IXR-D2 has four fan modules, and the IXR-D3 has five fan modules.

3.3 7220 IXR-DL series

SR Linux is supported by the 7220 IXR-D2L and 7220 IXR-D3L platforms. The following table summarizes the features of these routers.

Table 4: 7220 IXR-D2L and 7220 IXR-D3L specifications

Parameter	7220 IXR-D2L	7220 IXR-D3L
Height	1 RU	1RU
Depth	53.6 cm	51.5 cm
System Capacity (FD)	2T	3.2T

3.3.1 Architecture

The 7220 IXR-DL series routers are high-performance, high-density, fixed configuration devices designed for data center leaf-spine deployments. The 7220 IXR-DL series platforms deliver capabilities including IP routing, Layer 2 switching, QoS, router security, scalable telemetry, and model-driven management.

3.3.2 Chassis components

The 7220 IXR-DL series chassis vary in system capacity. Table 4: 7220 IXR-D2L and 7220 IXR-D3L specifications compares the differences between the 7220 IXR-D2L and 7220 IXR-D3L. They provide high-density QSFP28, SFP28, SFP+, and RJ-45 ports. The router supports native 100GE, 40GE, 25GE, 10GE, and 1GE port options.

3.3.3 Power and cooling

The 7220 IXR-DL series chassis can be powered by either dual removable AC PSUs or DC PSUs. The chassis support two PSUs with 1+1 redundancy.

The 7220 IXR-DL series chassis are equipped with N+1 hot-swappable fans with front-to-back or back-to-front airflow for cooling. The chassis each have six fan modules.

3.4 7220 IXR-H series

SR Linux is supported by the 7220 IXR-H2 and 7220 IXR-H3 platforms. The following table summarizes the features of these routers.

Table 5: 7220 IXR-H2 and 7220 IXR-H3 specifications

Parameter	7220 IXR-H2	7220 IXR-H3
Height	4 RU	1 RU
Depth	55 cm	55 cm
System Capacity (FD)	12.8T	12.8T

3.4.1 Architecture

The 7220 IXR-H series routers are high-performance, high-density, fixed configuration devices designed for data center leaf-spine deployments. The 7220 IXR-H series platforms deliver capabilities including IP routing, Layer 2 switching, QoS, router security, scalable telemetry, and model-driven management.

3.4.2 Chassis components

The 7220 IXR-H chassis vary in system height. Table 5: 7220 IXR-H2 and 7220 IXR-H3 specifications compares the differences between 7220 IXR-H2 and 7220 IXR-H3. They provide high-density QSFPDD, QSFP28, SFP+, and RJ-45 ports. The router supports native 400GE, 200GE, 100GE, 50GE, 40GE, 25GE, and 10GE port options.

3.4.3 Power and cooling

The 7220 IXR-H series chassis can be powered by either removable AC PSUs or DC PSUs. The 7220 IXR-H2 supports four PSUs with 2+2 redundancy and the 7220 IXR-H3 supports two PSUs with 1+1 redundancy.

The chassis are equipped with N+1 hot-swappable fans with front-to-back or back-to-front airflow for cooling. The 7220 IXR-H2 has eight fan modules and the 7220 IXR-H3 has six fan modules.

4 SR Linux architecture overview

The sections that follow describe components of the SR Linux architecture and how they work together.

4.1 SR Linux components

At a high level, SR Linux can be divided into three components: a mainline Linux kernel, an operating system, and a suite of modular, lightweight applications, each supporting a different protocol or function (IS-IS, BGP, AAA, and so on). These applications use gRPC and APIs to communicate with each other and external systems over TCP.

4.1.1 Linux kernel

The kernel is the core of a computer operating system, with complete control over everything in the system. The kernel is loaded and executed by the boot loader, then handles the system startup as well as input/output requests from software, translating them into data-processing instructions for the CPU. The kernel handles memory and peripherals, as well as interactions with the switch ASIC.

SR Linux uses a mainline Linux kernel with no modifications or customization. This allows kernel upgrades to be performed using normal upgrade mechanisms (yum on CentOS/RHEL-based systems, and aptitude on Debian-based systems).

4.1.2 Operating system

SR Linux is designed to be OS agnostic, with support for all enterprise-class Linux operating systems, with current deployments focused on CentOS.

To follow the Linux Filesystem Hierarchy Standard (FHS), SR Linux software is distributed and laid out as a closed-source, third-party application on the OS. The FHS clearly defines where configuration and binaries should be kept; for SR Linux, all application configuration is kept in the `/opt/srlinux` directory.

4.1.3 Modular applications

SR Linux is a suite of applications running like any others would in a Linux environment. The applications communicate with the IDB to process configuration and state. Figure 7: SR Linux applications and IDB shows the relationship between the IDB and the applications that run in SR Linux.

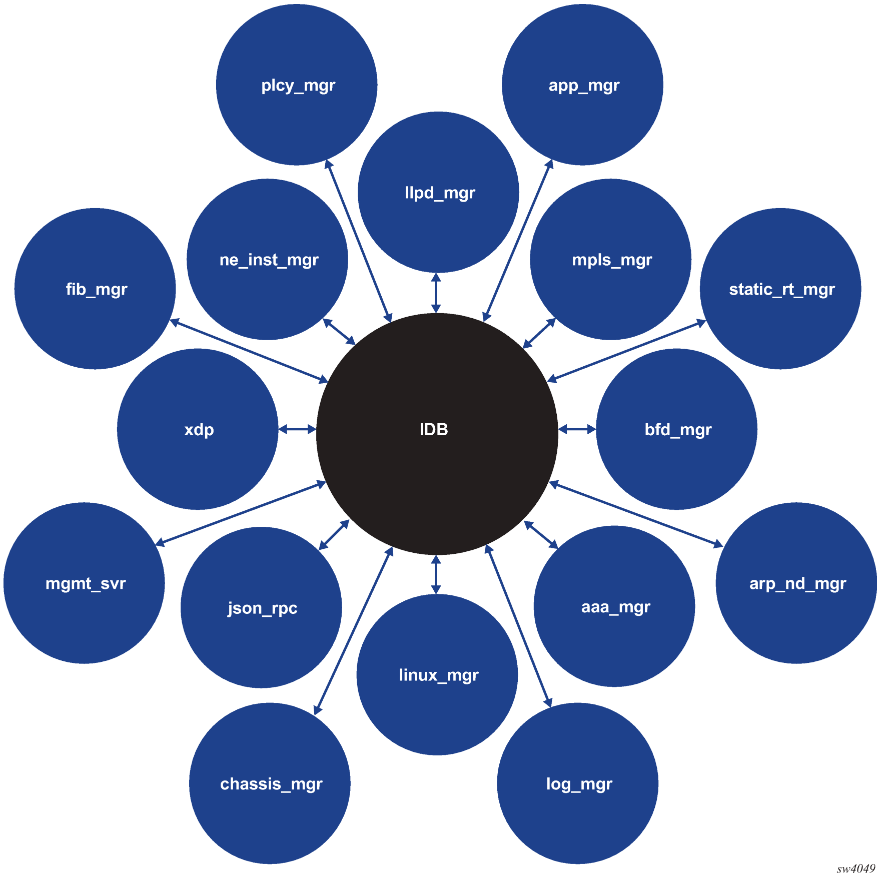

Figure 7: SR Linux applications and IDB

Messaging between applications is controlled by IDB, and configuration via supported APIs is controlled by the management server application.

Table 6: SR Linux applications describes the key SR Linux applications. Use the **show system application** CLI command to display information about all applications running on the system.

Table 6: SR Linux applications

Application name	Description
IDB	Controls messaging between SR Linux applications.
mgmt_server	Controls interaction between external APIs and the IDB, handles application-specific YANG models, and translates them into protobufs for IDB. See SR Linux management server.
aaa_mgr	Performs AAA for end users connecting to the system.
fib_mgr	Responsible for route resolution and route selection.
lldp_mgr	Responsible for sending and receiving LLDP packets via XDP.

Application name	Description
static_route_mgr	Creates/updates/deletes static routes.
arp_nd_mgr	Responsible for ARP resolution for IPv4 and Neighbor Discovery for IPv6.
bgp_mgr	Runs the BGP control plane.
chassis_mgr	Monitors interface/subinterface state and synchronizes interfaces between Linux and the ASIC.
xdp_mgr	Handles data path programming. The xdp_mgr runs on each line card and is split into two components: platform independent and platform dependent. The platform-independent component handles communication between IDB and the ASIC, and the platform-dependent component is an extensible framework for plugging in data plane programming software.
linux_mgr	Creates routes and neighbors in Linux. As control packets enter the Linux host (and kernel), the Linux network stack responds to them. Synchronizing routes and neighbors to Linux stops Linux from ARPing/routing via the default gateway when routes exist at the data plane but not within Linux.
plcy_mgr	Enforces routing policies.
app_mgr	Monitors the health of the processes running SR Linux applications, and restarts them if they fail.
net_inst_mgr	Synchronizes VRFs between the switch ASIC and Linux.
log_mgr	Controls the log infrastructure, implemented by rsyslog.
acl_mgr	Controls ACLs in the system, both on the Linux host and on the CPM.
bfd_mgr	Controls BFD sessions on the system.

IDB stores data as protobufs. Protobufs use a `.proto` file to define how data is structured. Protobufs are not human-readable, but an IDB client is provided with the IDB server, which allows you to read from different topics, decoding the protobuf in real time.

4.1.3.1 Application manager

The application manager is responsible for monitoring the health of the processes running each SR Linux application, and restarting them if they fail.

Each application has specific YAML configuration. The application manager reads in the application-specific YAML configuration and starts each application. It allows applications to not start if no configuration exists for them.

The application manager functions as a replacement for the Linux systemd. At boot time, systemd starts the application manager, which in turn starts SR Linux applications that it manages (based on YAML configuration). The device_mgr is the first application started, then IDB, then other applications are started based on the YAML configuration. The application manager loads the YANG model for each application into the management server.

The application manager is also responsible for restarting the entire system if a critical application cannot be restarted successfully; this restart mechanism is controlled through configuration.

5 SR Linux management overview

This chapter describes the system management functions of SR Linux, including the role of the SR Linux management server and external management APIs (CLI, gNMI, and JSON-RPC). It describes SR Linux configuration modes, methods for securing access to the device, and logging functions.

5.1 SR Linux management server

Configuration and state for the modular SR Linux applications are driven by centralized data models (YANG) that are managed by the SR Linux management server application.

The SR Linux management server provides a central point for external clients and APIs to access the system. The supported external APIs (CLI, JSON-RPC, and gNMI) communicate with the management server via its gRPC interface.

The management server manages the YANG models, which are loaded into the management server by the application manager, based on the requirements of each application. The management server translates these models into protobufs for the IDB. This allows other applications to read their own configuration, by subscribing to the management server topic representing the configuration.

The management server owns the configuration of each application, so each application does not have write access to its own configuration.This provides a central point of configuration enforcement.

Agents built with the NDK function similar to other applications provided with SR Linux. SR Linux applications share state details with each other using a publish/subscribe (pub/sub) architecture. Agents have their own table space within the IDB and can subscribe and receive a notification to events occurring on the device, or create their own table space and publish data to it. This data can be read by other applications within SR Linux, allowing route modifications by publishing routes to the IDB for selection by the FIB manager.

5.2 SR Linux CLI

The SR Linux CLI is an interactive interface for configuring, monitoring, and maintaining the SR Linux via an SSH or console session. The SR Linux CLI operates as a client that communicates with the management server via gRPC. The command tree in the CLI is derived from the SR Linux YANG models.

The SR Linux CLI supports command autocompletion, aliases, annotation, and standard Linux output modifiers such as `grep`. Command output can be displayed in JSON format.

See the "CLI interface" chapter of the *SR Linux System Management Guide* for information about CLI features.

5.3 JSON-RPC server

A JSON-RPC server can be enabled on the SR Linux device, which allows JSON-formatted requests to be issued to the device to retrieve and set configuration and state. You can use the JSON-RPC API to run CLI commands and standard Get and Set methods. The SR Linux device returns responses in JSON-format.

When the JSON-RPC server is enabled, the application passes the requests to the SR Linux management server via the gRPC interface.This JSON-RPC API uses HTTP and HTTPS for transport, and users are authenticated with the aaa_mgr application. HTTPS requests can be authenticated using TLS.

See the "JSON interface" chapter of the *SR Linux System Management Guide* for more information.

5.4 gNMI server

The gRPC-based gNMI protocol is used for the modification and retrieval of configuration from a target device, as well as the control and generation of telemetry streams from a target device to a data collection system.

SR Linux can enable a gNMI server that allows external gNMI clients to connect to the device and modify the configuration and collect state information.

When the gNMI server is enabled, the SR Linux gnmi_mgr application functions as a target for gNMI clients. The gnmi_mgr application validates gNMI clients and passes Get, Set, and Subscribe RPCs to the SR Linux mgmt_svr application via the gRPC interface.

See the "gNMI interface" chapter in the *SR Linux System Management Guide* for information about the supported RPCs.

5.5 Zero Touch Provisioning

Zero Touch Provisioning (ZTP) automates the process of booting the SR Linux device, obtaining an address on the network, then downloading and executing a Python script to configure the system.

The system ships with a operating system image and a boot file (`grub.conf`) installed on the SR Linux compact flash. When the system boots, if `autoboot = enabled` is set in the `grub.conf` file, it initiates the ZTP process. The system then obtains an IP address via DHCPv4 or v6, downloads a Python script to configure the device, and executes the script.

The script can contain URLs to an updated image and a new kernel. The ZTP process can download these and place them on the compact flash, where they become active at the next reboot.

See the "Zero Touch Provisioning" chapter of the *SR Linux Software Installation Guide* for information about using ZTP to initialize the SR Linux.

5.6 SR Linux configuration

SR Linux uses transaction-based configuration, which allows the operator to make changes to the configuration, and then explicitly commit the configuration to apply it.

By default, the SR Linux configuration file is located in `/etc/opt/srlinux/config.json`. At boot time, the management server loads the configuration and publishes content to IDB for applications to consume.

Configuration modes define how the system is running when transactions are performed. Supported modes are the following:

- **Candidate** – is used to modify a configuration. Modifications are not applied to the running system until a **commit** command is issued. When committed, the changes are copied to the running configuration and become active.

- **Running** – is used to display the currently running or active configuration. Configurations cannot be edited in this mode.

When a configuration is committed, it is first validated for YANG syntax, then tested by each application, then validated by the forwarding plane. If validation fails at any stage, the configuration is not committed.

A configuration candidate can be either shared or private:

- **Shared** – is the default configuration candidate for CLI sessions. Multiple users can modify the shared candidate concurrently. When the configuration is committed, the changes from all of the users are applied.
- **Private** – is the default configuration candidate when using JSON-RPC or gNMI clients, and can optionally be used in the CLI. With a private candidate, each user modifies their own separate instance of the configuration candidate. When a user commits their changes, only the changes from that user are committed.

By default, there is a single unnamed global configuration candidate. You can optionally configure one or more named configuration candidates, which function identically to the global configuration candidate. Both shared and private configuration candidates support named versions.

You can optionally create a rescue configuration, which is loaded if the startup configuration fails to load. If the startup configuration fails to load, and no rescue configuration exists, the system is started using the factory default configuration.

When you upgrade the SR Linux software image, the configuration in the startup `config.json` file is read into the running configuration and automatically upgraded to ensure compatibility with the new software version.

See the "Configuration management" chapter in the *SR Linux Configuration Basics Guide* for more information.

5.7 Securing access

The SR Linux is able to secure access to the device for users connecting via SSH or the console port, as well as for applications and FTP access. The SR Linux performs authentication, authorization, and accounting (AAA) functions for each user type.

Authentication can be performed for users configured within the underlying Linux OS, and for administrative users configured within the SR Linux.

Authorization is performed through role-based access control. Users can be configured with a set of one or more roles that indicate the privileges for which they are authorized in the system. You can configure the SR Linux to use information from a TACACS+ server to assign roles to an authenticated user.

The SR Linux supports command accounting, including the entire CLI string that a user enters on the command line, including any pipes or output redirects specified in the command. The accounting records can be sent to a destination such as a TACACS+ server group or the local system.

See the "Securing access" chapter of the *SR Linux Configuration Basics Guide* for more information.

5.8 SR Linux logging

SR Linux implements logging via the standard Linux syslog libraries. The SR Linux device uses rsyslog in the underlying Linux OS to filter logs and pass them on to remote servers or other specified destinations.

The SR Linux supports configuration of Linux facilities and SR Linux subsystems as sources for log messages to filter. See the “Logging” chapter of the *SR Linux Configuration Basics Guide* for information about configuring input sources, filters, and output destinations for log messages.

See the *SR Linux Log Events Guide* for properties and descriptions of the log messages that can be generated by SR Linux subsystems.

6 SR Linux interfaces

This chapter describes SR Linux interface types, subinterfaces, and support for Link Aggregation Groups (LAGs). See the "Interfaces" chapter in the *SR Linux Configuration Basics Guide* for configuration examples.

6.1 SR Linux interface types

On the SR Linux, an interface is any physical or logical port through which packets can be sent to or received from other devices.

The SR Linux supports the following interface types:

- Loopback

  Loopback interfaces are virtual interfaces that are always up, providing a stable source or destination from which packets can always be originated or received. The SR Linux supports up to 256 loopback interfaces system-wide, across all network instances. Loopback interfaces are named **lo***N*, where *N* is 0 to 255.

- System

  The system interface is a type of loopback interface that has characteristics that do not apply to regular loopback interfaces:

  - The system interface can be bound to the default network-instance only.
  - The system interface does not support multiple IPv4 addresses or multiple IPv6 addresses.
  - The system interface cannot be administratively disabled. When configured, it is always up.

  The SR Linux supports a single system interface named **system0**. When the system interface is bound to the default network-instance, and an IPv4 address is configured for it, the IPv4 address is the default local address for multi-hop BGP sessions to IPv4 neighbors established by the default network-instance, and it is the default IPv4 source address for IPv4 VXLAN tunnels established by the default network-instance. The same functionality applies with respect to IPv6 addresses / IPv6 BGP neighbors / IPv6 VXLAN tunnels.

- Network

  Network interfaces carry transit traffic, as well as originate and terminate control plane traffic and in-band management traffic.

  The physical ports in line cards installed in the SR Linux are network interfaces. A typical line card has a number of front-panel cages, each accepting a pluggable transceiver. Each transceiver may support a single channel or multiple channels, supporting one Ethernet port or multiple Ethernet ports, depending on the transceiver type and its breakout options.

In the SR Linux CLI, each network interface has a name that indicates its type and its location in the chassis. The location is specified with a combination of slot number and port number, using the following formats:

ethernet-*slot/port*

For example, interface **ethernet-2/1** refers to the line card in slot 2 of the SR Linux chassis, and port 1 on that line card.

On 7220 IXR-D3 systems, the QSFP28 connector ports (ports 1/3-1/33) can operate in breakout mode. Each QSFP28 connector port operating in breakout mode can have four breakout ports configured, each operating at 25G. Breakout ports are named using the following format:

`ethernet-`*`slot/port/breakout-port`*

For example, if interface **ethernet 1/3** is enabled for breakout mode, its breakout ports are named as follows:

- **ethernet 1/3/1**
- **ethernet 1/3/2**
- **ethernet 1/3/3**
- **ethernet 1/3/4**

- Management

  Management interfaces are used for out-of-band management traffic. The SR Linux supports a single management interface named **mgmt0**.

  The **mgmt0** interface supports the same functionality and defaults as a network interface, except for the following:

  - Packets sent and received on the **mgmt0** interface are processed completely in software.
  - The **mgmt0** interface does not support multiple output queues, so there is no output traffic differentiation based on forwarding class.
  - The **mgmt0** interface does not support pluggable optics. It is a fixed 10/100/1000-BaseT copper port.

- Integrated Routing and Bridging (IRB)

  IRB interfaces enable inter-subnet forwarding. Network instances of type **mac-vrf** are associated with a network instance of type **ip-vrf** via an IRB interface.

6.2 LAG interfaces

A LAG, based on the IEEE 802.1ax standard (formerly 802.3ad), increases the bandwidth available between two network devices, depending on the number of links installed. A LAG also provides redundancy if one or more links participating in the LAG fail. All physical links in a LAG combine to form one logical interface.

LAGs can be either statically configured, or formed dynamically with Link Aggregation Control Protocol (LACP). Load sharing is executed in hardware, which provides line rate forwarding for all port types. A LAG consists of ports of the same speed.

6.3 Subinterfaces

On the SR Linux, each type of interface can be subdivided into one or more subinterfaces. A subinterface is a logical channel within its parent interface.

Traffic belonging to one subinterface can be distinguished from traffic belonging to other subinterfaces of the same port using encapsulation methods such as 802.1Q VLAN tags.

While each port can be considered a shared resource of the router that is usable by all network instances, a subinterface can only be associated with one network instance at a time. To move a subinterface from one network instance to another, you must disassociate it from the first network instance before associating it with the second network instance.

You can configure ACL policies to filter IPv4 and IPv6 packets entering or leaving a subinterface.

The SR Linux supports policies for assigning traffic on a subinterface to forwarding classes or remarking traffic at egress before it leaves the router. DSCP classifier policies map incoming packets to the appropriate forwarding classes, and DSCP rewrite-rule policies mark outgoing packets with an appropriate DSCP value based on the forwarding class.

6.4 DHCP relay

DHCP relay refers to the router's ability to act as an intermediary between DHCP clients requesting configuration parameters, such as a network address, and DHCP servers when the DHCP clients and DHCP servers are not attached to the same broadcast domain, or do not share the same IPv6 link (in the case of DHCPv6).

SR Linux supports DHCP relay for IRB subinterfaces and Layer 3 subinterfaces. The DHCP server network can be in the same IP-VRF network-instance of the Layer 3 subinterfaces that require DHCP relay, or it can be in a different IP-VRF network-instance or the default network instance.

7 SR Linux routing functions

This chapter describes key elements of SR Linux routing functions: network instances, support for standard routing protocols including BGP, OSPF, and IS-IS, as well as ACLs, routing policies. and Quality of Service.

7.1 Network instances

On the SR Linux, you can configure one or more virtual routing instances, known as network instances. Each network instance has its own interfaces, its own protocol instances, its own route table, and its own FIB.

When a packet arrives on a subinterface associated with a network instance, it is forwarded according to the FIB of that network instance. Transit packets are normally forwarded out another subinterface of the network instance.

SR Linux supports the following types of network instances:

- default
- ip-vrf
- mac-vrf

The initial startup configuration for SR Linux has a single default network instance. By default, there are no ip-vrf or mac-vrf network instances; these must be created by explicit configuration. The ip-vrf network instances are the building blocks of Layer 3 IP VPN services, and mac-vrf network instances are the building blocks of EVPN services.

Within a network instance, you can configure BGP, OSPF, and IS-IS protocol options that apply only to that network instance. See the *SR Linux Configuration Basics Guide* for configuration information.

7.2 Static routes

Within a network instance, you can configure static routes. Each static route is associated with an IPv4 prefix or an IPv6 prefix, which represents the packet destinations matched by the static route. Each static route belongs to a specific network instance. Different network instances can have overlapping routes (static or otherwise) because each network instance installs its own routes into its own set of route tables and FIBs.

Each static route must be associated with a statically configured next-hop group, which determines how matching packets are handled: either perform a blackhole discard action or a forwarding action. The next-hop group can specify a list of one or more next-hops, each identified by an IPv4 or IPv6 address and a resolve flag. If the resolve flag is set to false, only a direct route can be used to resolve the IPv4 or IPv6 next-hop address; if the resolve flag is set to true, any route in the FIB can be used to resolve the IPv4 or IPv6 next-hop address.

Each static route has a specified metric and preference. The metric is the IGP cost to reach the destination. The preference specifies the relative degree this static route is preferred compared to other static and non-static routes available for the same IP prefix in the same network instance.

A static route is installed in the FIB for the network instance if the following conditions are met:

- The route has the lowest preference value among all routes (static and non-static) for the IP prefix.
- The route has the lowest metric value among all static routes for the IP prefix.

If BGP is running in a network instance, all static routes of that same network instance are automatically imported into the BGP local RIB, so that they can be redistributed as BGP routes, subject to BGP export policies.

See the “Network instances” chapter of the *SR Linux Configuration Basics Guide* for configuration information.

7.3 Aggregate routes

You can specify aggregate routes for a network instance. Each aggregate route is associated with an IPv4 prefix or an IPv6 prefix, which represents the packet destinations matched by the aggregate route. As with static routes, each aggregate route belongs to a specific network instance, though different network instances can have overlapping routes because each network instance installs its own routes into its own set of route tables and FIBs.

An aggregate route can become active when it has one or more contributing routes. A route contributes to an aggregate route if all of the following conditions are met:

- The prefix length of the contributing route is greater than the prefix length of the aggregate route.
- The prefix bits of the contributing route match the prefix bits of the aggregate route up to the prefix length of the aggregate route.
- There is no other aggregate route that has a longer prefix length that meets the previous two conditions.
- The contributing route is actively used for forwarding and is not an aggregate route itself.

That is, a route can only contribute to a single aggregate route, and that aggregate route cannot recursively contribute to a less-specific aggregate route.

Aggregate routes have a fixed preference value of 130. If there is no route to the aggregate route prefix with a numerically lower preference value, then the aggregate route, when activated by a contributing route, is installed into the FIB with a blackhole next-hop. It is not possible to install an aggregate route into the route table or as a BGP route without also installing it in the FIB.

The aggregate routes are commonly advertised by BGP or another routing protocol so that the individual contributing routes no longer need to be advertised. This can speed up routing convergence and reduce RIB and FIB sizes throughout the network. If BGP is running in a network instance, all active aggregate routes of that network instance are automatically imported into the BGP local RIB so they can be redistributed as BGP routes, subject to BGP export policies.

See the “Network instances” chapter of the *SR Linux Configuration Basics Guide* for configuration information.

7.4 BGP feature support

As with other functions on SR Linux, BGP operates as a modular application. The BGP manager application is responsible for running the BGP control plane. It subscribes to the IDB for configuration updates and listens for network instance and routing policy updates.

SR Linux supports the following BGP features:

Basic features

- Global AS configuration with local AS override per session. SR Linux always advertises 4-byte ASN capability.
- Local-address/source selection per neighbor
- EBGP and IBGP sessions
- Peer groups
- Configurable route table preference, with separate control for EBGP and IBGP routes
- Configurable TCP MSS per-neighbor
- EBGP split-horizon (always enabled)
- EBGP multihop
- BGP import and export policies
- IPv4/IPv6 default originate independent from default routes in the FIB
- AS path loop detection, with configurable threshold
- RFC 7606 error handling
- Tools commands to hard reset peer or soft reset with ROUTE_REFRESH
- Configurable trace options
- Routing policy actions to replace AS_PATH
- Options for handling the AS_PATH in received BGP routes
- BGP route reflection

Failure detection features

- Session KEEPALIVEs, with configurable hold-time and keepalive
- BGP next-hop tracking
- Fast failover (subinterface down)
- Bidirectional Forwarding Detection (BFD) for single-hop and multi-hop IPv4 and IPv6 sessions

Convergence features

- Configurable min-route-advertisement interval
- Rapid-withdrawal
- Option to wait for FIB install before advertising reachability
- Option to delay route advertisement until the address family has reached converged state (or timeout occurs)

Graceful restart

- Helper/receiving router role
- Restarting router role during warm restart

Dynamic/unconfigured BGP neighbors

- Accept incoming sessions from allowed prefix/AS ranges

Address family support

- IPv4 unicast address family with IPv4 next-hops

- IPv4 unicast address family with IPv6 next-hops: requires MP_REACH_NLRI encoding and RFC 5549 capability advertisement
- IPv6 unicast address family with IPv6 next-hops
- Configurable limits on received routes per session per-AF > log when any threshold is exceeded

Multipath/ECMP

- Each BGP route supports multiple ECMP next-hops
- Two-level ECMP: Level 1 selects BGP path/next-hop, Level 2 selects a next-hop of the route that resolves the BGP next-hop
- Max Level 1 next-hops per route and max Level 2 next-hops per BGP next-hop are configurable per NLRI address family

See the *SR Linux Configuration Basics Guide* for a BGP configuration example. See the *SR Linux Advanced Solutions Guide* for a BGP underlay routing example.

7.5 IS-IS feature support

The SR Linux IS-IS manager application includes support for the following features:

- Level 1, Level 2, and Level 1/2 IS types
- Configurable Network Entity Title (NET) per IS-IS instance
- Support for IPv4/v6 routing
- ECMP support per destination
- IS-IS export policies (redistribution of other types of routes into IS-IS)
- Authentication of CSNP, PSNP, and IIH PDUs with authentication type and key, configurable per instance and per instance level. Authentication type and key for IIH PDUs are also configurable per interface and level.
- Support for authentication keychains
- Purge Originator ID TLV (RFC 6232)
- Options to ignore and suppress the attached bit
- Ability to set the overload bit immediately or to set the bit after each subsequent restart of the IS-IS manager application and leave it on for a configurable duration each time
- Control over the Link-State PDU (LSP) MTU size, with range from 490 bytes to 9490 bytes
- Configuration control over timers related to LSP lifetime, LSP refresh interval, SPF calculation triggers, and LSP generation
- Support for hello padding (strict, loose, and adaptive modes)
- Support for graceful restart, but only acting as a helper of the restarting router
- Level 1 to Level 2 route summarization
- BFD for fast failure detection
- Configurable hello timer/multiple per interface and level
- Support for wide metrics (configurable per level)
- Configurable route preference for each route type: Level 1-internal, Level 1-external, Level 2-internal and Level 2-external.

- Use of route policies to add/remove/replace one IS-IS route tag

See the *SR Linux Configuration Basics Guide* for an IS-IS configuration example.

7.6 OSPF feature support

The SR Linux supports OSPFv2 and OSPFv3. The following features are supported:

- Reference bandwidth
- OSPF areas
- Stub areas
- Not So Stubby Areas (NSSAs)
- Passive interfaces
- Authentication
- Route policies
- Route redistribution to other protocols
- BFD for monitoring OSPF adjacencies
- Overload support and associated options
- Export policies (route redistribution from other protocols into OSPF, including ASBR support)
- Graceful restart

See the *SR Linux Configuration Basics Guide* for an OSPF configuration example.

7.7 Routing policies

The SR Linux supports policy-based routing. Policy-based routing controls the size and content of the routing tables, the routes that are advertised, and the best route to take to reach a destination. SR Linux routing policies allow for detailed control of IP routes learned and advertised by routing protocols such as BGP.

Each routing policy has a sequence of rules (called entries or statements) and a default action. Each statement has a numerical sequence identifier that determines its order relative to other statements in that policy. When a route is analyzed by a policy, it is evaluated by each statement in sequential order.

Each policy statement has zero or more match conditions and a base action (either accept or reject); the statement may also have route-modifying actions. A route matches a statement if it meets all of the specified match conditions.

The first statement that matches the route determines the actions that are applied to the route. If the route is not matched by any statements, the default action of the policy is applied. If there is no default action, a protocol- and context-specific default action is applied.

See the “Routing policies” chapter in the *SR Linux Configuration Basics Guide* for a list of valid match conditions and policy actions, as well as configuration examples.

7.8 ECMP load balancing

Static, BGP, OSPF, and IS-IS routes to IPv4 and IPv6 destinations are programmed into the datapath by their respective applications, with multiple IP ECMP next-hops. SR Linux load-balances packets across these IP ECMP next-hops.

When an IPv4/IPv6 packet is received on a subinterface, and it matches a route with a number of ECMP hops, the next-hop used to forward the packet is determined from a hash calculation based on the packet type (IPv4/IPv6, TCP/UDP).

SR Linux attempts to keep packets in the same flow on the same network path while distributing traffic so that each of the *N* ECMP next-hops carries approximately 1/*N*th of the load.

On some platforms, SR Linux supports resilient hashing, which allows SR Linux to move as few flows as possible when removing or adding members to the ECMP set. Resilient hashing is particularly useful when the ECMP next-hops of an IP route correspond to network appliances or host servers that maintain state for the flows that they service, and moving flows would require state to be rebuilt.

7.9 Access Control Lists

An Access Control List (ACL) is an ordered set of rules that are evaluated on a packet-by-packet basis to determine whether access should be provided to a specific resource. ACLs can be used to drop unauthorized or suspicious packets from entering or leaving a routing device via specified interfaces.

SR Linux supports the following types of ACLs:

- Interface ACLs restrict the traffic allowed to pass through a specific set of subinterfaces. An interface ACL can be applied to the input and/or output traffic of one or more subinterfaces.
- CPM-filter ACLs filter IPv4 or IPv6 traffic that is locally terminating on the router, regardless of the subinterface, port, or line card where it enters.
- Capture-filter ACLs filter all transit and terminating IPv4 or IPv6 traffic that is arriving on any subinterface of the router. Traffic matching a capture-filter ACL can be copied to the control plane for packet capture and analysis.
- System filter ACLs evaluate traffic early in the ingress pipeline, at a stage before tunnel termination occurs and before interface filters are run. For VXLAN traffic, system filters can match and drop unauthorized VXLAN tunnel packets before they are decapsulated, based on information in the outer header.

The rules of each ACL policy are evaluated in sequential order. Filter evaluation stops at the first matching entry where all match conditions are met, at which point the actions specified in the ACL are applied to the packet.

For information about configuring ACLs, see the *SR Linux Configuration Basics Guide*.

7.10 Quality of Service

SR Linux supports QoS policies for assigning traffic to forwarding classes or remarking traffic at egress before it leaves the router. DSCP classifier policies map incoming packets to the appropriate forwarding classes, and DSCP rewrite-rule policies mark outgoing packets with an appropriate DSCP value based on the forwarding class.

Each received packet is classified as belonging to one of eight forwarding classes. The classification depends on the packet type and subinterface configuration.

Egress queues are scheduled directly to the output port. In general, higher-numbered queues are serviced before lower-numbered queues.

Scheduling for each queue can be configured as either Strict Priority (SP) or Weighted Round-Robin (WRR). Queues configured as SP are served (in priority order, from highest forwarding class to lowest) before any of the WRR queues, even if some of the WRR queues carry higher forwarding-class traffic than some of the SP queues. After the SP queues are served, the WRR queues use the remaining bandwidth according to their configured weights.

See the *SR Linux Quality of Service Guide* for information about how transit and router-originated traffic is processed, and for configuration information.

7.11 MPLS feature support

Multiprotocol Label Switching (MPLS) provides the ability to set up connection-oriented paths over a connectionless IP network. MPLS facilitates network traffic flow and provides a mechanism to engineer network traffic patterns independently from routing tables.

Label Distribution Protocol (LDP) is a protocol used to distribute MPLS labels in non-traffic-engineered applications. LDP allows routers to establish label switched paths through a network by mapping network-layer routing information directly to data link layer-switched paths.

SR Linux supports the following MPLS and LDP functionality:

MPLS

- Statically configured MPLS forwarding entries
- Configurable label range
- MPLS label manager that shares the MPLS label space among client applications that require MPLS labels

LDP

- LDPv4 implementation compliant with RFC 5036
- LDP support in the default network-instance only
- Label distribution using DU (downstream unsolicited), ordered control
- Platform label space only
- Configurable label range (dynamic, non-shared label-block)
- Support for overload TLV when label-block has no free entries
- Configurable timers (hello-interval, hello-holdtime, keepalive-interval)
- Ingress LER, transit LSR, and egress LER roles for /32 IPv4 FECs
- Automatic FEC origination of the system0.0 /32 IPv4 address prefix
- /32 IPv4 FEC resolution using IGP routes, with longest prefix match option
- ECMP support with configurable next-hop limit (up to 64)
- Automatic installation of all LDP /32 IPv4 prefix FECs into TTM
- Per-peer configurable FEC limit

- Graceful restart helper capability
- BGP shortcuts for IPv4 traffic
- Protocol debug/trace-options
- LDP-IGP synchronization
- Advertise address mapping message for primary IPv4 address of the adjacent interface only.
- Non-configurable capability advertisement in INITIALIZATION messages only claiming support for:
  - State advertisement control (SAC) with interest in IPv4 prefix FECs only
  - Fault tolerance (Graceful Restart)
  - Nokia overload TLV
  - Unrecognized notification
- Split-horizon support: A label-mapping message is not advertised to a peer if the FEC matches an address sent by that peer in an Address Mapping message.

See the *SR Linux MPLS Guide* for configuration information.

8 SR Linux services

SR Linux services facilitate EVPN-VXLAN deployments in data centers. Ethernet Virtual Private Network (EVPN), along with Virtual eXtensible LAN (VXLAN), is a technology that allows Layer 2 and Layer 3 traffic to be tunneled across an IP network.

The SR Linux EVPN-VXLAN solution supports using Layer 2 Broadcast Domains (BDs) in multi-tenant data centers using EVPN for the control plane and VXLAN as the data plane. It includes the following features:

- EVPN for VXLAN tunnels (Layer 2), extending a BD in overlay multi-tenant DCs
- EVPN for VXLAN tunnels (Layer 3), allowing inter-subnet-forwarding for unicast traffic within the same tenant infrastructure

These features are summarized in the following sections. See the *SR Linux EVPN-VXLAN Guide* for descriptions of supported features and configuration examples.

8.1 Layer 2 services

Layer 2 services refers to the infrastructure implemented on SR Linux to support multiple virtual switches on the same system.

To do this, SR Linux uses a network instance of type **mac-vrf**, which functions as a broadcast domain. Each **mac-vrf** network instance builds a bridge table composed of MAC addresses that can be learned via the data path on network instance interfaces or via static configuration. You can configure the size of the bridge table for each **mac-vrf** network instance, as well as the aging for dynamically learned MAC addresses and other parameters related to the bridge table.

The **mac-vrf** network instance is associated with a network instance of type **default** or **ip-vrf** via an Integrated Routing and Bridging (IRB) interface. IRB interfaces enable inter-subnet forwarding.

Figure 8: MAC-VRF, IRB interface, and IP-VRF shows the relationship between an IRB interface and **mac-vrf**, and **ip-vrf** network instance types.

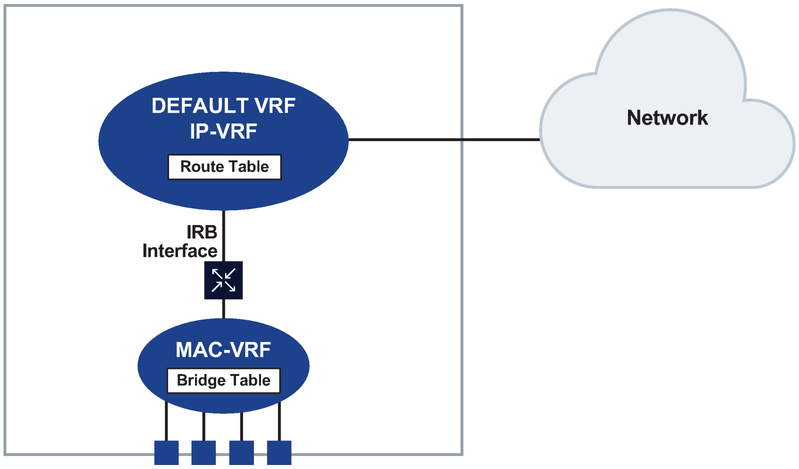

Figure 8: MAC-VRF, IRB interface, and IP-VRF

See the “Layer 2 services infrastructure” chapter of the *SR Linux EVPN-VXLAN Guide* for a description of Layer 2 services components and configuration examples.

8.2 EVPN-VXLAN Layer 2

EVPN for VXLAN tunnels (Layer 2) allows for the extension of a BD in overlay multi-tenant DCs. To support this topology, SR Linux includes the following features:

- Bridged subinterface extensions, including a default subinterface that captures untagged and non-explicitly configured VLAN-tagged frames on tagged subinterfaces
- EVPN-VXLAN control and data plane extensions as described in RFC 8365
- Distributed security and protection
- EVPN L2 multi-homing, including the ES model definition for all-active and single-active multi-homing

See the “EVPN for VXLAN tunnels (Layer 2)” chapter of the *SR Linux EVPN-VXLAN Guide* for a description of supported features, basic configuration information, and EVPN L2 multi-homing configuration examples.

8.3 EVPN-VXLAN Layer 3

SR Linux supports EVPN for VXLAN tunnels (Layer 3) for inter-subnet-forwarding for unicast traffic within the same tenant infrastructure. SR Linux features that support this topology fall into the following categories:

- EVPN-VXLAN L3 control plane (RT5) and data plane as described in draft-ietf-bess-evpn-prefix-advertisement
- EVPN L3 multi-homing on MAC-VRFs with IRB interfaces that use anycast GW IP and MAC addresses in all leafs attached to the same BD
- Host route mobility procedures to allow fast mobility of hosts between leaf nodes attached to the same BD

Other supported features include:

- Interface-less (IFL) model interoperability with unnumbered interface-ful (IFF) model
- ECMP over EVPN
- Support for interface-level OAM (ping) in anycast deployments
- EVPN interoperability with VLAN-aware bundle services

See the "EVPN for VXLAN tunnels (Layer 3)" chapter of the *SR Linux EVPN-VXLAN Guide* for EVPN Layer 3 basic configuration information and examples.

9 SR Linux troubleshooting tools

This chapter describes the troubleshooting and diagnostic tools available on SR Linux, including BFD support, sFlow support, traffic monitoring, and packet-tracing functions.

9.1 BFD support

Bidirectional Forwarding Detection (BFD) is a lightweight mechanism used to monitor the liveliness of a remote neighbor. Because of this lightweight nature, BFD can send and receive messages at a much higher rate than other control plane hello mechanisms. This attribute allows connection failures to be detected faster than other hello mechanisms.

SR Linux supports BFD asynchronous mode, where BFD control packets are sent between two systems to activate and maintain BFD neighbor sessions between them.

BFD can be configured to monitor connectivity for the following:

- BGP peers
- Next-hops for static routes
- OSPF adjacencies
- IS-IS adjacencies

Micro-BFD, where BFD sessions are established for individual members of a LAG, is also supported. If the BFD session for one of the links indicates a connection failure, the link is taken out of service from the perspective of the LAG.

See the “BFD” chapter in the *SR Linux Configuration Basics Guide* for configuration information.

9.2 sFlow support

The SR Linux supports sFlow version 5 behavior and formats. sFlow is used to monitor data traffic flows traversing different points in a network. The sFlow functionality uses an sFlow agent and an sFlow collector. The agent is software that runs on a network element and samples and reports flow headers and statistics. The collector is software that typically runs on a remote server and receives the flow headers and statistics from one or more sFlow agents.

On the SR Linux, sFlow samples flow data and reports the samples to configured sFlow collectors. Up to eight sFlow collectors can be configured. When sFlow is enabled on an interface, the sFlow agent streams interface statistics to the configured sFlow collectors.

See the “sFlow” chapter of the *SR Linux Troubleshooting Toolkit* for configuration information and examples.

9.3 Interactive traffic monitoring tool

SR Linux features an interactive traffic monitoring tool that samples packets entering the system on any interface matching a set of parameters, and streams the header details either to the current login session or to a specified output file.

When the traffic monitoring tool is activated, mirroring policies are dynamically populated on all ingress ports, and matching packets are sent to the CPM for display. Header information for the matching packets is displayed in either tcpdump format or hex format, depending on the options chosen.

When the traffic monitoring tool is deactivated, the mirroring policies are automatically removed from all ingress interfaces.

See the "Interactive traffic monitoring" chapter of the *SR Linux Troubleshooting Toolkit* guide for usage information.

9.4 Packet-trace tool

SR Linux includes a packet-trace tool that reports the forwarding behavior of a probe packet. The probe packet is injected into a specified interface forwarding context, and the packet-trace tool records the forwarding destination or egress port for the probe packet, as well as any matched ACL records or reasons for discarding the packet. The probe packet can be specified in Scapy format, base64 format, or pcap file format.See the "Packet-trace tool" chapter of the *SR Linux Troubleshooting Toolkit* guide for usage information.

10 Standards compliance and protocol support

This chapter defines the supported standards and protocols for SR Linux.

Standards Compliance

IEEE 802.1ab-REV/D3 Station and Media Access Control Connectivity Discovery

IEEE 802.1p/Q VLAN Tagging

IEEE 802.1AX (formerly 802.3AD) Link Aggregation

IEEE 802.3ae 10Gbps Ethernet

IEEE 802.3ba 100 Gb/s Ethernet

IEEE 802.3u 100BaseTX

IEEE 802.3z 1000BaseSX/LX

Protocol support

BGP

RFC 1772 Application of BGP in the Internet

RFC 1997 BGP Communities Attribute

RFC 2385 Protection of BGP Sessions via MD5

RFC 2918 Route Refresh Capability for BGP-4

RFC 5492 Capabilities Advertisement with BGP-4

RFC 4271 BGP-4

RFC 4360 BGP Extended Communities Attribute

RFC 4760 Multi-protocol Extensions for BGP

RFC 4893 BGP Support for Four-octet AS Number Space

RFC 5668 4-Octet AS Specific BGP Extended Community Capability for BGP-4

RFC 4724 - Graceful Restart Mechanism for BGP

RFC 5549 - Advertising Ipv4 Network Layer Reachability Information with an IPv6 Next Hop

RFC 2545 - Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing

RFC 7705 - Autonomous System Migration Mechanisms and Their Effects on the BGP AS_PATH Attribute

RFC 4456 - BGP Route Reflection - An Alternative to Full Mesh Internal BGP (IBGP)

RFC 4486 - Subcodes for BGP Cease Notification Message

RFC 7606 - Revised Error Handling for BGP

RFC 6286 - Autonomous-System Wide Unique BGP Identifier for BGP-4

RFC 8212 - Default External BGP (EBGP) Route Propagation Behavior without Policies

DHCP relay

RFC2131 Dynamic Host Configuration Protocol

RFC2132 DHCP Options and BOOTP Vendor Extensions

RFC1542 Clarifications and Extensions for the Bootstrap Protocol

RFC3046 DHCP Relay Agent Information Option

RFC8415 Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

RFC4649 DHCPv6 Relay Agent Remote-ID Option

DHCP server

RFC 2131 - Dynamic Host Configuration Protocol (only static address allocation. No dynamic address)

RFC 2132 (sections 3.1, 3.2, 3.3 , 3.5 , 3.8, 3.14, 3.17, 9.2, 9.6)

RFC 3315 (sections 17, 17.2,17.2.1,17.2.2,17.2.3,18.2, 22.1, 22.2, 22.3 ,22,4, 22,6 22.7,22.13)

EVPN

RFC8365 A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)

RFC9135 Integrated Routing and Bridging in EVPN (except the Symmetric IRB section)

RFC9136 IP Prefix Advertisement in EVPN (Interface-less IP-VRF-to-IP[1]VRF Model)

IPv4

RFC 768 UDP

RFC 791 IP

RFC 792 ICMP

RFC 793 TCP

RFC 826 ARP

RFC 1519 CIDR

RFC 1812 Requirements for IPv4 Routers

RFC 1191 - Path MTU Discovery

RFC 5227 - IPv4 Address Conflict Detection

IPv6

RFC 8200 Internet Protocol, Version 6 (IPv6) Specification

RFC 4861 Neighbor Discovery for IPv6

RFC 4443 Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 Specification

RFC 2464 Transmission of IPv6 Packets over Ethernet Networks

RFC 3587 IPv6 Global Unicast Address Format

RFC 4007 IPv6 Scoped Address Architecture

RFC 4291 IPv6 Addressing Architectur

RFC 6164 Using 127-Bit IPv6 Prefixes on Inter-Router Links

RFC 2131 - Dynamic Host Configuration Protocol

RFC 8201 - Path MTU discovery for IPv6

RFC 8415 - Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

IS-IS

RFC 1142 OSI IS-IS Intra-domain Routing Protocol (ISO 10589)

RFC 1195 Use of OSI IS-IS for routing in TCP/IP and dual environments

RFC 3719 Recommendations for Interoperable Networks using IS-IS

RFC 3787 Recommendations for Interoperable IP Networks

RFC 5301 Dynamic Hostname Exchange for IS-IS

RFC 5302 Domain-wide Prefix Distribution with Two-Level IS-IS

RFC 5303 Three-Way Handshake for Intermediate System to Intermediate System (IS-IS) Point-to-Point Adjacencies

RFC 5304 Intermediate System to Intermediate System (ISIS) Cryptographic Authentication

RFC 5306 Restart Signaling for IS-IS – GR helper

RFC 5120 - M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)

RFC 5130 - A Policy Control Mechanism in IS-IS Using Administrative Tags

RFC 5308 - Routing IPv6 with IS-IS

RFC 6232 - Purge Originator Identification TLV for IS-IS

MPLS

RFC 5036 LDP Specification

RFC 3032 MPLS Label Stack Encoding

RFC 3443 Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks

RFC 4950 ICMP Extensions for Multiprotocol Label Switching

RFC 3270 Multi-Protocol Label Switching (MPLS) Support of Differentiated Services

Network management

RFC 1907 SNMPv2-MIB (partial compliance only)

RFC 2863 IF MIB (partial compliance only)

RFC 3164 Syslog

RFC 3411 An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks

RFC 3412 - Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)

RFC 3413 - Simple Network Management Protocol (SNMP) Applications

RFC 8343 - A YANG Data Model for Interface Management

RFC 8344 - A YANG Data Model for IP Management

OAM

RFC 5880 Bidirectional Forwarding Detection

RFC 5881 BFD IPv4 (Single Hop)

RFC 5883 BFD for Multihop Paths

RFC 7130 BFD on Link Aggregation Group (LAG) Interfaces

OSPF

RFC 1765 OSPF Database Overflow

RFC 2328 OSPF Version 2

RFC 2740 OSPF for IPv6 (OSPFv3)

RFC 3101 OSPF NSSA Option

RFC 3137 OSPF Stub Router Advertisement

RFC 3623 Graceful OSPF Restart

Segment routing

RFC 8667 IS-IS Extensions for Segment Routing

SSH

draft-ietf-secsh-architecture.txt SSH Protocol Architecture

draft-ietf-secsh-userauth.txt SSH Authentication Protocol

draft-ietf-secsh-transport.txt SSH Transport Layer Protocol

draft-ietf-secsh-connection.txt SSH Connection Protocol

draft-ietf-secsh-newmodes.txt SSH Transport Layer Encryption Modes

TACACS+

draft-grant-tacacs-02.txt (partial compliance- accounts and authentication only)

Customer document and product support

Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback