Enterprise SONiC Distribution by Dell Technologies

User Guide Release 3.1

July 2020
Rev. A03

DELL EMC

Notes, cautions, and warnings

NOTE: A NOTE indicates important information that helps you make better use of your product.

CAUTION: A CAUTION indicates either potential damage to hardware or loss of data and tells you how to avoid the problem.

WARNING: A WARNING indicates a potential for property damage, personal injury, or death.

Contents

1 Revision history

Table 1. Revision history

Release	Revision	New features
3.0.1	A02 (April 2020)	Updated command syntaxes and examples
3.0.0	A01 (March 2020)	Initial release

Additional documentation

Table 2. Additional documentation

Document	Description
Enterprise SONiC Distribution by Dell Technologies, User Guide Release 3.0	User guide
Enterprise SONiC Distribution by Dell Technologies, Quick Start Guide Release 3.0	Installation and initial configuration
Enterprise SONiC Distribution by Dell Technologies, Management Framework CLI Reference Guide Release 3.0	Management Framework CLI command syntaxes and examples
Enterprise SONiC Distribution by Dell Technologies, Release Notes Release 3.0	New features introduced in the release; known and fixed issues

2

SONiC overview

What is SONiC

SONiC is an open-source, Linux-based network operating system (NOS) that runs on switches from multiple vendors and ASICs. SONiC stands for Software for Open Networking in the Cloud. It implements standard Layer 2 and Layer 3 protocols, and provides developers with a straightforward way to add new features.

SONiC offers teams the flexibility to create data center networking solutions, while using the collective strength of a large ecosystem with an active developer community. SONiC is designed for scalability and is in production today in large data center fabrics. Some of SONiC's benefits include:

- Hardware independence
- Containerized architecture
- Open-sourced
- Access to a growing community

For more information, go to What is SONiC?.

Topics:

SONiC and Dell Technologies

Dell Technologies introduces Enterprise SONiC Distribution as a hardened, validated, and supported version of SONiC for switch configuration and monitoring. It includes distribution of open-source community SONiC, and additional features to support the ecosystem and partners.

Enterprise SONiC supports an intuitive command-line interface, and object-based administration through a REST interface and Google's gRPC Network Management Interface (gNMI).

Enterprise SONiC Distribution by Dell Technologies

Enterprise SONiC is offered in the following bundles. Customers can deploy the most appropriate bundle for their network requirements:

- Cloud standard
- Cloud premium
- Enterprise standard
- Enterprise premium

Dell EMC PowerSwitch platforms

The following ONIE-enabled Dell EMC PowerSwitch platforms are supported:

- 25G: S5248F-ON and S5296F-ON
- 100G: S5232F-ON, Z9100-ON, and Z9264F-ON
- 400G: Z9332F-ON

3

Management Framework

When configuring SONiC, a common challenge is navigating between different shells, and different JSON and XML files to configure the switch. Different configuration commands and syntaxes are used across shells, such as Linux and FRR. As a result, the likelihood of configuration errors increases. Also, it is not possible to back up the switch configuration from a single place, requiring multiple steps for system upgrades.

The Management Framework is introduced to resolve these challenges. The Management Framework is a SONiC application that provides a consistent, holistic way to validate, configure, and manage the features running on the SONiC OS. The Management Framework consists of the following user interfaces:

Management Framework CLI	Next generation, intuitive command-line interface for IT administrators that supports centralized switch operation and management, and augments the existing SONiC CLIs: • Linux shell commands — Configuration commands that you enter in the Linux shell, such as `useradd`, `ip`, and `ifconfig`. • SONiC CLI — The existing legacy CLI that is based on a Python library. • Free Range Routing shell CLI — Configuration commands for routing protocols, such as BGP, OSPF, and EVPN. Free Range Routing is also referred to as *FRR* and *FRRouting*.
Programmatic northbound interfaces	• REST API — See REST API • gRPC network management interface (gNMI) — See gRPC Network Management Interface

The Management Framework supports standard and custom YANG models for communication with management systems. The Management Framework runs in a single container called sonic-mgmt-framework.

Linux shell

When you log in to the SONiC NOS, you are placed in the Linux shell.

```
admin@10.10.10.10's password:
Linux sonic 4.9.0-11-2-amd64 #1 SMP Debian 4.9.189-3+deb9u2 (2019-11-11) x86_64
You are on

 ____   ___   _   _ _  ____
/ ___| / _ \| \ | (_)/ ___|
\___ \| | | |  \| | | |
 ___) | |_| | |\  | | |___
|____/ \___/|_| \_|_|\____|

-- Software for Open Networking in the Cloud --

Unauthorized access and/or use are prohibited.
All access and/or use are subject to monitoring.

Help:    http://azure.github.io/SONiC/

Last login: Tue Mar  3 11:10:28 2020 from 192.168.35.3
admin@sonic:~$
```

To display the contents of a folder:

```
admin@sonic:~$ sudo ls /etc/sonic/
agent_config.cfg       frr                       snmp.yml
asic_config_checksum   generated_services.conf   sonic_branding.yml
config_db.json         hamd                      sonic_version.yml
constants.yml          init_cfg.json             updategraph.conf
```

To display the contents of a file:

```
admin@sonic:~$ sudo cat /etc/sonic/config_db.json
{
   "DEVICE_METADATA": {
      "localhost": {
          "docker_routing_config_mode": "split",
          "hostname": "Leaf1",
          "hwsku": "DellEMC-S5248f-P-25G",
          "mac": "8c:04:ba:a7:ee:c0",
          "platform": "x86_64-dellemc_s5248f_c3538-r0",
          "type": "LeafRouter"
      }
   },
   "FLEX_COUNTER_TABLE": {
      "PFCWD": {
         "FLEX_COUNTER_STATUS": "enable"
      },
   "PORT": {
      "FLEX_COUNTER_STATUS": "enable"
   },
   "QUEUE": {
      "FLEX_COUNTER_STATUS": "enable"
   }
},
"HARDWARE": {
   "ACCESS_LIST": {
      "COUNTER_MODE": "per-rule",
      "LOOKUP_MODE": "optimized"
      }
   },
   "INTERFACE": {
      "Ethernet52": {
         "ipv6_use_link_local_only": "enable"
      }
   },
   "LOOPBACK_INTERFACE": {
      "Loopback0": {},
      "Loopback0|10.0.1.1/32": {}
   },
   "MGMT_INTERFACE": {
      "eth0": {},
      "eth0|100.67.204.35/24": {}
   },
   "MGMT_PORT": {
      "eth0": {
         "admin_status": "up",
         "autoneg": "true",
         "description": "Management0",
         "mtu": "1500",
         "speed": "1000"
      }
   }
   ...
```

SONiC CLI

The SONiC CLI is a set of user-space Linux applications, such as `config` and `show`, that allow you to configure and provision SONiC switches.

The SONiC CLI uses the Python Click library. The Click library provides developers with a customizable approach to create command-line tools, perform various configurations, and leverage the defaults provided with the package. For more information about the Click Python package, go to Welcome to Click.

When you log in, you are placed in the Linux shell. To display the available configurations supported by the SONiC CLI, enter `sudo config ?`. Root privilege is required to run the configuration and show commands.

```
admin@sonic:~$ sudo config ?
Usage: config [OPTIONS] COMMAND [ARGS]...
```

```
  SONiC command line - 'config' command

Options:
  --help  Show this message and exit.

Commands:
  aaa                     AAA command line
  acl                     ACL-related configuration tasks
  bgp                     BGP-related configuration tasks
  classifier              Classifiers related configuration tasks
  copp                    Configure COPP
  core                    Configure coredump
  custom_assert           Configuration action on assert
  ecn                     ECN-related configuration tasks
  export
  flow                    Flow related configuration tasks
  hardware                Configure hardware parameters
  hostname
  igmp_snooping           igmp-snooping configuration tasks
  interface               Interface-related configuration tasks
  ...
```

To get help at the command level, enter `?` in a command syntax; for example:

```
admin@sonic:~$ sudo config ztp ?
Usage: config ztp [OPTIONS] COMMAND [ARGS]...

  Configure Zero Touch Provisioning

Options:
  -?, -h, --help  Show this message and exit.

Commands:
  disable  Administratively Disable ZTP.
  enable   Administratively Enable ZTP.
  run      Restart ZTP of the device.
```

To save SONiC CLI configuration changes, enter the `config save` command:

```
admin@sonic:~$ sudo config save
Existing file will be overwritten, continue? [y/N]: y
Running command: /usr/local/bin/sonic-cfggen -d --print-data > /etc/sonic/config_db.json
admin@sonic:~$
```

Enter `show` commands to display switch configuration; for example:

```
admin@sonic:~$ sudo show ztp
ZTP Admin Mode : False
ZTP Service    : Inactive
ZTP Status     : Not Started

ZTP Service is not running
```

FRR shell CLI

To configure routing, you can also use the FRRouting shell. FRR provides a suite of standard IP routing protocols, including BGP, RIP, OSPF, EIGRP, BABEL, IS-IS, PIM, and VRRP. In FRR, each routing protocol operates independently with a separate daemon to ensure high resiliency in this open-source application. For more information about FRR, go to FRR Overview.

SONiC routing protocols are configured in the FRR shell. To enter the FRR shell from SONiC, use the `vtysh` command.

```
admin@sonic:~$ vtysh

Hello, this is FRRouting (version 7.2-sonic).
COpyright 1996-2005 Kunihiro Ishiguro, et al.

sonic#
```

Enter routing configuration commands at the prompt. For example, using the FRR shell, you can configure eBGP with these commands:

```
sonic# configure
sonic(config)# router bgp 1234
sonic(config-router)# bgp router-id 10.0.2.1
sonic(config-router)# bgp graceful-restart
sonic(config-router)# bgp bestpath as-path multipath-relax
sonic(config-router)# neighbor SPINE peer-group
sonic(config-router)# neighbor SPINE timers 3 9
sonic(config-router)# neighbor SPINE advertisement-interval 5
sonic(config-router)# neighbor 192.168.1.0 peer-group SPINE
sonic(config-router)# neighbor 192.168.1.0 remote-as 65100
sonic(config-router)# neighbor 192.168.2.0 peer-group SPINE
sonic(config-router)# neighbor 192.168.2.0 remote-as 65100
sonic(config-router)# address-family ipv4 unicast
sonic(config-router-af)# exit-address-family
sonic(config-router)# exit
sonic(config)#
```

Management Framework CLI

When you configure and monitor a SONiC switch, to avoid the need of using different management interfaces with different commands — Python-based SONiC CLI, Linux shell, and FRRouting shell — you can use a single, centralized management framework. The SONiC Management Framework CLI includes:

- Holistic, intuitive command-line interface — Described in the chapters in this document.
- REST API — See REST API.
- gNMI — See gRPC Network Management Interface.
- gNOI — See gRPC Network Operations Interface.

In addition to using the Management Framework command-line interface, you can also use a REST API and gNMI that access YANG-modeled data. The Management Framework supports both standards-based data models, such as `OpenConfig`, and SONiC YANG models for switch configuration.

To start the Management Framework CLI, go to Getting Started.

4

Getting Started

After you log in to a switch, you can start using the Management Framework CLI to configure and monitor the SONiC device.

For information about how to use the REST API or gNMI, see REST API and gNMNI Network Management Interface.

Topics:

Log in to SONiC

All SONiC devices support both a serial console-based and SSH-based login. If you use an SSH login, log in to the Management interface (eth0) IP address — see SSH login. To log in from a console:

1. Log in to the Linux shell from the console or through an SSH connection using the default username `admin` and password `YourPaSsWoRd`.

```
At Console:
Debian GNU/Linux 9 sonic ttyS1

sonic login: admin
Password: YourPaSsWoRd

SSH from any remote server to sonic can be done by connecting to the IP address of the
Management interface
user@debug:~$ ssh admin@sonic_ip_address(or SONIC DNS Name)
admin@sonic's password:
```

2. When you log in for the first time, you are prompted to change your password. Follow the screen instructions to change your password and log in to the Linux shell. If this is not your first login, go to Step 3.

```
You are required to change your password immediately (root enforced)
Changing password for admin.
(current) UNIX password: YourPaSsWoRd
Enter new UNIX password: *********
Retype new UNIX password: ********
Linux sonic 4.9.0-11-2-amd64 #1 SMP Debian 4.9.189-3+deb9u2 (2019-11-11) x86_64
You are on

 ____   ___   _   _ _____
/ ___| / _ \| \ | (_)/ ___|
\___ \| | | |  \| | | |
 ___) | |_| | |\  | | |___
|____/ \___/|_| \_|_|\____|

-- Software for Open Networking in the Cloud --

Unauthorized access and/or use are prohibited.
All access and/or use are subject to monitoring.

Help:    http://azure.github.io/SONiC/

admin@sonic:~$
```

3. Enter `sonic-cli` to access the Management Framework CLI in EXEC mode. From this mode, you can run `show` commands to monitor various functions on the switch or `debug` commands to troubleshoot switch operation. Use `?` to display the available commands.

```
admin@sonic:~$ sonic-cli
sonic# ?
  clear      Clear commands
  configure  Enter configuration mode
  copy       Perform file copy operations
  debug      Debug commands
  exit       Exit from the CLI
  image      Image related commands
  no         No commands under exec mode
  show       Show running system information
  system     System command
  write      Save config
  ...
```

4. Access CONFIGURATION mode to configure switch settings. In CONFIGURATION mode, you can change the current running configuration. Configuration changes are not automatically saved by default. You must use the `write memory` command.

```
sonic# config terminal
sonic(config)#
```

To return to the Linux shell:

```
sonic(config)# end
admin@sonic:~$
```

CLI basics

This information describes how to use the Management Framework CLI from the console or through a network connection to configure and monitor a SONiC device. The Management Framework CLI runs on top of a Linux-based operating system kernel.

CLI command modes

The Management Framework CLI has two top-level modes:

- EXEC mode — Monitor, troubleshoot, check status, and network connectivity
- CONFIGURATION mode — Configure network devices

NOTE: When you enter CONFIGURATION mode, you are changing the current running configuration. By default, configuration changes are not automatically saved. To save changes, you must enter the `write memory` command.

CLI command hierarchy

Management Framework CLI commands are organized in a hierarchy, which you step through to configure the switch. To move up one command mode, enter the `exit` command. To move directly to the EXEC mode from any submode, enter the `end` command.

```
sonic# config terminal
sonic(config)# interface Ethernet 21
sonic(conf-if-Ethernet21)# no shutdown
sonic(conf-if-Ethernet21)# exit
sonic(config)# exit
sonic# write memory
sonic#
```

OR

```
sonic# config terminal
sonic(config)# interface Ethernet 21
sonic(conf-if-Ethernet21)# no shutdown
sonic(conf-if-Ethernet21)# end
```

```
sonic# write memory
sonic#
```

Management Framework CLI features

Consistent command names	Commands that start the same type of function have the same name. For example, `show` commands display hardware or software information and statistics; `clear` commands erase various types of system information.
Available commands	Information about available commands is provided at each level of the CLI command hierarchy. To view a list of available commands, along with a short description of each command, enter a question mark (`?`) in the command-line interface.
Command completion	Command completion for command names (keywords) and for command options is available. To complete a command or option that you have partially entered, press the **Tab** key or the **Spacebar**. If the partially entered letters are a string that uniquely identifies a command, the complete command name appears. A beep indicates that you have entered an ambiguous command, and the possible completions display. Completion also applies to other strings, such as interface names and configuration statements.

SSH login

By default, SONiC uses DHCP to obtain an IP address for the Management interface (`eth0`) from a DHCP server — see Using a DHCP server. To set up a remote SSH login to configure the switch:

1. Log in to the switch from the serial console. If you did not already change your password after login, the default credentials are username `admin` and password `YourPaSsWoRd`.
2. Disable the DHCP client, and configure an IP address on the Management interface. Configure a Management route for remote access.

```
sonic# config terminal
sonic(config)# interface Management 0
sonic(conf-if-eth0)# no ip address dhcp
sonic(conf-if-eth0)# ip address 10.1.1.10/24 gwaddr 10.1.1.1
sonic(conf-if-eth0)# no shutdown
sonic(conf-if-eth0)# exit
sonic(config)#
```

3. Log in to the Management interface (`eth0`) IP address in an SSH session. The Management interface must be `UP` and have an IP address.

```
At Console:
Debian GNU/Linux 9 sonic ttyS1

sonic login: admin
Password: YourPaSsWoRd

SSH from any remote server to sonic can be done by connecting to SONiC IP
user@debug:~$ ssh admin@sonic_ip_address(or SONIC DNS Name)
admin@sonic's password:
```

   - When you log in with an `admin` role, you are placed in the Linux shell. The prompt is `admin@sonic:~$`. To access the Management Framework CLI, enter the `sonic-cli` command — see Log in to SONiC.
   - When you log in with an `operator` role, you are placed in the Management Framework CLI. The prompt is `sonic#`. An `operator` user cannot access the Linux shell.

   For more information, see Role-based access control.

Using a DHCP server to automate the initial configuration

When you use a DHCP server to boot SONiC, you can load the startup configuration from the `config_db.json` file that is stored on a remote server. The Management interface uses DHCP by default to obtain the Management interface IP address from a DHCP server.

If you use DHCP, you must first configure the DHCP server to provide the necessary information, such as management interface IP address, default route, configuration file name, and the server IP address from which the configuration file is downloaded. SONiC contacts the remote server, downloads the `config_db.json` file, and loads the configuration in the file.

ZTP also uses a DHCP server to download and install a SONiC image and the startup `config_db.json` file. ZTP supports HTTP, TFTP, and FTP to download files, such as `ztp.json`, `config_db.json`, software images, and scripts. For more information, see ZTP DHCP server configuration.

Default startup configuration

When you install SONiC, if you do not configure switch settings after the switch reboots, the default startup configuration is loaded as the running configuration. See the *Enterprise SONiC Distribution by Dell Technologies, Quick Start Guide* for complete information.

On a SONiC switch, the startup configuration is stored in the `/etc/sonic/config_db.json` file if ZTP is disabled. These keys are configured by default in the `config_db.json` file.

```
1) DEVICE_METADATA
2) MAP_PFC_PRIORITY_TO_QUEUE
3) QUEUE
4) PORT
5) CRM
6) PORT_QOS_MAP
7) NTP_SERVER
8) BUFFER_QUEUE
9) WRED_PROFILE
10) TC_TO_PRIORITY_GROUP_MAP
11) BUFFER_PROFILE
12) DEVICE_NEIGHBOR
13) DSCP_TO_TC_MAP
14) TC_TO_QUEUE_MAP
15) CABLE_LENGTH
16) SCHEDULER
17) BUFFER_POOL
```

Show SONiC version

To display the software version of the running SONiC image, use the `show version` command.

```
sonic# show version
Software Version:'3.0.0-Enterprise_Base'
```

Show system status

To monitor switch hardware operation, you can display information about the system status.

View system information

```
sonic# show system
-------------------------------------------------------------
Attribute            Value/State
-------------------------------------------------------------
Hostname            :sonic
Boot Time           :1563911113
Current Datetime    :2019-07-24 18:34:43+00:00
Domain Name         :None
```

View CPU usage

```
sonic# show system cpu
------------------------------------------------------------------------
CPU                  %KERNEL             %USER               %IDLE
------------------------------------------------------------------------
CPU-total             17                   17                   62
CPU-1                 17                   17                   62
CPU-2                 17                   16                   63
CPU-3                 17                   16                   62
CPU-4                 17                   17                   61
```

View memory usage

```
sonic# show system memory
------------------------------------------------------------
Attribute             Value/State
------------------------------------------------------------
Used                 :1304976
Total                :8162872
```

View system processes

```
sonic# show system processes
------------------------------------------------------------
PID        %CPU    %MEMORY    MEM-USAGE(Bytes)    NAME
------------------------------------------------------------
1          0        0        58761216    /sbin/init
10         0        0        0           [lru-add-drain]
100        0        0        0           [scsi_eh_0]
1000       0        0        409763840   docker
101        0        0        0           [scsi_tmf_0]
10179      0        0        12124160    /bin/bash
102        0        0        0           [scsi_eh_1]
10217      0        0        42135552    python
103        0        0        0           [scsi_tmf_1]
107        0        0        0           [bioset]
10862      0        0        256139264   /usr/sbin/rsyslogd
109        0        0        0           [kworker/3:1H]
11         0        0        0           [watchdog/0]
110        0        0        0           [kworker/2:1H]
11044      0        0        111427584   containerd-shim
11088      0        0        109920256   containerd-shim
111        0        0        0           [kworker/0:1H]
11119      0        0        109920256   containerd-shim
11140      0        0        59592704    /usr/bin/python
11177      0        0        111362048   containerd-shim
112        0        0        0           [kworker/1:1H]
11204      0        0        59600896    /usr/bin/python
11223      0        0        61095936    /usr/bin/python
11272      0        0        189263872   /usr/bin/orchagent
11308      0        0        58249216    /usr/bin/python
```

```
sonic# show system processes pid 1
-------------------------------------------
Attribute             Value/State
-------------------------------------------
Memory Usage         :58761216
Uptime               :84024
Start Time           :1563911114000000000
Name                 :/sbin/init
Args                 :None
Pid                  :1
Memory Utilization   :0
Cpu Utilization      :0
Cpu Usage System     :494
Cpu Usage User       :443
```

View EEPROM information

```
sonic# show platform syseeprom
-------------------------------------------------------------
Attribute            Value/State
-------------------------------------------------------------
Hardware-version    :A00
Mfg-name            :Dell
Name                :System Eeprom
Oper-status         :ACTIVE
Empty               :False
Part-no             :08YWFG
Id                  :S6000-ON
Location            :Slot 1
Removable           :False
Serial-no           :CN08YWFG282983AR0146A00
```

View fan, temperature, power supply, and adapter status

```
sonic# show platform environment

Fan Trays:
   Fan Tray 1:
       Fan State:                    Normal
       Fan1 Speed:                   12300 RPM
       Fan2 Speed:                   14850 RPM
   Fan Tray 2:
       Fan State:                    Normal
       Fan1 Speed:                   12150 RPM
       Fan2 Speed:                   15000 RPM
   Fan Tray 3:
       Fan State:                    Normal
       Fan1 Speed:                   12150 RPM
       Fan2 Speed:                   14850 RPM
   ...

Onboard Temperature Sensors:
   Fan U52:                         25 degrees C
       Baseboard U3:                 25 degrees C
       Fan U17:                      21 degrees C
       Near CPU:                     51 degrees C
       PSU1 hotspot:                 30 degrees C
       PSU1 inlet:                   22 degrees C
       PSU2 hotspot:                 31 degrees C
       PSU2 inlet:                   23 degrees C
       SW U04:                       28 degrees C
       SW U14:                       22 degrees C
       SW U16:                       21 degrees C
       SW U4403:                     40 degrees C
       SW U52:                       21 degrees C
       SW interal:                   43 degrees C

PSUs:
   PSU1:
       FAN RPM:                      15800 RPM
       Hotspot Temperature:          30 degrees C
       Inlet Temperature:            22 degrees C
       Input Current:                1.04 Amps
       Input Power:                  210 Watts
       Input Voltage:                207.90 Volts
       Output Current:               16.50 Amps
       Output Power:                 198 Watts
       Output Voltage:               12 Volts
   ...

coretemp-isa-0000
   Adapter: ISA adapter
       Core 0:                       +52.0 C  (high = +82.0 C, crit = +104.0 C)
       Core 1:                       +52.0 C  (high = +82.0 C, crit = +104.0 C)
       Physical id 0:                +52.0 C  (high = +82.0 C, crit = +104.0 C)
```

View fan status

```
sonic# show platform fanstatus
------------------------------------------------------------------------
Fan            Status         Speed (RPM)         Direction
------------------------------------------------------------------------
FAN 1          ACTIVE         6321                FAN_DIRECTION_EXHAUST
FAN 2          ACTIVE         6393                FAN_DIRECTION_EXHAUST
FAN 3          ACTIVE         6271                FAN_DIRECTION_EXHAUST
FAN 4          ACTIVE         6435                FAN_DIRECTION_EXHAUST
FAN 5          ACTIVE         6331                FAN_DIRECTION_EXHAUST
FAN 6          ACTIVE         6331                FAN_DIRECTION_EXHAUST
FAN 7          ACTIVE         6115                FAN_DIRECTION_EXHAUST
FAN 8          ACTIVE         6352                FAN_DIRECTION_EXHAUST
FAN 9          ACTIVE         6281                FAN_DIRECTION_EXHAUST
FAN 10         ACTIVE         6435                FAN_DIRECTION_EXHAUST
```

View temperature

```
sonic# show platform temperature
(TH - Threshold)
NAME    Temperature  High TH  Low TH  Crit High TH  Crit Low TH
------  -----------  -------  ------  ------------  -----------
CPU     45           90       10      100           0
```

View power supply status

```
sonic# show platform psustatus
--------------------------------------
PSU                  Status
--------------------------------------
PSU 1                ACTIVE
PSU 2                INACTIVE
```

```
sonic# show platform psusummary
PSU 1:
    description          02RPHX
    mfg-name             DELL
    fans                 2
    status-led           green
    oper-status          ACTIVE
    serial-no            CN-02RPHX-17972-56M-009Y-A00
    output-current (W)   1.29
    output-power (W)     129.50
    output-voltage (V)   12.19
    fan-speed (RPM)      5731
    fan-direction        FAN_DIRECTION_EXHAUST
PSU 2:
    description          02RPHX
    mfg-name             DELL
    oper-status          INACTIVE
    serial-no            CN-02RPHX-17972-56M-00A8-A00
    fan-speed (RPM)      0
    fan-direction        FAN_DIRECTION_EXHAUST
```

5

Software image management

SONiC allows you to install a maximum of two software images. The available images can be the current running image or the next-boot image. The current and next-boot images may be the same.

Use `show image list` to view the available images.

```
sonic# show image list
Current: SONiC-OS-3.0.0_RC10-Enterprise_Base
Next: SONiC-OS-3.0.0-Enterprise_Base
Available:
SONiC-OS-3.0.0-Enterprise_Base
SONiC-OS-3.0.0_RC10-Enterprise_Base
```

Install or upgrade SONiC image

You can install an image that you downloaded to the local file system or an image that is stored at a remote location using HTTP. The installed image is stored as the next-boot image.

```
sonic# image install file-url
```

The *file-url* parameter defines the location of the image in the format:

- `http[s]://`*hostip*`:/`*filepath* — Install the image from a remote HTTP or HTTPS server.
- `//`*filepath* — Install the image from the local or a USB file system.

If the current running image contains any modified text files or installed custom packages, they are not available in a different image. Back up the modified files and reinstall the packages after downloading a new image.

Set next-boot image

You can change the next-boot image by entering the image filename that is displayed in `show image list` output. You can set the same image as the current and next-boot image.

```
sonic# image set-default SONiC-OS-3.0.0-Enterprise_Base
sonic# show image list
Current: SONiC-OS-3.0.0-Enterprise_Base
Next: SONiC-OS-3.0.0-Enterprise_Base
Available:
SONiC-OS-3.0.0-Enterprise_Base
SONiC-OS-3.0.0_RC10-Enterprise_Base
```

Remove an image

You can delete an unused SONiC image by entering the image filename that is displayed in `show image list` output. You cannot remove the current running image.

```
sonic# image remove SONiC-OS-HEAD.140-dirty-20200105.093102
Remove image SONiC-OS-HEAD.138-dirty-20200103.154042? [y/N]:y
```

6

Zero touch provisioning

Zero touch provisioning (ZTP) automates SONiC switch deployment using common configuration templates. ZTP performs these tasks:

- Upgrades an existing SONiC image
- Runs a CLI batch file to configure the switch
- Runs a post-ZTP script to perform additional functions

ZTP is enabled by default when you boot a switch with a factory-installed SONiC for the first time, or when you perform an `ONIE: OS Install` from the ONIE boot menu. The switch communicates with a remote provisioning server, downloads configuration files, and runs scripts to configure the switch. ZTP supports any file transfer protocol, such as HTTP, FTP, and TFTP. SONiC ZTP uses a JSON file — `ztp.json` — to provision the switch. ZTP allows you to automate switch configuration without user intervention.

Topics:

ZTP operation

ZTP requires a user-defined input file in JSON format. The JSON file contains the data and logic to configure SONiC software modules.

At switch boot, ZTP checks for a startup configuration file: `/etc/sonic/config_db.json`. If the startup configuration file is not found, ZTP creates a temporary switch configuration to perform DHCP discovery to receive information about the location of the `ztp.json` file. If ZTP is disabled, ZTP exits and the switch boots with the factory default configuration.

ZTP JSON file

When a SONiC switch boots for the first time, ZTP checks if there is an existing ZTP JSON file. If no JSON file exists, DHCP Option 67 value is used to obtain the URL of the file. ZTP then downloads the ZTP JSON file and processes it. If the DHCP Option 67 value is not included, ZTP waits indefinitely until it is provided by the DHCP server. Other switch services, including the SWSS service, continue to boot.

A ZTP JSON file consists of multiple configuration sections that are defined by the user. All sections are enclosed in a `ztp` object. The ZTP JSON file example in the next section has four configuration sections: `01-firmware`, `02-configdb-json`, `03-provisioningscript`, and `04-connectivity-check`, which are included as values in the `ztp` object.

The top level `ztp` object also contains other objects, which specify the progress of the complete ZTP service, and input to the ZTP service on how to process the ZTP JSON file. In a ZTP JSON file, it is mandatory to define a `ztp` object as the first object. All other objects must appear nested in it. If the top level `ztp` object is not found, ZTP does not parse the JSON data. Also, the ZTP JSON file must be syntactically correct.

The configuration sections that are defined in the ZTP JSON file are processed in the lexical order of their names. It is recommended that you name the sections with a leading numerical number, such as `01`, `02`, `03` in the example, to indicate the order in which they are run.

Example: ZTP JSON file

The ZTP JSON file supports various options to fine-tune and extend switch configuration. For more information, see the functional description in Zero Touch Provisioning on GitHub. In the following example:

1. The SONiC firmware image is downloaded and installed on the switch. If you included the `"reboot-on-success": true` command is in the ztp jason file, the switch automatically reboots to use the newly installed image.
2. The startup configuration file `config_db.json` associated with the switch is downloaded and loaded. The `config_db.json` file is stored with the filename `$hostname_config_db.json` at the web root of the HTTP server with the IP address `192.168.1.1`. This filename allows ZTP to uniquely identify the configuration file associated with the switch.
3. A postprovisioning script `post_install.sh` is downloaded and run. The switch reboots if the `post_install.sh` script exits with a successful exit code 0.

4. A postboot connectivity check is performed by pinging hosts `10.1.1.1` and `yahoo.com` to verify connectivity.

```
{
   "ztp": {
     "01-firmware": {
       "install": {
          "url": "http://10.11.8.184/tftpboot/users/dtawde/image",
          "set-default": true
       },
       "reboot-on-success": true
     },
     "02-configdb-json": {
       "dynamic-url": {
         "source": {
           "prefix": "http://192.168.1.1/",
           "identifier": "hostname",
           "suffix": "_config_db.json"
         }
       }
     },
     "03-provisioning-script": {
       "plugin": {
         "url":"http://192.168.1.1/post_install.sh"
       },
       "reboot-on-success": true
       },
     "04-connectivity-check": {
       "ping-hosts": [ "10.1.1.1", "yahoo.com" ]
     }
   }
}
```

DHCP bootup configuration

When a switch boots with SONiC in ZTP mode, it starts the DHCP client on all interfaces — management and front-panel ports. ZTP configures all interfaces for untagged VLAN traffic. DHCPv4 and DHCPv6 address discovery are performed on all interfaces.

The first interface that receives a valid DHCP offer is used to determine the URL for the location of the ZTP JSON file. The DHCP offer sent to a DHCP server must include DHCPv4 Option 67 or DHCPv6 Option 59 to specify the ZTP JSON file URL. When the switch receives an IP address and a ZTP provisioning script URL from the DHCP server, it downloads and runs the JSON script.

If a ZTP JSON file is already present on the switch, ZTP processes it and does not download a ZTP JSON from a remote server. ZTP performs configuration steps in the file which may involve multiple switch reboots. After SONiC processes the ZTP JSON file, if the startup configuration file `/etc/sonic/config_db.json` is not found, ZTP restarts DHCP discovery to obtain a new ZTP JSON file and processes it.

- If the switch accesses the DHCP server using a front-panel port, the port interface must be in non-breakout mode.
- At least one of the front-panel ports that are connected to the network on which the DHCP server is running must be in non-breakout mode.

Cancel ZTP in progress

To exit ZTP operation and manually configure a switch by entering CLI commands, stop the ZTP process with `no ztp enable`.

- A factory default configuration is created and saved as the switch startup configuration. The switch continues to operate with the loaded factory default configuration.
- The CLI session is terminated. You must log on again to access the CLI.

```
sonic(config)# no ztp enable
```

To re-enable ZTP, enter `ztp enable` and reboot the switch. When you re-enable ZTP, the ZTP process does not start until you reboot the switch.

```
sonic(config)# ztp enable
```

ZTP logs

ZTP generates log messages on the console about its operational status.

View ZTP status

To view the status of the current ZTP session and the contents of the ZTP switch configuration, use the `show ztp status` command. The `show` output displays the configuration sections in the ZTP JSON file that are being processed and information about the last completed ZTP session.

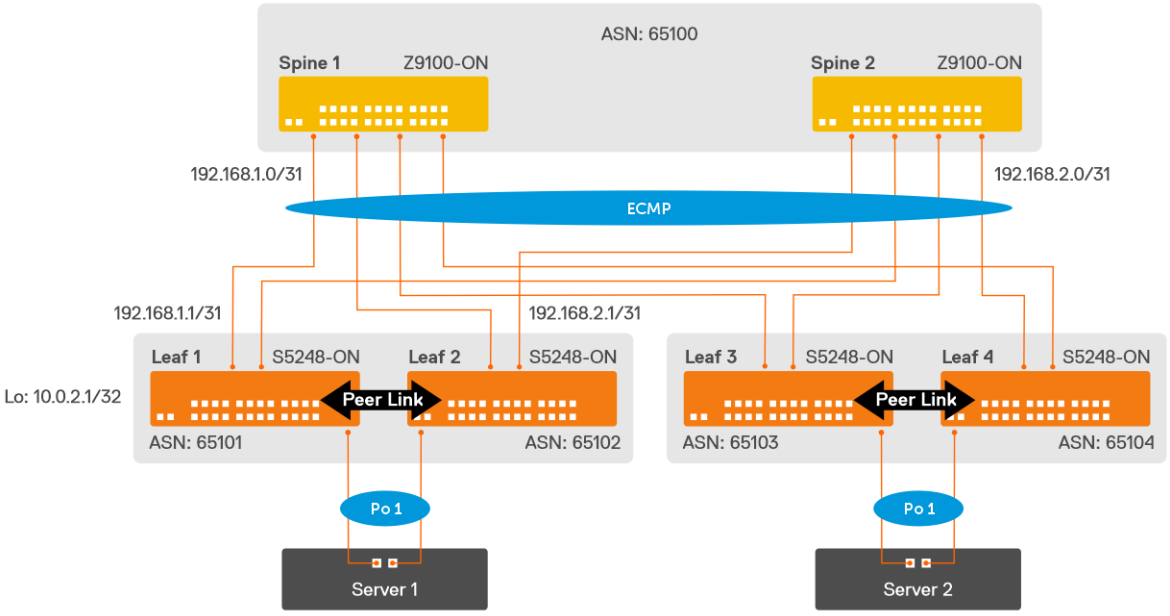
```
sonic# show ztp-status
=======================================
ZTP
=======================================
ZTP Admin Mode : True
ZTP Service    : Inactive
ZTP Status     : SUCCESS
ZTP Source     : dhcp-opt67 (eth0)
Runtime        : 05m 31s
Timestamp      : 2019-09-11 19:12:16 UTC
ZTP JSON Version : 1.0

ZTP Service is not running

---------------------------------------
01-configdb-json
---------------------------------------
Status          : SUCCESS
Runtime         : 02m 48s
Timestamp       : 2019-09-11 19:11:55 UTC
Exit Code       : 0
Ignore Result   : False

---------------------------------------
02-connectivity-check
---------------------------------------
Status          : SUCCESS
Runtime         : 04s
Timestamp       : 2019-09-11 19:12:16 UTC
Exit Code       : 0
Ignore Result   : False
```

- `ZTP Admin Mode` — Displays if ZTP is administratively enabled or disabled (`True` or `False`).
- `ZTP Service` — Displays ZTP status:
  - `Active Discovery` — ZTP is operational and performing DHCP discovery to learn the switch provisioning.
  - `Processing` — ZTP has discovered the switch provisioning information and is processing it.
- `ZTP Status` — Displays the current state and result of ZTP sessions:
  - `IN-PROGRESS` — ZTP is processing switch configuration information.
  - `SUCCESS` — ZTP has successfully processed the switch configuration information.
  - `FAILED` — ZTP failed to process the switch configuration information.
  - `Not Started` — ZTP has not started processing the discovered switch configuration information.
- `ZTP Source` — Displays the DHCP option and interface name from which the switch configuration information originated.
- `Runtime` — Displays the time that is taken for the ZTP process to complete from start to finish. For individual configuration sections, it indicates the time taken to process the associated configuration section.
- `Timestamp` — Displays the date/time stamp when the status field last changed.
- `ZTP JSON Version` — Version of the ZTP JSON file used for processing switch configuration information.
- `Status` — Displays the current state and result of processing a ZTP JSON file:
  - `IN-PROGRESS` — The configuration section that is currently being processed.
  - `SUCCESS` — The configuration section was processed successfully.
  - `FAILED` — The configuration section failed to execute successfully.
  - `Not Started` — ZTP has not started processing the configuration section.
  - `DISABLED` — The configuration section has been marked as disabled and will not be processed.
- `Exit Codes` — Displays the program exit code of the configuration section that was processed. A non-zero exit code indicates that the configuration section failed to execute successfully.
- `Ignore Results` — `True` indicates that the result of processing a configuration section is ignored and not used to evaluate the overall ZTP result.

- `Activity String` — Displays the current activity string, including the current action performed by ZTP, and how long it has been performing the activity.

Example: ZTP with JSON configuration file

In this example, an enterprise data center has deployed top-of-rack (ToR) leaf-switch pairs that are connected to downstream servers and upstream spine switches. ZTP is used to automate switch configuration and firmware upgrades. You avoid the need to manually configure individual switches.



The configuration areas that are defined in the ZTP JSON file are processed in the numerical order of their names. For example, `01-firmware`, then `02-configdb-json`, and so on. The numbered configuration sections provide a clear way to determine the order in which the tasks are run. This sample ZTP JSON file shows how to update the firmware and configure the Leaf3 switch.

```
{
  "ztp": {
      "01-firmware": {
          "install": {
             "url": "http://192.168.1.1/broadcom-sonic-v1.0.bin"
           }
    },
    "02-configdb-json": {
         "dynamic-url": {
             "source": {
                 "prefix": "http://192.168.1.1/",
           "identifier": "Leaf3",
                    "suffix": "¬_config_db.json"
      }
         }
      },
     "03-provisioning-script": {
          "plugin": {
            "url":"http://192.168.1.1/post_install.sh"
           },
           "reboot-on-success": true
      },
      "04-connectivity-check": {
          "ping-hosts": [ "172.16.11.1", "172.16.11.2" ]
        }
     }
}
```

At the first switch bootup, the ZTP JSON file performs these configuration steps:

1. The SONiC firmware image is downloaded and installed on the switch. The switch automatically reboots to load the newly installed image.

2. The startup configuration file `config_db.json` associated with the switch is downloaded and loaded as the running configuration. The `config_db.json` file is stored with the file name `$Leaf3_config_db.json` at the web root of the HTTP server with address 192.168.1.1. ZTP uniquely identifies the configuration file that is associated with each switch. In this way, multiple `config_db.json` files that are associated with different switches can be used.
3. A postprovisioning script `post_install.sh` is downloaded and run. The switch reboots if the `post_install.sh` script exits with a successful exit code `0`.
4. A postboot connectivity check is performed by pinging hosts 172.16.11.1 and 172.16.11.2 to verify connectivity.

ZTP DHCP server configuration

For ZTP operation, configure a DHCP server in the network by adding the required ZTP option shown in **bold**. Note that multiple `ztp.json` files are used to configure different switches.

```
option domain-name "dell.org";
option domain-name-servers ns1.dell.org, ns2.dell.org;
#To install from ONIE
#option default-url  "http://10.32.0.1/sonic-broadcom-cloud-base.bin";
option sonic-ztp code 67 = text;

default-lease-time 600;
max-lease-time 7200;

subnet 10.0.0.0 netmask 255.255.252.0 {
   option routers 10.0.0.253;

group {
   host leaf1 {
      hardware ethernet 0C:04:BA:AD:DE:40;
      fixed-address 10.0.0.10;
      option sonic-ztp "http://10.0.0.1/ztp/configs/s5248f/ztp.json"; }
   host leaf2 {
      hardware ethernet 28:4F:64:8B:52:19;
      fixed-address 10.0.0.11;
      option sonic-ztp "http://10.0.0.1/ztp/configs/s5248f/ztp.json"; }
   host spine1 {
      hardware ethernet 8C:04:BA:BA:45:A1;
      fixed-address 10.0.0.12;
      option sonic-ztp "http://10.0.0.1/ztp/configs/s5232f/ztp.json"; }
   host spine2 {
      hardware ethernet 8C:04:BC:AD:2B:B2;
      fixed-address 10.0.0.13;
      option sonic-ztp "http://10.0.0.1/ztp/configs/s5232f/ztp.json"; }
   }
}
```

7

Role-based access control

Role-based access control (RBAC) provides control for access and authorization. Users are granted permissions based on defined roles — not on their individual system user ID. Create user roles based on job functions to allow users appropriate system access.

RBAC places limitations on each role's permissions to allow you to partition tasks. You can assign each user only a single role, and many users can have the same role. A user role authenticates and authorizes a user at login, and places the user in EXEC mode.

Enterprise SONiC supports two predefined roles — `admin` and `operator`. Each user role assigns permissions that determine the commands that a user can enter, and the actions a user can perform. RBAC provides an efficient way to administer user rights. If a user's role matches one of the allowed user roles for a command, command authorization is granted.

- An `admin` user has full read/write access to system. When you log in with an `admin` role, you are placed in the Linux shell. The prompt is `admin@sonic:~$`.
- An `operator` user has only read access to system. When you log in with an `operator` role, you are placed in the Management Framework CLI. The prompt is `sonic#`.

Topics:

Create user and assign role

To limit switch access, assign a role when you configure each user.

- Enter a username, password, and role.

```
sonic(config)# username username password password role role
```

  - `username` *username* — Enter a text string (up to 32 alphanumeric characters; 1 character minimum)
  - `password` *password* — Enter a text string (up to 32 alphanumeric characters; 1 character minimum.)
  - `role` *role* — Enter a user role:
    - `admin` — Full access to all commands in the system, exclusive access to commands that change the file system, and access to the system shell. An administrator can create user IDs and assign roles.
    - `operator` — Access to EXEC mode to view the current configuration. An operator cannot modify configuration settings on a switch.

Create user and assign role

```
sonic(config)# username smith password silver403! role admin
```

Delete user

```
sonic(config)# no username smith
```

8

Authentication, authorization, and accounting

Authentication, authorization, and accounting (AAA) services secure networks against unauthorized access. Besides local authentication, Enterprise SONiC supports remote authentication dial-in user service (RADIUS) and terminal access controller access control system (TACACS+) client/server authentication systems.

For RADIUS and TACACS+, a switch acts as a client and sends authentication requests to a server that contains all user authentication and network service access information.

AAA configuration consists of setting up access control and accounting services.

1. Configure the authentication methods used to allow access to the switch.
2. Configure the level of command authorization for authenticated users.
3. Configure the collection of security information of user sessions.

Topics:

Configure authentication

AAA authentication verifies and grants user access to the switch. You can configure authentication to use the local username/password database or one or more remote TACACS+ servers.

Authentication methods

A switch uses a list of authentication methods to define the types of authentication and the sequence in which they apply. By default, only the `local` authentication method is used by authenticating users with the local user database. You can also configure one or more TACACS servers. The authentication methods in the method list run in the order you configure them. Re-enter the methods to change the order.

```
sonic(config)# aaa authentication login-method local
sonic(config)# aaa authentication login-method tacacs+
```

OR

```
sonic(config)# aaa authentication login-method local tacacs+
```

You must configure the TACACS+ server correctly and ensure that the connectivity to TACACS+ server is available through the Management interface. If you configure remote authentication using a TACACS+ server, all user logins are authenticated by the TACACS+ server. If the authentication fails, AAA checks the `failthrough` configuration and authenticates the user based on local database if fail-through is enabled.

Enable failthrough for TACACS+ authentication

Use the fail-through option if you configure TACACS+-based authentication with more than one remote server. The fail-through feature continues to access each TACACS+ server in the method list if an authentication request fails on one server. Enable or disable authentication fail-through with the command. Authentication fail-through is enabled by default.

```
sonic(config)# aaa authentication failthrough {enable | disable}
```

View AAA configuration

```
sonic# show aaa
------------------------------
AAA Authentication Information
------------------------------
failthrough  : False
login-method : local, tacacs+
```

Configure TACACS+ server

You can configure remote TACACS+ servers for user authentication. Enable failthrough for TACACS+ authentication

Add TACACS+ server to authentication list

You can configure up to seven TACACS+ servers for remote user authentication. When you use `tacacs-server host`, the configured TACACS+ server addresses are updated in the */etc/pam.d/common-auth-sonic* configuration file that is used by the TACACS service.

When a user logs in, TACACS+ servers perform authentication in the order the servers are listed in the method list. When a TACACS+ server times out, the switch contacts the next server in the method list according to the configured priority value. For information on using the fail-through option for TACACS+-based authentication, see Configure authentication.

```
sonic(config)# tacacs-server host {ip-address | ipv6-address} [port port-number] [timeout
seconds] [key secret] [type type] [priority value]
```

To configure a TACACS+ server, enter its IP or IPv6 address and these optional values:

- TCP port number on the server (1 to 65535; default 49)
- Transmission timeout in seconds (1 to 60; default 5)
- Secret key text that is shared between a TACACS+ server and the switch (up to 32 characters)
- Authentication type — `chap`, `pap`, or `mschap`; default `pap`. The authentication algorithm is used to encrypt/decrypt data that is sent and received between the switch and the TACACS+ server.
- Priority that used to access multiple TACACS+ servers to authenticate users (1 highest priority to 64; default 1)

```
sonic(config)# tacacs-server host 1.1.1.1 port 11 timeout 10 key mykey type pap priority 11
```

View configured TACACS+ servers

```
sonic# show tacacs-server host
------------------------------------------------------------------------------
HOST                 AUTH-TYPE       KEY         PORT       PRIORITY   TIMEOUT
------------------------------------------------------------------------------
1.1.1.1              pap             mykey       11         11         10
2.2.2.2              mschap          mykey2      20         20         20

sonic# show tacacs-server host 1.1.1.1
------------------------------------------------------------------------------
HOST                 AUTH-TYPE       KEY         PORT       PRIORITY   TIMEOUT
------------------------------------------------------------------------------
1.1.1.1              pap             mykey       11         11         10
```

Configure global TACACS+ authentication type

You can configure a default TACACS+ authentication type that is used for remote TACACS+-based user authentication. If you do not specify a *type* value in the `tacacs-server host` command, the default value is used. Different authentication types use different access-request and access-challenge messages.

- `chap` — Challenge handshake authentication protocol
- `pap` — Password authentication protocol (default)
- `mschap` — Microsoft challenge handshake authentication protocol

```
sonic(config)# tacacs-server auth-type authtype {pap | chap | mschap}
```

Configure global TACACS+ shared key

You can configure a global shared secret key that is used by the switch as a TACACS+ client to authenticate itself on a TACACS+ server (up to 32 characters). Valid characters are 0 to 9, A to Z, and a to z. The global shared key is used only on TACACS+ authentication servers which were configured without a `key` *`secret`* value in the `tacacs-server host` command.

```
sonic(config)# tacacs-server key testing123
```

Configure global TACACS+ source IP address

You can configure the source IP address that is used by the switch to communicate with TACACS+ servers. By default, no source IP address is configured.

```
sonic(config)# tacacs-server source-ip {ip-address | ipv6-address}
```

Configure global TACACS+ timeout

You can configure a global timeout value for all TACACS+ servers that are used for remote authentication (1 to 60 seconds; default 5). The global timeout is used only on TACACS+ authentication servers which were configured without a specified timeout with the `tacacs-server host` command.

```
sonic(config)# tacacs-server timeout 60
```

View global TACACS+ server settings

```
sonic# show tacacs-server global
 ---------------------------
TACACS Global Configuration
---------------------------
source-ip  : 1.1.1.1
timeout    : 10
auth-type  : chap
key        : mykey
```

9

Interfaces

Use the Management Framework CLI to configure Ethernet, VLAN, loopback, port channel, and the management interface. Enter configuration commands in Interface Configuration mode.

Ethernet interfaces

To configure an Ethernet interface, enter the front-panel port number.

```
sonic(config)# interface Ethernet 4
sonic(conf-if-Ethernet4)#
```

Management interface

To configure the management interface, enter `interface Management 0`.

```
sonic(config)# interface Management 0
sonic(conf-if-eth0)#
```

Loopback interfaces

To configure a loopback interface, enter `interface Loopback` *`number`* from 0 to 16383.

```
sonic(config)# interface Loopback 10
sonic(conf-if-lo10)#
```

VLAN interfaces

To configure a VLAN interface, enter `interface Vlan` *`vlan-id`* from 1 to 4094.

```
sonic(config)# interface Vlan 10
sonic(conf-if-Vlan10)#
```

Port-channel interfaces

To configure a port-channel interface, enter `interface PortChannel` *`number`* from 1 to 128.

```
sonic(config)# interface PortChannel 10
sonic(conf-if-po10)#
```

Topics:

Basic interface configuration

Use the commands in this section to configure basic settings on Ethernet, Management, loopback, VLAN, and port-channel interfaces. By default, an Ethernet or port-channel interface cannot receive or transmit L2 or L3 traffic unless it is configured as a VLAN member or assigned an IP address.

Autonegotiation — (Management interface only) Set a Management interface to autonegotiate speed with a connected device. Autonegotiation is disabled by default. Both sides of a link must have autonegotiation enabled or disabled for the link to come up.

- `autoneg on` — Enables interface speed autonegotiation.
- `autoneg off` — Disables interface speed autonegotiation.
- `no autoneg` — Resets interface speed autonegotiation to the default: `autoneg off`.

Description — (Ethernet and Management interfaces only) Enter a text description of an interface. Use the `show interface` command to view the descriptions configured for each interface.

- `description string` — Enter a maximum of 240 characters. To use special characters as part of the description string, enclose the string in double quotes. To use spaces between characters, enclose the entire description in quotation marks; for example, “text description”. The text string that you enter overwrites any previously configured text string.

IP address — Configure an IPv4 address on an interface.

- `ip address ip-address/mask` — Enter the IP address in dotted decimal format *A.B.C.D*. The `no ip address ip-address` command removes the IP address on the interface.

IPv6 address — Configure an IPv6 address on an interface.

- `ipv6 address ipv6-address/prefix-length` — Enter a full 128-bit IPv6 address with the network prefix length, including the 64-bit interface identifier. You can configure multiple IPv6 addresses on an interface. The `no ipv6 address ipv6-address` command removes an IPv6 address on the interface.

  To configure an IPv6 address besides the link-local address, use the `ipv6 address ipv6-address/prefix-length` command and specify the complete 128-bit IPv6 address.

MTU — Configure the maximum transmission unit (MTU) frame size for IP traffic on an interface.

- `mtu value` — Enter the maximum frame size in bytes (1312 to 9216; default 9216). Enter `no mtu` to reset the default value.

Shutdown — Disable an interface for incoming and outgoing traffic. To re-enable an interface, use the `no shutdown` command. When you shut down a VLAN, the L3 functions within the VLAN are disabled, and L2 traffic continues to flow.

Speed — (Management interface only) Configure the speed at which a Management interface transmits and receives data.

- `speed {10 | 100 | 1000 | auto}` — Enter the interface speed in Mbps. 1000 Mbps is 1 Gbps. The default is 10000, or 10 Gbps. Enter `auto` to configure autonegotiated speed on a Management interface.

Ethernet interface configuration example

```
sonic(config)# interface Ethernet 4
sonic(conf-if-Ethernet4)# description "RADIUS server 2"
sonic(conf-if-Ethernet4)# mtu 2000
sonic(conf-if-Ethernet4)# ip address 1.1.1.1/24
sonic(conf-if-Ethernet4)# ipv6 address a::d/64
sonic(conf-if-Ethernet4)# no shutdown
```

Management interface configuration example

```
sonic(config)# interface Management 0
sonic(conf-if-eth0)# description "Management 0"
sonic(conf-if-eth0)# mtu 2500
sonic(conf-if-eth0)# ip address 2.2.2.2/24
sonic(conf-if-eth0)# ipv6 address a::e/64
sonic(conf-if-eth0)# speed 100
sonic(conf-if-eth0)# autoneg off
sonic(conf-if-eth0)# no shutdown
```

Display interface configuration

```
# show interface status
------------------------------------------------------------------
Name            Description     Admin    Oper     Speed    MTU
------------------------------------------------------------------
Ethernet0       -               up       down     40GB     9100
Ethernet4       -               up       up       40GB     9100
Ethernet8       -               up       down     40GB     9100
Ethernet12      Ethernet12      up       down     40GB     9100
Ethernet16      -               up       down     40GB     9100
Ethernet20      -               up       down     40GB     9100
Ethernet24      -               up       down     40GB     9100
eth0            Management0     up       up       1000MB   1500
```

```
# show interface Management 0
eth0 is up, line protocol is up
Hardware is Mgmt
IPV4 address is 44.2.3.4/24
Mode of IPV4 address assignment: MANUAL
IPV6 address is a::e/64
Mode of IPV6 address assignment: MANUAL
IP MTU 1500 bytes
```

```
LineSpeed 1000MB, Auto-negotiation on
Input statistics:
        0 packets, 0 octets
        0 Multicasts, 0 Broadcasts, 0 Unicasts
        0 error, 0 discarded
Output statistics:
        0 packets, 0 octets
        0 Multicasts, 0 Broadcasts, 0 Unicasts
        0 error, 0 discarded
```

```
# show interface counters
-------------------------------------------------------------------
Interface    State  RX_OK  RX_ERR  RX_DRP  TX_OK  TX_ERR  TX_DRP
-------------------------------------------------------------------
Ethernet0    D      0       0       0       0      0       0
Ethernet4    U      1064    0       0       438    0       0
Ethernet8    D      0       0       0       0      0       0
Ethernet12   D      0       0       0       0      0       0
Ethernet16   D      0       0       0       0      0       0
Ethernet20   D      0       0       0       0      0       0
Ethernet24   D      0       0       0       0      0       0
Ethernet28   D      0       0       0       0      0       0
Ethernet32   U      431     0       0       438    0       0
Ethernet36   D      0       0       0       0      0       0
Ethernet40   D      0       0       0       0      0       0
eth0         U      23233   0       0       33220  0       0
```

```
sonic# clear counters interface Ethernet 0
Clear counters on Ethernet0 [confirm y/N]: y
sonic#

sonic# clear counters interface Ethernet 4
Clear counters on Etehrnet4 [confirm y/N]: y
sonic#
```

Configure access and trunk interfaces

By default, an Ethernet or port-channel interface is not in L2 mode. You must configure the interface to operate either as a trunk (tagged traffic) or access (untagged traffic) port:

- An access port sends and receives untagged frames from connected L2 devices. To assign an interface to the access VLAN, use the `switchport access Vlan` *`vlan-id`* command. To remove an interface from the access VLAN, enter the `no switchport access Vlan` command
- A trunk port sends and receives tagged frames from multiple VLANs and untagged frames from the access VLAN. To configure trunk VLANs, use the `switchport trunk allowed Vlan` *`vlan-id`* command. To remove a tagged VLAN membership, enter the `no switchport trunk allowed Vlan` *`vlan-id`* command.

```
sonic(config)# interface Ethernet 4
sonic(conf-if-Ethernet4)# switchport access Vlan 5
sonic(conf-if-Ethernet4)# exit
sonic(config)# interface PortChannel 4
sonic(conf-if-po4)# switchport access Vlan 5
```

```
sonic(config)# interface Ethernet 4
sonic(conf-if-Ethernet4)# switchport trunk allowed Vlan 5, 20-22
sonic(conf-if-Ethernet4)# exit
sonic(config)# interface port-channel 4
sonic(conf-if-po4)# switchport trunk allowed vlan 5
```

```
sonic(config)# interface Ethernet 4
sonic(conf-if-Ethernet4)# no switchport trunk allowed Vlan 21
sonic(conf-if-Ethernet4)# exit
sonic(config)# interface PortChannel 4
sonic(conf-if-po4)# no switchport access Vlan
```

10

Port channels

To provide redundancy, increased bandwidth, and better traffic load-balancing, bundle multiple physical interfaces into a single logical interface — a link aggregation group (LAG) or port channel. A port channel aggregates the bandwidth of member links. If a member port fails, traffic is redirected to the remaining ports.

A physical interface can belong to only one port channel at a time. A port channel must contain interfaces of the same speed. You can add up to 32 physical interfaces to a port channel. The configuration applied to the port channel bundle is used on all member interfaces. A port channel can operate in Active or On modes:

- Member interfaces in Active mode automatically initiate negotiations with other ports by using LACP packets, and set up a port channel with other ports in Active mode. Dynamic LACP is enabled on port interfaces in Active mode.
- Member interfaces in On mode are manually assigned as port-channel members. The LACP capability to dynamically group ports into a port channel is disabled.

Topics:

Port channel configuration

A port channel operates in either L2 (default) or L3 mode. To place a port channel in L3 mode and remove L2 configuration before you configure an IP address, use the `no switchport` command. To reconfigure a port channel in L2 mode, use the `switchport mode` command.

Perform these basic configuration tasks to set up port channels in your network.

Create a port channel

```
sonic# interface PortChannel portchannel-number [mode {active | on}] [min-links number]
[fallback]
```

- Port-channel ID numbers are from 0 to 9999.
- By default, the admin status is UP, the MTU is 9100 bytes, fallback is disabled, and the LACP mode is active. The LACP Fallback feature allows an active member interface to establish a connection with a peer interface before the port channel receives the LACP protocol negotiation from the peer.
- You can configure the minimum number of required links, from 1 to 255; default 0.
- To unconfigure a port channel, remove all member interfaces and enter the `no interface PortChannel portchannel-number` command.
- Configure the LACP mode for port-channel members:
  - `mode active` — All interfaces in the port channel start up with LACP enabled. Active ports dynamically negotiate with peer ports. Similarly configured ports are bundled as members of an active port channel. (Default)
  - `mode on` — Member ports operate as part of a static port channel with LACP disabled. A static port-channel member transmits IP traffic if the port interface is up, but does not dynamically negotiate with peer ports.

```
sonic(config)# interface PortChannel 1 mode active min-links 2 fallback

sonic(config)# interface PortChannel 2 mode active fallback

sonic(config)# interface PortChannel 3 mode on min-links 3
```

Add member interfaces

To create a static port channel LAG, manually assign member interfaces using the `channel-group` command.

- All members must be the same interface type; for example, Ethernet or Fibre Channel.
- All member interfaces must have the same speed.
- An interface must not contain non-default L2/L3 configuration settings. Only the `description` and `shutdown` or `no shutdown` commands are supported. You cannot add an IP address or static MAC address to a member interface.

- You can assign an interface to only one port channel.
- To remove a member interface, use the `no channel-group` command in Interface Configuration mode.

```
sonic(config)# interface Ethernet4
sonic(conf-if-Ethernet4)# channel-group 1
```

NOTE: After you assign a Layer 2 port to a port channel, all switchport configurations must be done on the port channel. You can no longer apply switchport configurations to individual port-channel members. Also, you cannot apply Layer 3 configurations to an individual port-channel member either. You must apply configuration changes to the entire port channel.

Configure a port channel

- **IP address** — Configure an IPv4 address on a port-channel interface.
  - `ip address` *`ip-address/mask`* — Enter the IP address in dotted decimal format *`A.B.C.D`*. The `no ip address` *`ip-address/mask`* command removes the IP address on the interface.

  IPv6 address — Configure an IPv6 address on a port-channel interface.
  - `ipv6 address` *`ipv6-address/prefix-length`* — Enter a full 128-bit IPv6 address with the network prefix length, including the 64-bit interface identifier. You can configure multiple IPv6 addresses on an interface. The `no ipv6 address` *`ipv6-address`* command removes the IPv6 address on the interface.
    - To configure an IPv6 address besides the link-local address, use the `ipv6 address` *`ipv6-address/prefix-length`* command and specify the complete 128-bit IPv6 address.
    - To configure a globally unique IPv6 address by entering only the network prefix and length, use the `ipv6 address` *`ipv6-address/prefix-length`* `eui-64` command.

  MTU — Configure the maximum transmission unit (MTU) frame size for a port channel.
  - `mtu` *`value`* — Enter the maximum frame size in bytes, from 1280 to 65535. The default port channel MTU is 9100 bytes. Enter `no mtu` to reset the default value.
  - Configure the MTU on port channel members first before you configure the port channel MTU. All members of a port channel must have the same MTU value. Tagged members must have a link MTU 4 bytes higher than untagged members to account for the packet tag.
  - Ensure that the MTU of port channel members is greater than or equal to the port channel MTU. If you configure the MTU on port channel members after you configure the port channel MTU, the port channel MTU may not be updated. The system selects the lowest MTU value that is configured on the port channel or port channel members to be the port channel MTU.

    For example, the port channel contains tagged members with Link MTU of 1522 and IP MTU of 1500 and untagged members with Link MTU of 1518 and IP MTU of 1500. The port channel's Link MTU cannot be higher than 1518 bytes and its IP MTU cannot be higher than 1500 bytes.
- **No shutdown** — Enable the port channel by entering `no shutdown`. To disable a port channel and place all member interfaces in an operationally down state, enter the `shutdown` command.

```
sonic# interface PortChannel 10
sonic(conf-if-po10)# mtu 2500
sonic(conf-if-po10)# ip address 2.2.2.2/24
sonic(conf-if-po10)# ipv6 address a::b/64
sonic(conf-if-po10)# no shutdown
```

Assign a port channel to a trunk or access VLAN

A port channel can operate in either a trunk or access VLAN — see Configure access and trunk interfaces. Configure access and trunk operation in port-channel configuration mode.

- An access port channel sends and receives untagged frames from connected L2 devices. The channel group interface is assigned to the access VLAN, which is VLAN 1 by default. To change the default access VLAN, use the `switchport access vlan` *`vlan-id`* command.
- A trunk port channel sends and receives tagged frames from multiple VLANs and untagged frames from the access VLAN. To configure a trunk port channel group, use the `switchport trunk allowed vlan` *`vlan-id`* command. To remove a tagged VLAN, enter the `no switchport trunk allowed vlan` *`vlan-id`* command.

To configure a port channel to operate in L2 Access or Trunk mode, use the `switchport mode` command. To restore a trunk port channel to L2 Access mode on VLAN1, use the `no switchport mode` command.

If you assign an IP address to a port channel, you cannot use the `switchport mode` command to enable L2 switching — you must first remove the IP address. Before you configure L3 mode on a port channel, use the `no switchport` command to remove all L2 configurations.

```
sonic(config)# interface PortChannel 4
sonic(conf-if-po4)# switchport mode access
sonic(conf-if-po4)# switchport access Vlan 5
```

```
sonic(config)# interface PortChannel 4
sonic(conf-if-po4)# switchport trunk allowed vlan 5
```

View port channel configuration

```
sonic# show PortChannel summary
Flags(oper-status):  D - Down U - Up
-------------------------------------------------------------
Group   PortChannel           Type    Protocol   Member Ports
-------------------------------------------------------------
100     PortChannel100 (D)    Eth     LACP       Ethernet84(D)
100     PortChannel200 (D)    Eth     NONE       Ethernet44(U)
100     PortChannel300 (D)    Eth     LACP       Ethernet40(U)
```

```
sonic# show interface PortChannel 100
PortChannel100 is down, line protocol is down, mode LACP
Minimum number of links to bring PortChannel up is 1
Fallback: Enabled
MTU 9100
LACP mode ACTIVE interval SLOW priority 65535 address 90:b1:1c:f4:9c:9f
Members in this channel: Ethernet84(Selected), Ethernet80
LACP Actor port 85  address 90:b1:1c:f4:9c:9f key 100
LACP Partner port 0  address 00:00:00:00:00:00 key 0
Last clearing of "show interface" counters: 1970-01-01 00:00:00
Input statistics:
          2972 packets, 712962 octets
2972 Multicasts, 0 Broadcasts, 0 Unicasts
0 error, 2972 discarded
Output statistics:
          2969 packets, 712560 octets
2969 Multicasts, 0 Broadcasts, 0 Unicasts
0 error, 0 discarded
```

11

VLANs

Virtual local area networks (VLANs) are logical interfaces that connect devices for common communication needs, independent of physical location. By default, VLANs operate in L2 mode. Physical interfaces and port channels can be members of VLANs.

Topics:

VLAN configuration

To configure a VLAN, create and configure the VLAN. Then add Ethernet and port-channel member interfaces.

```
sonic# interface Vlan 10
sonic(conf-if-Vlan10)#
```

- VLAN IDs are from 1 to 4094; there is no default VLAN.
- To disable a VLAN, use the `no interface Vlan` *`vlan-id`* command.

Perform these configuration tasks to set up VLANs in your network.

IP address — Configure an IPv4 address on a VLAN interface.

- `ip address` *`ip-address/mask`* — Enter the IP address in dotted decimal format *`A.B.C.D`*. The `no ip address` *`ip-address/mask`* command removes the IP address on the interface.

IPv6 address — Configure an IPv6 address on a VLAN interface.

- `ipv6 address` *`ipv6-address/prefix-length`* — Enter a full 128-bit IPv6 address with the network prefix length, including the 64-bit interface identifier. You can configure multiple IPv6 addresses on an interface. The `no ipv6 address` *`ipv6-address`* command removes an IPv6 address on the interface.

MTU — Configure the maximum transmission unit (MTU) frame size for IP traffic on VLAN member interfaces.

- `mtu` *`value`* — Enter the maximum frame size in bytes (1312 to 9216; default 9216 bytes) Enter `no mtu` to reset the default value.

  Configure the MTU on VLAN members first before you configure the VLAN MTU. All members of a VLAN must have the same MTU value. Tagged members must have a link MTU 4 bytes higher than untagged members to account for the packet tag.

VLAN configuration example

```
sonic# interface Vlan 10
sonic(conf-if-Vlan10)# mtu 2500
sonic(conf-if-Vlan10)# ip address 2.2.2.2/24
sonic(conf-if-Vlan10)# ipv6 address a::b/64
```

Display VLAN configuration

```
sonic# show Vlan
Q: A - Access (Untagged), T - Tagged
    NUM        Status       Q Ports
    5          Active       T Ethernet24
    10         Inactive
    20         Inactive     A PortChannel20
```

```
sonic# show Vlan 5
Q: A - Access (Untagged), T - Tagged
    NUM    Status     Q Ports
    5      Active     T Ethernet24
```

```
                       T PortChannel10
                       A Ethernet20
```

```
sonic# show interface Vlan 2
Vlan2 is up
Mode of IPV4 address assignment: not-set
Mode of IPV6 address assignment: not-set
IP MTU 9100 bytes
```

Add VLAN members

For information about how to add Ethernet and port-channel interfaces to an access or trunk VLAN, see Configure access and trunk interfaces.

12

Link Layer Discovery Protocol

The Link Layer Discovery Protocol (LLDP) enables devices on a local area network (LAN) to advertise and learn the capabilities of adjacent LAN devices. LLDP transmits and receives information from LLDP agents on other LAN devices. By default, LLDP is enabled on all interfaces.

- An LLDP-enabled interface supports only one neighbor.
- The switch receives and periodically transmits Link Layer Discovery Protocol Data Units (LLDPDUs). The default transmission interval is 30 seconds.
- LLDPDU information received from a neighbor expires after a specific amount of time, called time to live (TTL). The default TTL value is 120 seconds.
- Spanning-tree blocked ports allow LLDPDUs.
- Link layer discovery protocol-media endpoint discovery (LLDP-MED) is enabled on all interfaces by default.

View LLDP neighbors

```
sonic# show lldp table
-------------------------------------------------------------------------------
LocalPort    RemoteDevice        RemotePortID       Capability    RemotePortDescr
-------------------------------------------------------------------------------
Ethernet0    OS10                ethernet1/1/1:1    BOR           ethernet1/1/1:1
eth0         swtor-b2lab2-1409   ethernet1/1/9      BOR           ethernet1/1/9
```

```
sonic# show lldp neighbor
------------------------------------------------------------
LLDP Neighbors
------------------------------------------------------------
Interface:   Ethernet72,via: LLDP
  Chassis:
    ChassisID:    54:bf:64:b8:ce:c0
    SysName:      sonic
    SysDescr:     Debian GNU/Linux 9 (stretch) Linux 4.9.0-11-2-amd64 #1
                  SMP Debian 4.9.189-3+deb9u2 (2019-11-11) x86_64
    MgmtIP:
    Capability:   MAC_BRIDGE, ON
    Capability:   ROUTER, ON
  Port
    PortID:       tenGigE1/19/2
    PortDescr:    Ethernet73
------------------------------------------------------------
Interface:   Ethernet73,via: LLDP
  Chassis:
    ChassisID:    54:bf:64:b8:ce:c0
    SysName:      sonic
    SysDescr:     Debian GNU/Linux 9 (stretch) Linux 4.9.0-11-2-amd64 #1
                  SMP Debian 4.9.189-3+deb9u2 (2019-11-11) x86_64
    MgmtIP:
    Capability:   MAC_BRIDGE, ON
    Capability:   ROUTER, ON
  Port
    PortID:       tenGigE1/19/1
    PortDescr:    Ethernet72
------------------------------------------------------------
sonic#
```

13

Link Aggregation Control Protocol

Link aggregation control protocol (LACP) feature dynamically bundles multiple physical ports in a logical port channel or link aggregation group (LAG). A port channel is automatically created through the exchange of LACP packets between ports.

All port-channel member ports be the same speed and configured as either Layer 2 or Layer 3 interfaces. Traffic is load-balanced across all bundled links. If one member link fails, the LAG port channel continues to carry traffic over the remaining links.

LACP modes

LACP operates on an interface in Active and On modes.

Active	LACP runs on any interface/link that is configured in this mode. A port in Active mode automatically initiates negotiations with other ports using LACP packets, and sets up a port channel with another port that is in Active mode.
On	The interface acts as a member of a static LAG. Only manually assigned interfaces are included as members of a port channel in On mode.

When you add an interface as a member of a port channel, the interface is assigned the LACP mode that is configured for the port channel — see Port channel configuration. For example, if you configure the port channel to operate in On mode, all interfaces assigned to it operate in On mode.

```
sonic# interface portchannel 10 mode on
sonic(conf-if-po10)# exit
sonic(config)# interface Ethernet4
sonic(conf-if-Ethernet4)# channel-group 10
```

View LACP-enabled interfaces

To display the LACP-enabled port channels and their member interfaces, use the `show interface PortChannel` command. In the show output, mode LACP indicates the interfaces that are dynamically bundled by LACP in port channels.

```
sonic# show interface PortChannel
PortChannel1 is up, line protocol is down, mode LACP
Minimum number of links to bring PortChannel up is 1
Fallback: Enabled
MTU 9100
LACP mode ACTIVE interval SLOW priority 65535 address ec:f4:bb:fc:09:8b
Members in this channel: Ethernet64
LACP Actor port 65  address ec:f4:bb:fc:09:8b key 1
LACP Partner port 0  address 00:00:00:00:00:00 key 0
Members in this channel: Ethernet72
LACP Actor port 73  address ec:f4:bb:fc:09:8b key 1
LACP Partner port 0  address 00:00:00:00:00:00 key 0
Last clearing of "show interface" counters: 1970-01-01 00:00:00
Input statistics:
          0 packets, 0 octets
0 Multicasts, 0 Broadcasts, 0 Unicasts
0 error, 0 discarded
Output statistics:
          0 packets, 0 octets
0 Multicasts, 0 Broadcasts, 0 Unicasts
0 error, 0 discarded
```

14

Media access control

All Ethernet ports maintain media access control (MAC) address tables. To avoid flooding all ports when it forwards packets, a switch uses the MAC address table. This table is dynamically populated when the switch receives a frame. It associates the MAC address of the sending network device with the port on which the frame was received. The switch can later forward packets to a single port.

A MAC address table contains learned MAC addresses of network devices and port/VLAN information to determine how to forward traffic between ports to destination devices. MAC addresses are entered in the table as dynamic entries that age out or are statically configured with no age-out time. A MAC address is a 48-bit number in the format *nn.nn.nn.nn.nn.nn*.

Use `show mac-address table` commands to view MAC address table information.

```
show mac address-table [address mac-address] [interface ethernet port-number]
[vlan vlan-id] [portchannel number] [count] [static]
```

Display MAC address table entries

```
sonic# show mac address-table
------------------------------------------------------------
VLAN          MAC-ADDRESS          TYPE          INTERFACE
------------------------------------------------------------
10           00:00:00:00:00:01    STATIC        Ethernet0
11           00:00:00:00:00:01    STATIC        Ethernet0
100          00:00:00:00:00:10    DYNAMIC       Ethernet36
20           00:00:00:00:00:02    DYNAMIC       Ethernet4
30           00:00:00:00:00:03    STATIC        Ethernet8
40           00:00:00:00:00:04    DYNAMIC       Ethernet12
50           00:00:00:00:00:05    STATIC        Ethernet16
60           00:00:00:00:00:06    DYNAMIC       Ethernet20
70           00:00:00:00:00:07    STATIC        Ethernet24
80           00:00:00:00:00:08    DYNAMIC       Ethernet28
90           00:00:00:00:00:09    STATIC        Ethernet32
10           00:00:00:00:00:98    STATIC        Ethernet0
99           00:00:00:00:00:99    STATIC        PortChannel10
```

Display MAC address table entries per MAC address

```
sonic# show mac address-table address 00:00:00:00:00:01
------------------------------------------------------
VLAN          MAC-ADDRESS          TYPE          INTERFACE
------------------------------------------------------
10           00:00:00:00:00:01    STATIC        Ethernet0
11           00:00:00:00:00:01    STATIC        Ethernet0
```

Display VLAN entries in MAC address table

```
sonic# show mac address-table Vlan 10
------------------------------------------------------
VLAN          MAC-ADDRESS          TYPE          INTERFACE
------------------------------------------------------
10           00:00:00:00:00:01    STATIC        Ethernet0
10           00:00:00:00:00:101   STATIC        Ethernet0
```

```
show mac address-table static Vlan 11
------------------------------------------------------
VLAN          MAC-ADDRESS          TYPE          INTERFACE
------------------------------------------------------
11           00:00:00:00:00:01    STATIC        Ethernet0
```

```
sonic# show mac address-table dynamic Vlan 60
------------------------------------------------------
VLAN          MAC-ADDRESS          TYPE          INTERFACE
```

```
-----------------------------------------------------------
60          00:00:00:00:00:06   DYNAMIC       Ethernet20
```

Display MAC address count

```
sonic# show mac address-table count
MAC Entries for all vlans :  13
Dynamic Address Count :  5
Static Address (User-defined) Count :  8
Total MAC Addresses in Use: 13
```

Display MAC address entries per interface

```
sonic# show mac address-table interface Ethernet 0
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
10          00:00:00:00:00:01   STATIC        Ethernet0
11          00:00:00:00:00:01   STATIC        Ethernet0
10          00:00:00:00:00:98   STATIC        Ethernet0
```

```
sonic# show mac address-table static interface Ethernet 8
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
30          00:00:00:00:00:03   STATIC        Ethernet8
```

```
sonic# show mac address-table dynamic interface Ethernet 12
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
40          00:00:00:00:00:04   DYNAMIC       Ethernet12
```

Display MAC address entries per port channel

```
sonic# show mac address-table interface PortChannel 10
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
99          00:00:00:00:00:99   STATIC        PortChannel10
```

```
sonic# show mac address-table static interface PortChannel 10
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
99          00:00:00:00:00:99   STATIC        PortChannel10
```

```
sonic# show mac address-table dynamic interface PortChannel 11
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
98          00:00:00:00:00:95   DYNAMIC       PortChannel11
```

Display static MAC address entries

```
sonic# show mac address-table static
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
10          00:00:00:00:00:01   STATIC        Ethernet0
11          00:00:00:00:00:01   STATIC        Ethernet0
30          00:00:00:00:00:03   STATIC        Ethernet8
50          00:00:00:00:00:05   STATIC        Ethernet16
70          00:00:00:00:00:07   STATIC        Ethernet24
90          00:00:00:00:00:09   STATIC        Ethernet32
```

```
10          00:00:00:00:00:98   STATIC        Ethernet0
99          00:00:00:00:00:99   STATIC        PortChannel10
```

```
sonic# show mac address-table static address 00:00:00:00:00:01
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
10          00:00:00:00:00:01   STATIC        Ethernet0
11          00:00:00:00:00:01   STATIC        Ethernet0
```

Display dynamic MAC address entries

```
sonic# show mac address-table dynamic
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
100         00:00:00:00:00:010  DYNAMIC       Ethernet36
20          00:00:00:00:00:02   DYNAMIC       Ethernet4
40          00:00:00:00:00:04   DYNAMIC       Ethernet12
60          00:00:00:00:00:06   DYNAMIC       Ethernet20
80          00:00:00:00:00:08   DYNAMIC       Ethernet28
```

```
sonic# show mac address-table dynamic address 00:00:00:00:00:06
-------------------------------------------------------------
VLAN         MAC-ADDRESS         TYPE         INTERFACE
-------------------------------------------------------------
60          00:00:00:00:00:06   DYNAMIC       Ethernet20
```

15

Address resolution protocol

Address resolution protocol (ARP) allows IPv4 packets to be sent across networks by translating IP network addresses to MAC hardware addresses, and MAC addresses to IP addresses.

The MAC address and corresponding IP address of destination devices are maintained in the ARP table. Using the IP address, the ARP table allows the switch to quickly retrieve the associated MAC address, encapsulate the IP packet in a L2 frame, and transmit it over the network to a destination.

NOTE: For IPv6 address translation, SONiC uses the network discovery protocol (NDP).

Topics:

View IPv4 ARP entries

To display ARP table entries, use the `show ip arp` command. To filter the output, specify an interface, port channel, or VLAN, an IP address in the format *x.x.x.x*, a MAC address, or a combination of more than one value to match. You can also display summary information.

```
show ip arp [interface {Ethernet port-number [summary] | PortChannel number [summary] | Vlan
vlan-id [summary] | Management port-number [summary]}] [ip-address] [mac-address mac-address]
[summary]
```

```
sonic# show ip arp
------------------------------------------------------------------------
Address         Hardware address    Interface        Egress Interface
------------------------------------------------------------------------
192.168.1.4     00:01:02:03:44:55   Ethernet8        -
192.168.2.4     00:01:02:03:ab:cd   PortChannel200   -
192.168.3.6     00:01:02:03:04:05   Vlan100          Ethernet4
10.11.48.254    00:01:e8:8b:44:71   eth0             -
10.14.8.102     00:01:e8:8b:44:71   eth0             -
0.0.0.0         00:00:00:00:00:00   lo
```

NOTE: The last ARP entry for the loopback IP address 0.0.0.0 may be entered from the SONiC `NEIGH_TABLE` of `APPL_DB`. To delete the entry, you must manually delete it from the `REDIS_DB` using the redis-cli or any other application that allows you to edit the `REDIS_DB`.

```
sonic# show ip arp interface Vlan 20
Address        Hardware address    Interface         Egress Interface
------------------------------------------------------------------------
20.0.0.2       90:b1:1c:f4:9d:ba   Vlan20            Ethernet0
20.0.0.5       00:11:22:33:44:55   Vlan20            Ethernet0
```

```
sonic# show ip arp 20.0.0.2
Address        Hardware address    Interface         Egress Interface
------------------------------------------------------------------------
20.0.0.2       90:b1:1c:f4:9d:ba   Vlan20            Ethernet0
```

```
sonic# show ip arp mac-address 90:b1:1c:f4:9d:ba
Address        Hardware address    Interface         Egress Interface
```

```
-------------------------------------------------------------------
20.0.0.2       90:b1:1c:f4:9d:ba   Vlan20            Ethernet0
```

```
 sonic# show ip arp summary
Total Entries
-------------
     2
```

```
sonic# show ip arp vrf Vrf_1
-------------------------------------------------------------------
Address        Hardware address    Interface         Egress Interface
-------------------------------------------------------------------
20.0.0.2       90:b1:1c:f4:9d:ba   Vlan20            Ethernet0
20.0.0.5       00:11:22:33:44:55   Vlan20            Ethernet0
```

Clear IPv4 entries

To delete dynamically learned IPv4 entries from the ARP table, use the `clear ip arp` command. To specify the entries to be deleted, enter an interface, port channel, or VLAN, an IPv4 address, or a combination to match. Enter `force` to delete statically configured ARP entries. Use the `show ip arp` command to verify that the IPv4 entries have been deleted.

```
clear ip arp [interface {Ethernet port-number | PortChannel number | Vlan vlan-id |
Management port-number}] [ip-address] [force]
```

```
sonic# show ip arp
-------------------------------------------------------------------
Address         Hardware address    Interface        Egress Interface
-------------------------------------------------------------------
192.168.1.4     00:01:02:03:44:55   Ethernet8        -
192.168.2.4     00:01:02:03:ab:cd   PortChannel200   -
192.168.3.6     00:01:02:03:04:05   Vlan100          Ethernet4
10.11.48.254    00:01:e8:8b:44:71   eth0             -
10.14.8.102     00:01:e8:8b:44:71   eth0             -

sonic# clear ip arp interface Vlan 100 force
sonic# show ip arp
-------------------------------------------------------------------
Address         Hardware address    Interface        Egress Interface
-------------------------------------------------------------------
192.168.1.4     00:01:02:03:44:55   Ethernet8        -
192.168.2.4     00:01:02:03:ab:cd   PortChannel200   -
10.11.48.254    00:01:e8:8b:44:71   eth0             -
10.14.8.102     00:01:e8:8b:44:71   eth0             -

sonic# clear ip arp interface Management 0
sonic# show ip arp
-------------------------------------------------------------------
Address         Hardware address    Interface        Egress Interface
-------------------------------------------------------------------
192.168.1.4     00:01:02:03:44:55   Ethernet8        -
192.168.2.4     00:01:02:03:ab:cd   PortChannel200   -
10.14.8.102     00:01:e8:8b:44:71   eth0
```

View IPv6 NDP entries

To view MAC addresses that correspond to IPv6 addresses in the neighbor discovery protocol (NDP) table, use the `show ipv6 neighbors` command. To filter the output, specify an interface, port channel, or VLAN, or an IPv6 address in the format *A::B*, or a combination to match. You can also view summary information.

```
sonic# show ipv6 neighbors [interface {Ethernet port-number [summary] | PortChannel number
[summary] | Vlan vlan-id [summary]}] [ipv6-address] [mac-address mac-address] [summary]
```

```
sonic# show ipv6 neighbors
-------------------------------------------------------------------------------
Address                      Hardware address    Interface   Egress Interface
-------------------------------------------------------------------------------
```

```
20::2                         aa:bb:cc:dd:ee:ff   Ethernet8   -
fe80::e6f0:4ff:fe79:34c7      e4:f0:04:79:34:c7   eth0        -
```

```
sonic# show ipv6 neighbors 20::2
------------------------------------------------------------------------------
Address                       Hardware address    Interface   Egress Interface
------------------------------------------------------------------------------
20::2                         aa:bb:cc:dd:ee:ff   Ethernet8   -
```

```
sonic# show ipv6 neighbors mac-address e4:f0:04:79:34:c7
------------------------------------------------------------------------------
Address                     Hardware address    Interface   Egress Interface
------------------------------------------------------------------------------
fe80::e6f0:4ff:fe79:34c7    e4:f0:04:79:34:c7   eth0        -
```

```
sonic# show ipv6 neighbors interface Management 0
------------------------------------------------------------------------------
Address                    Hardware address    Interface    Egress Interface
------------------------------------------------------------------------------
fe80::e6f0:4ff:fe79:34c7   e4:f0:04:79:34:c7   eth0         -
```

Clear IPv6 entries

To delete dynamically learned IPv6 entries from the NDP table, use `clear ipv6 neighbors`. To specify the entries to delete, enter an interface, port channel, or VLAN, an IPv6 address, or a combination to match. Enter `force` to delete statically configured ARP entries. Use `show ipv6 neighbors` to verify that the IPv6 entries have been deleted.

```
sonic# clear ipv6 neighbors [interface {Ethernet port-number | PortChannel number | Vlan vlan-
id | Management port-number}] [ipv6-address] [force]
```

```
sonic# show ipv6 neighbors
------------------------------------------------------------------------------
Address                       Hardware address    Interface   Egress Interface
------------------------------------------------------------------------------
20::2                         aa:bb:cc:dd:ee:ff   Ethernet8   -
fe80::e6f0:4ff:fe79:34c7      e4:f0:04:79:34:c7   eth0        -

sonic# clear ipv6 neighbors 20::2 force
sonic# show ipv6 neighbor
------------------------------------------------------------------------------
Address                       Hardware address    Interface   Egress Interface
------------------------------------------------------------------------------
fe80::e6f0:4ff:fe79:34c7      e4:f0:04:79:34:c7   eth0        -

sonic# clear ipv6 neighbors
sonic# show ipv6 neighbors
sonic#
```

16

Border Gateway Protocol

The Border Gateway Protocol (BGP) transmits interdomain routing information within and between autonomous systems (AS). Each AS can use its own routing policy. BGP exchanges network reachability information and routing updates between peer devices.

BGP's best-path algorithm determines the best paths to a neighbor device. The use of multiple paths from one router to another adds reliability to network connections. BGP uses TCP as its transport protocol.

BGP peers exchange routes in their BGP routing tables and continue to send each other routing updates. BGP avoids creating loops between routing domains by rejecting path updates that contain the local AS.

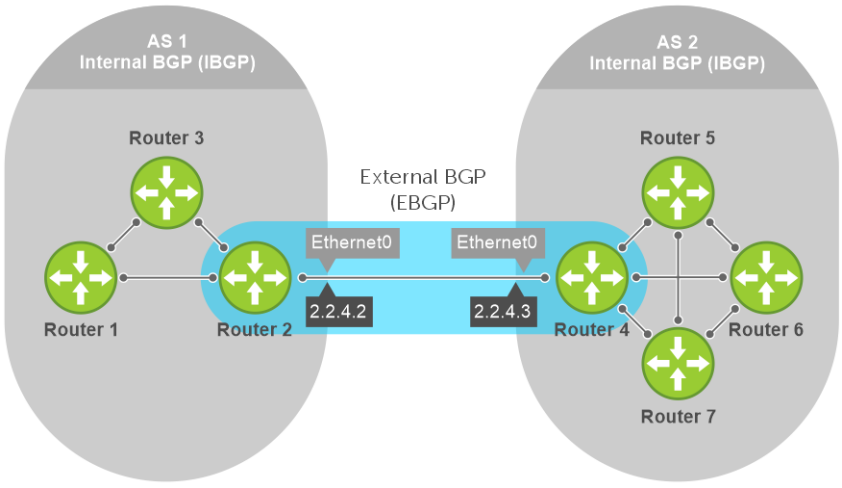
Autonomous systems

BGP autonomous systems are a collection of nodes in individual routing domains which use local routing policies. In an AS, multiple routing protocols are supported to exchange routing updates with BGP. Each AS has a number, which an Internet authority assigns—you do not assign the BGP number.

The Internet assigned numbers authority (IANA) identifies each network with a unique AS number (ASN). AS numbers 64512 through 65534 are reserved for private purposes. AS numbers 0 and 4294967245 cannot be used in a live environment. IANA assigns valid AS numbers in the range of 1 to 64511.

Multihomed AS	Maintains connections to more than one other AS. This group allows the AS to remain connected to the Internet if a complete failure occurs to one of their connections. This type of AS does not allow traffic from one AS to pass through on its way to another AS.
Stub AS	Connected to only one AS.
Transit AS	Provides connections through itself to separate networks. For example, Router 1 uses Router 2—the transit AS, to connect to Router 4. Internet service providers (ISPs) are always a transit AS because they provide connections from one network to another. An ISP uses a transit AS to sell transit service to a customer network.

When BGP operates inside an AS - AS1 **or** AS2, it functions as an Internal Border Gateway Protocol (IBGP). When BGP operates between AS endpoints - AS1 **and** AS2, it functions as an External Border Gateway Protocol (EBGP). IBGP provides routers inside the AS with the path to reach a router external to the AS. EBGP routers exchange information with other EBGP routers and IBGP routers to maintain connectivity and accessibility.



Topics:

- Enable BGP
- Configure BGP router
- Configure BGP address family
- Configure BGP neighbor
- BGP peer groups
- BGP routing policy filters

- Unnumbered BGP
- View BGP configuration

Enable BGP

BGP is disabled by default. To enable BGP routing with peer devices, you must enable BGP and have at least one BGP neighbor configured on the router. The BGP neighbor can be statically or dynamically configured.

A BGP router does not automatically discover other BGP devices. You must manually configure them. To establish BGP sessions and route traffic, configure at least one BGP neighbor or peer. In BGP, routers with established TCP connections are called *neighbors* or *peers*. After a connection establishes, the neighbors exchange full BGP routing tables with incremental updates afterward. Neighbors also exchange the keepalive messages to maintain the connection.

NOTE: *Neighbors* and *peers* both mean the same thing; namely, BGP-enabled routers that are directly connected or remotely connected over other BGP routers.

You can classify BGP neighbor routers and peers as internal or external. Connect eBGP peers directly, unless you enable eBGP multihop — iBGP peers do not need direct connection. The IP address of an eBGP neighbor is usually the IP address of the interface that is directly connected to the router. The BGP process first determines if all internal BGP peers are reachable, then it determines which peers outside the AS are reachable.

These steps are required configuration settings. Repeat the configuration steps on each peer router to enable BGP route exchange between the routers.

1. Assign an autonomous system, and enter BGP configuration mode. Enter an AS number to use in the BGP routing process. Only one AS number is supported on a router.

   ```
   sonic(config)# router bgp local_asn [vrf vrf-name]
   ```

   - For `local_asn`, enter a local AS number from 1 to 4294967295.
   - (Optional) Enter a VRF name to configure BGP in a VRF instance.
2. Configure a unique router ID using an IPv4 address. The BGP router ID is used to identify BGP peers in routing sessions.

   ```
   sonic(config-router-bgp)# router-id ipv4-address
   ```

   - By default, BGP sets the router ID to the IPv4 address of the loopback interface. If the loopback interface is not configured, the highest IPv4 address available in the system is used.
   - When you configure a router ID, all active BGP peer sessions are reset.
3. Configure each BGP neighbor by entering an IPv4 or IPv6 address or the local interface that is connected to the peer.

   ```
   sonic(config-router-bgp)# neighbor {ip-address | ipv6-address | interface {Ethernet port-number | PortChannel number | Vlan vlan-id}}
   ```
4. Configure the AS number of the peer. Enter `internal` to configure a peer in the local AS to exchange routing information through internal BGP (iBGP) peer sessions. Enter `external` to configure a peer in an external or remote AS to exchange routing information through external BGP (eBGP) peer sessions.

   ```
   sonic(config-router-bgp-neighbor)# remote-as {peer_asn | internal | external}
   ```
5. (Optional) Enter a text description of the neighbor.

   ```
   sonic(config-router-bgp-neighbor)# description text
   ```
6. Enable BGP routing with the configured peer.

   ```
   sonic(config-router-bgp-neighbor)# no shutdown
   ```
7. Repeat the steps on the peer to activate a BGP exchange with the local router.

Enable BGP example

```
sonic(config)# router bgp 100
sonic(config-router-bgp)# router-id 11.1.1.1
sonic(config-router-bgp)# neighbor 5.1.1.1
sonic(config-router-neighbor)# remote-as 1
sonic(config-router-neighbor)# description  n1_abcd
sonic(config-router-neighbor)# no shutdown
```

View routes in BGP table

```
sonic# show ip bgp [vrf vrf-name] {ipv4 | ipv6} [summary]
```

```
sonic# show ip bgp summary
BGP router identifier 2.2.2.2, local AS number 100
Neighbor      V   AS    MsgRcvd   MsgSent   InQ   OutQ   Up/Down    State/PfxRcd
101.2.2.2     4   100   36        37        0     0      00:19:31   10
```

```
sonic# show ip bgp ipv4 summary
BGP router identifier 2.2.2.2, local AS number 100
Neighbor      V   AS    MsgRcvd   MsgSent   InQ   OutQ   Up/Down    State/PfxRcd
101.2.2.2     4   100   36        37        0     0      00:19:31   10
```

```
sonic# show ip bgp ipv4
BGP routing table information for VRF default
Router identifier 2.2.2.2, local AS number 100
Route status codes: * - valid, > - best
Origin codes: i - IGP, e - EGP, ? - incomplete
     Network          Next Hop    Metric   LocPref   Path
*    10.0.0.0/8       101.2.2.2   0        100       ?
*>                    0.0.0.0     0                  ?
*>   21.0.0.0/8       101.2.2.2   0        100       i
*>   22.0.0.0/8       101.2.2.2   0        100       i
*>   31.0.0.0/8       101.2.2.2   0        100       i
*>   32.0.0.0/8       101.2.2.2   0        100       i
*>   34.0.0.0/8       101.2.2.2   0        100       i
*    101.2.2.0/24     101.2.2.2   0        100       ?
*>                    0.0.0.0     0                  ?
```

```
sonic# show ip bgp ipv6
BGP routing table information for VRF default
Router identifier 2.2.2.2, local AS number 100
Route status codes: * - valid, > - best
Origin codes: i - IGP, e - EGP, ? - incomplete
     Network            Next Hop              Metric   LocPref   Path
*>   2121::/64          fe80::92b1:1cff:fef4:ab9b0   100       i
*>   2122::/64          fe80::92b1:1cff:fef4:ab9b0   100       i
*>   2123:3322::/64     fe80::92b1:1cff:fef4:ab9b0   100       i
```

```
sonic# show ip bgp ipv6 summary
BGP router identifier 2.2.2.2, local AS number 100
Neighbor      V   AS    MsgRcvd   MsgSent   InQ   OutQ   Up/Down     State/PfxRcd
101.2.2.2     4   100   37        38        0     0      00:20:14    10
```

Configure BGP router

After you enable BGP and enable BGP routing with peer devices, perform the tasks in this section to configure additional BGP features or reconfigure default settings.

BGP defaults

BGP is enabled with these default settings:

- IPv4 unicast — Disabled
- Neighbor state-change logs — Disabled
- Fast external failover — Enabled
- Graceful restart — Disabled
- Local preference — 100
- 4-byte AS — Enabled
- Multiexit discriminator (MED) — 0
- Route flap dampening:
  - half-life — 15 minutes
  - max-suppress-time — 60 minutes

    - reuse — 750
    - suppress — 2000
- Timers:
    - keepalive — 60 seconds
    - holdtime — 180 seconds
- Add-path — Disabled

Set BGP timers

(Optional) Configure the time (in seconds) between sending keepalive messages to a BGP neighbor, from 1 to 65535; default 60. Configure the hold-time to wait (in seconds) to receive a keepalive message before considering a BGP peer to be dead, from 3 to 65535; default 180.

```
sonic(config-router-bgp)# timers keepalive holdtime
```

Reset fast fallover

(Optional) By default, a BGP session with a neighbor is automatically reset if the link goes down. If you disable fast fallover with the `no fast-external-fallover` command, you can re-enable it.

```
sonic(config-router-bgp)# fast-external-fallover
```

- Configure BGP fast fallover settings in a VRF instance from BGP VRF configuration mode:

```
sonic(config)# router bgp local_asn vrf vrf-name
sonic(config-router-bgp-vrf)# fast-external-fallover
```

Log neighbor changes

(Optional) By default, changes in neighbor status — up and down — are automatically logged for troubleshooting purposes. To view the log, use the `bgp config` command. If you disable this log, you can re-enable it with this command.

```
sonic(config-router-bgp)# log-neighbor-changes
```

- In a VRF instance, re-enable neighbor change logging from BGP VRF configuration mode:

```
sonic(config)# router bgp local_asn vrf vrf-name
sonic(config-router-bgp)# log-neighbor-changes
```

Best-path selection

(Optional) By default, the BGP algorithm selects the best path to a destination when given multiple alternative paths, using the prefix and path information stored in the BGP routing table. You can reconfigure the default settings that determine the best-path selection, including:

- `as-path confed` — Selects a best path based on AS paths in the confederation
- `as-path ignore` — Does not take into account the AS path length. The default is to use the AS path in best-path calculation.
- `as-path multipath-relax` — Permits paths of equal length to be selected to allow for load sharing. The default is to select a best path that is an exact match from multiple alternative paths.
- `as-path med {confed | missing-as-worst}` — Uses the multi-exit discriminator (MED) value to select the best path learned from confederation peers if no external AS is in the path, or assigns an infinite MED value to routes that are missing the MED attribute, causing them to be the worst path alternative. The default is to assign 0 to routes with a missing MED value so that they are considered as the best alternative route.

```
sonic(config-router-bgp)# bestpath {as-path {confed | ignore | multipath-relax [as-set] |
med {confed | missing-as-worst}}
```

Graceful restart

(Optional) The graceful restart capability allows BGP peers to avoid a routing flap when a peer restarts. Both peers must have graceful restart enabled. A BGP peer maintains the routes learned from its neighbor and re-establishes the session by forwarding traffic again so that no traffic loss occurs. By default, graceful restart is disabled and a traffic loss occurs until the peer restarts. When you enable graceful restart, you can reconfigure the default restart timers:

- `restart-time seconds` — Sets the maximum time to wait for a restarting peer to resume normal operation, from 1 to 3600 seconds; default 120.

- `stalepath-time` *`seconds`* — Sets the maximum time to hold the paths learned from a restarting peer before deleting them, from 1 to 3600 seconds; default 360.
- `preserve-fw-state` — Preserves the fw state on the router after a graceful restart is performed.

```
sonic(config-router-bgp)# graceful-restart [restart-time seconds | stalepath-time seconds
| preserve-fw-state]
```

Coalesce timer

(Optional) Configure the coalesce timer (in milliseconds) used in the sub-AS group to which the router belongs, from 1 to 4294967295.

```
sonic(config-router-bgp)# coalesce-time milliseconds
```

Read and write quanta

(Optional) Configure the maximum number of packets that can be sent (`write-quanta`) or received (`read-quanta`) in a BGP peer socket during a transmission, from 1 to 10.

```
sonic(config-router-bgp)# read-quanta packets
sonic(config-router-bgp)# write-quanta packets
```

Route reflection

(Optional) You can enable the sharing of route information between members of a peer group that is configured as a BGP route reflector client. Route information received from one peer-group member is sent to all other members.

```
sonic(config-router-bgp)# client-to-client reflection
```

The peer-group is configured as a route reflector and its members/clients form a cluster. In a cluster, there may be multiple route reflectors for redundancy. Configure the cluster ID of the reflector accessed by the local router by entering its IP address in dotted-decimal format: *`A.B.C.D`*.

```
sonic(config-router-bgp)# cluster-id ip-address
```

If you configure the router as a route reflector, you can enable route policies to be applied on the outbound routes advertised in BGP updates to neighbor clients.

```
sonic(config-router-bgp)# route-reflector allow-outbound-policy
```

Deterministic MED

(Optional) You can require that the MED value in paths received from the same AS is used to select the best path. The path with the lowest MED is preferred. The default is to not require the comparison of MED attribute values for best-path selection from the paths received from an AS.

```
sonic(config-router-bgp)# deterministic med
```

Maximum MED at startup

(Optional) Configure the MED value to send in BGP packets when the router starts up. The MED value determines the path choice to an AS with multiple entry points. A lower MED value is preferred in route selection to a higher value. You can enter the time (in seconds) to use the configured MED; from 5 to 86400. Enter the maximum MED value to use, from 0 to 4294967295.

```
sonic(config-router-bgp)# max-med on-startup {time | max-med-value}
```

Disable eBGP route check

(Optional) You can disable the process that verifies the required single-hop connection with an eBGP peer. This eBGP connection verification is enabled, by default, when you configure the `ebgp-mutihop` value as 1 for eBGP neighbors.

```
sonic(config-router-bgp)# disable-ebgp-connected-route-check
```

Graceful shutdown

(Optional) Enable the graceful shutdown feature on a BGP router to avoid traffic loss when links in iBGP and eBGP sessions are shut down for maintenance. The local preference for the affected paths is lowered to minimize loss of inbound and outbound traffic during BGP reconvergence in existing sessions.

```
sonic(config-router-bgp)# graceful shutdown
```

BGP dynamic neighbors

(Optional) Enable the dynamic neighbor process to allow eBGP connections with a group of remote neighbors identified with a subnet range of IP addresses. When a BGP session starts with an IP address in the configured range, the remote peer is dynamically configured as a new member of the peer group, called a *listen range group*. The new remote peer inherits the BGP configuration of the group. No further BGP configuration is necessary. After you configure the IP address range, you can re-enter the command to configure the maximum number of listen-group members allowed in the subnet.

```
sonic(config-router-bgp)# bgp listen [limit max-number | range ip-address/prefix-length peer-
group peer-group-name]
```

Set BGP session defaults

(Optional) Configure the default settings used in BGP peer sessions when a session is established, including:

- `ipv4-unicast` — Resets the automatic exchange of IPv4 address prefixes as the default when a BGP peer session starts. The default is to advertise only IPv4 prefix routes if you previously changed the default by entering the `no default ipv4-unicast` command.
- `local-preference` *`value`* — Sets the default local preference value (0 to 4294967295; default 100). During BGP best-path selection, the local preference value is applied to the routes exchanged with a neighbor.
- `show-hostname` — Displays the host name in addition to the IP address of the local BGP router in `show` output.
- `shutdown` — Shuts down (disables) an active BGP peer session.
- `subgroup-pkt-queue-max` *`value`* — Sets the maximum number of packets allowed in a retransmit queue in a sub-AS.

```
sonic(config-router-bgp)# default {ipv4-unicast | local-preference value | show-hostname |
shutdown | subgroup-pkt-queue-max value}
```

Import routes

(Optional) Checks if a BGP network route exists in an interior gateway protocol (IGP) table before importing it into the BGP routing table.

```
sonic(config-router-bgp)# network import-check
```

Route-map filters

(Optional) You can apply a route map to filter the exchange of incoming and outgoing BGP IPv4 or IPv6 routes. Configure the time interval (in seconds) to wait before processing received filtered routes in the BGP routing table (from 0 to 600; there is no default).

```
sonic(config-router-bgp)# route-map delay-timer seconds
```

Route-update delay

(Optional) You can set the time delay (in seconds) before sending routes for best-path selection and routing updates to BGP neighbors. The best-path delay is from 0 to 3600; default 120. The route-update delay is from 1 to 3600.

```
sonic(config-router-bgp)# update-delay seconds [best-path] [route-updates]
```

Always compare MED

(Optional) Compare the MED value received from BGP neighbors during best-path selection.

```
sonic(config-router-bgp)# always-compare-med
```

Configure BGP address family

Use BGP address-family configuration mode to configure global address settings used to exchange IPv4 and IPv6 unicast routes, and L2VPN EVPN routes with a BGP neighbor. You can configure these address-family settings:

- Route redistribution policies
- Filter for downloading BGP routing table to RIB routes

- Multiple-path packet forwarding for iBGP and eBGP routes
- Backdoor routes for BGP address prefixes
- Aggregate addresses
- Administrative distance

BGP address-family configuration mode

1. Enter BGP configuration mode by entering the local AS number, from 1 to 4294967295.

```
sonic(config)# router bgp 100
```

2. Enter BGP address-family mode to configure IPv4, IPv6, or EVPN routes.

```
sonic(config-router-bgp)# address-family {ipv4 unicast | ipv6 unicast | l2vpn evpn}
sonic(config-router-bgp-af)#
```

Configure route redistribution

Configure the redistribution of BGP-learned routes — IPv4, IPv6, or L2VPN — into another routing domain table for a different L3 protocol. Redistribute routes that are statically configured or learned through BGP connections. You can apply a route-map to redistribute only the routes that match an entry in the route map.

```
sonic(config-router-bgp-af)# redistribute {static | connected | ospf} [route-map route-map-
name]
```

Filter BGP routes

Use a table map to filter the BGP routes downloaded as updates to the routing table. The downloaded BGP routes must match an entry in the specified route map.

```
sonic(config-router-bgp-af)# table-map route-map-name
```

Multipath load sharing

Configure the maximum number of eBGP and iBGP routes that can be used to forward packets over BGP using multiple paths to a neighbor:

- eBGP sessions: 1 to 256; default 1.
- iBGP sessions: 1 to 256; default 1.

```
sonic(config-router-bgp)# maximum-paths {ibgp number | ebgp number [equal-cluster-length]
```

Backdoor routes

Advertise a specified network in BGP routing updates. Configure a backdoor network prefix that is learned from BGP neighbors. The backdoor prefix provides additional routing information for iBGP sessions. Configure a route map to filter the advertised networks in BGP updates. The advertised networks must match an entry in the route map.

```
sonic(config-router-bgp-af)# network ip-prefix [backdoor] [route-map map-name]
```

Aggregate addresses

Configure an aggregate address entry in the BGP routing table. Aggregate entries reduce the size of the routing table. An aggregate prefix combines contiguous networks into one summarized set of IP addresses. Enter *ip-prefix* in the format *ip-address/mask*.

- Use the `as-set` option to advertise the aggregate routes contained in the summary aggregate-prefix entry.
- Use the `summary-only` option to suppress the advertisement of specific routes in the prefix range to neighbors.

```
aggregate-address ip-prefix [as-set] [summary-only]
```

Configure administrative distance

Routers use administrative distance to determine the best path between two or more routes to reach the same destination. Administrative distance indicates the reliability of the route; the lower the administrative distance, the more reliable the route. If the routing table receives route updates from one or more routing protocols for a single destination, it chooses the best route based on the administrative distance.

You can assign an administrative distance for external (eBGP), internal (iBGP), and local BGP routes:

- External (eBGP): From 1 to 255; default 20
- Internal (iBGP): From 1 to 255; default 200

- Local BGP routes: From 1 to 255; default 200

```
sonic(config-router-bgp-af)# distance bgp external-distance internal-distance local-distance
```

```
sonic(config)# router bgp 100
sonic(config-router-bgp)# address-family ipv4 unicast
sonic(config-router-bgp-af)# distance bgp 50 100 10
sonic(config-router-bgp-af)# exit
sonic(config-router-bgp)# address-family ipv6 unicast
sonic(config-router-bgp-af)# distance bgp 50 100 10
sonic(config-router-bgp-af)# exit
```

Route dampening

(Optional) You can enable BGP route dampening and reconfigure the default settings. Route dampening prevents the advertising of routes that continue to flap (go up and down). Each time a route flaps, it is assigned a penalty value of 100, which is added each time the route flaps. By default, BGP route dampening is disabled. When enabled, these default values are used:

- *half-life-time* — Sets the time (in minutes) to wait before decreasing the accumulated penalty for a route that no longer flaps, from 1 to 45 minutes; default 15.
- reuse-time *value* — Sets the penalty value used to unsuppress a route when it falls below the configured value, from 1 to 20000; default 750.
- suppress-route-time — Sets the maximum time (in minutes) for a flapping route to be suppressed, from 1 to 255 minutes; default 60.
- suppress-stable-route-time — Sets the maximum time (in minutes) for a stable route to be suppressed, from 1 to 255 minutes; default 60.

```
sonic(config-router-bgp)# dampening [half-life-time reuse-time suppress-route-time suppress-
stable-route-time]
```

Configure BGP neighbor

BGP neighbor configuration mode settings:

- Route redistribution policies
- Filter for downloading BGP routing table to RIB routes
- Multiple-path packet forwarding for iBGP and eBGP routes
- Backdoor routes for BGP address prefix
- Aggregate addresses
- Administrative distance

1. Enter the local AS number (1 to 65535 for a 2-byte, and 1 to 4294967295 for a 4-byte).

   ```
   sonic(config)# router bgp 100
   ```

2. Enter an IPv4 address or the local port interface number that is connected to each BGP neighbor.

   ```
   sonic(config-router-bgp)# neighbor {ip-address | ipv6-address | interface {Ethernet port-
   number | PortChannel number | Vlan vlan-id}}
   ```

3. Configure the AS number of the neighbor. Enter `internal` to configure a neighbor in the local AS to exchange routing information through internal BGP (iBGP) peer sessions. Enter `external` to configure a neighbor in an external or remote AS to exchange routing information through external BGP (eBGP) peer sessions.

   ```
   sonic(config-router-bgp-neighbor)# remote-as {peer_asn | internal | external}
   ```

4. (Optional) Enter a description of the neighbor.

   ```
   sonic(config-router-bgp-neighbor)# description text
   ```

BGP routing with the configured neighbor is enabled by default. To disable BGP routing with the neighbor, enter the `no shutdown` command.

```
sonic(config-router-bgp-neighbor)# no shutdown
```

Assign peer group

(Optional) The neighbor inherits the BGP configuration in the specified peer group (see BGP peer groups).

```
sonic(config-router-bgp-neighbor)# peer-group peer-group-name
```

eBGP multihops

(Optional) In an eBGP session, configure the maximum number of hops allowed to communicate with a peer in a remote network (1 to 255; default 1).

```
sonic(config-router-bgp-neighbor)# ebgp-multihop hop-number
```

Neighbor timers

(Optional) Configure the time (in seconds) between sending keepalive messages to a BGP neighbor (1 to 65535; default 60). Configure the hold-time to wait (in seconds) to receive a keepalive message before considering a BGP peer to be dead (3 to 65535; default 180). Enter a connect value (in seconds) for the retry timer (from 1 to 65535; default 60). The timer values that you enter override the settings that are configured for an applied peer group.

```
sonic(config-router-bgp-neighbor)# timers keepalive holdtime [connect value]
```

Update source

(Optional) Configure a BGP router from which to receive routing updates. Enter the IP address of the router.

```
sonic(config-router-bgp-neighbor)# update-source {ip-address | ipv6-address | interface
{Ethernet port-number | PortChannel number | Vlan vlan-id}
```

Configure route advertisment interval

(Optional) Configure the time (in seconds) between sending BGP route updates to neighbors (1 to 600; Defaults: eBGP 30 seconds, iBGP 5 seconds).

```
sonic(config-router-bgp-neighbor)# advertisement-interval seconds
```

Enable BFD

(Optional) Enable bi-directional forwarding detection (BFD) to detect forwarding-path failures in BGP routes. Use BFD to reduce BGP convergence time if there is a link failure.

```
sonic(config-router-bgp-neighbor)# bfd check-control-plane-failure
```

Dynamic BGP peering

(Optional) Enable dynamic BGP peering to exchange routes with remote neighbors defined by a subnet IP address. Use the `capability dynamic` command to configure dynamic BGP peering. Enter the `extended-multihop` option to accept and attempt BGP connections to BGP neighbors on remote networks that are not directly connected.

```
sonic(config-router-bgp-neighbor)# capability {dynamic | extended-multihop}
```

Disable connected checking

(Optional) Disable the automatic checking of the connected eBGP neighbor.

```
sonic(config-router-bgp-neighbor)# disable-connected-check
```

Disable multipath negotiation

(Optional) Disable automatic negotiation with the specified neighbor.

```
sonic(config-router-bgp-neighbor)# dont-capability-negotiate
```

Enforce first AS

(Optional) Deny any route updates received from the eBGP neighbor that does not have its AS number at the beginning of the AS_PATH attribute in the route update.

```
sonic(config-router-bgp-neighbor)# enforce-first-as
```

Enforce multihop

(Optional) Deny any route updates received from an eBGP neighbor that is directly connected.

```
sonic(config-router-bgp-neighbor)# enforce-multihop
```

Advertise local AS

(Optional) Advertise routes with the local AS number to the BGP peer if it is part of the local autonomous system.

```
sonic(config-router-bgp-neighbor)# local-as AS-number
```

Override capability

(Optional) Enable the capability to override the negotiated best path selection.

```
sonic(config-router-bgp-neighbor)# override-capability
```

Configure passive transport

(Optional) Configure a passive transport connection with the neighbor so that the local router does not initiate a session.

```
sonic(config-router-bgp-neighbor)# passive
```

Configure password

(Optional) Configure a password for MD5 authentication on the connection with the BGP neighbor. Enter a text string in plain text or encrypted format. If you enter an encrypted password, you must specify the encrypted option.

```
sonic(config-router-bgp-neighbor)# password text [encrypted]
```

Configure local port

(Optional) Configure the local port on which BGP routing updates are sent and received.

```
sonic(config-router-bgp-neighbor)# port number
```

Configure solo

(Optional) Configure the capability that prevents routes advertised by the specified neighbor from being reflected back to the neighbor.

```
sonic(config-router-bgp-neighbor)# solo
```

Strict capability match

(Optional) Configure the requirement that the configured local BGP capabilities must exactly match the configured BGP capabilities on the neighbor.

```
sonic(config-router-bgp-neighbor)# strict-capability-match
```

Maximum TTL security hops

(Optional) Configure the maximum number of hops between the router and the BGP neighbor (1 to 254; no default).

```
sonic(config-router-bgp-neighbor)# ttl-security hops number
```

BGP neighbor address family

Use BGP neighbor address-family configuration mode to configure global settings for the IPv4 and IPv6 unicast routes, and L2VPN EVPN routes exchanged with a BGP neighbor. You can configure various neighbor address-family settings, including:

- Advertise all paths and best path.
- Advertise ORF capabilities.
- Originate the default route.
- Use a filter list or prefix list for inbound and outbound route updates.
- Configure route reflector and client.
- Soft reconfiguration, maximum number of received prefixes, and default weight.

BGP neighbor address-family configuration mode

1. Enter BGP configuration mode by entering the local AS number, from 1 to 4294967295.

```
sonic(config)# router bgp 100
```

2. Enter a BGP neighbor with the IPv4 address or the local port interface number connected to the peer.

```
sonic(config-router-bgp)# neighbor {ip-address | port-number}
```

3. Enter BGP neighbor address-family mode by specifying the address-family routes — IPv4 unicast, IPv6 unicast, or L2VPN EVPN.

```
sonic(config-router-bgp-neighbor)# address-family {ipv4 unicast | ipv6 unicast | l2vpn
evpn}
sonic(config-router-bgp-neighbor-af)#
```

Activate route exchange

Enable route advertisement for a specified address family with a BGP neighbor. To disable address-family exchange, enter the `no activate address-family` command.

```
sonic(config-router-bgp-af)# activate
```

Allow AS path

Configure the AS number to be accepted in the route exchange with a BGP neighbor.

```
sonic(config-router-bgp-af)# allow as-in AS-count [origin]
```

- Enter an `as-count` value for the number of times that an AS number is allowed in the AS path attribute in BGP route updates.
- Enter `origin` to accept only route updates that originate from the local AS.

Advertise all paths

Enable advertisement of all routing paths to a BGP neighbor.

```
sonic(config-router-bgp-af)# addpath-tx-all-paths
```

Advertise best path

Enable the advertisement of only the best path to each AS for a BGP neighbor.

```
sonic(config-router-bgp-af)# addpath-tx-bestpath-per-AS
```

Override route exchange

Enable the override of outbound route updates to an AS path that includes the remote AS configured with the BGP neighbor `remote-as` command.

```
sonic(config-router-bgp-af)# as-override
```

Do not change BGP attribute

Do not change the AS-path, MED, or next-hop attribute value in routes received from a BGP neighbor.

```
sonic(config-router-bgp-af)# attribute-unchanged {as-path | med | next-hop}
```

Advertise ORF capability

Configure an outbound route filter (ORF) capability for sending and receiving routes from a BGP neighbor. Enter the prefix list used to filter the routes. Configure the prefix list with the `prefix-list prefix-list-name {in | out | both}` command.

```
sonic(config-router-bgp-af)# capability orf prefix-list {send | receive | both}
```

Originate default route

Configure the default route 0.0.0.0 on the local router to be used as the originating route in route advertisements sent to a BGP neighbor. Configure a route map to select, using matching route entries, the routes in which to advertise the default originating route.

```
sonic(config-router-bgp-af)# default-originate [route-map route-map-name]
```

Use filter list

Configure a BGP filter list to be used on inbound and outbound route updates. Configure the ACL using the `ip access-list` command. Enter `in` or `out` to apply the matching permit and deny entries in the ACL to inbound or outbound route advertisements.

```
sonic(config-router-bgp-af)# filter-list acl-name {in | out}
```

Use prefix list

Configure a prefix filter list to be used on inbound and outbound route updates. Configure the prefix list using the `ip prefix-list` command. Enter `in` or `out` to apply the matching permit and deny entries in the prefix list to inbound or outbound route advertisements.

```
sonic(config-router-bgp-af)# prefix-list prefix-list-name {in | out}
```

Configure next-hop self

Configure the local router to be the next-hop for outgoing routes on a BGP neighbor. Enter `force` to set the next hop to the local router for reflected routes. This command is useful in an AS subnet in which all neighbors do not speak to each other.

```
sonic(config-router-bgp-af)# next-hop-self [force]
```

Remove private AS numbers

Remove private AS numbers from advertised AS route-paths to a BGP neighbor. Private AS numbers are from 64512 to 65535.

```
sonic(config-router-bgp-af)# remove-private-AS [all] [replace-AS]
```

- Enter `all` to remove all private AS numbers in advertised AS route paths.
- Enter `replace-AS` to replace advertised AS numbers with the local AS.

Configure route reflector and client

Configure the local router as a route reflector in the AS and the BGP neighbor as a client. As a route reflector, the router passes on all BGP-learned routes to neighbor clients. Use a route reflector in iBGP sessions when all BGP routers in the AS are not fully meshed.

```
sonic(config-router-bgp-af)# route-reflector-client
```

Configure route server and client

Configure the local router as a route server in the AS and a BGP neighbor as a client. As a route server, the router sends all BGP-learned routes with a common next-hop, AS-path, and MED values to neighbor clients.

```
sonic(config-router-bgp-af)# route-server-client
```

Send communities attribute

Configure the local router to send the communities attribute value in route updates to a BGP neighbor. Specify whether to send only standard communities, only extended communities, or both. By default, a communities attribute value is not sent in route updates.

```
sonic(config-router-bgp-af)# send-community {standard | extended | both}
```

Configure soft reconfiguration

A soft BGP reconfiguration allows the router to start storing inbound route updates received from a router when a BGP session is established.

```
sonic(config-router-bgp-af)# soft-reconfiguration inbound
```

Unsuppress routes

Reconfigure the advertisement of previously suppressed routes that were aggregated into a single entry in the BGP routing table. Configure a route map to select, using matching route entries, the routes to unsuppress and advertise.

```
sonic(config-router-bgp-af)# unsurpress-map route-map-name
```

Set default weight

Configure the default weight to assign to routes learned from a BGP neighbor, from 0 to 65535; default 0. When multiple routes are available to an eBGP, the route with the highest weight is used.

```
sonic(config-router-bgp-af)# weight value
```

Set maximum prefix

Configure the maximum number of prefixes that can be received from a neighbor and stored in the BGP routing table.

- Enter a *threshold-value* to specify the percentage of the *maximum-number* when a warning message is displayed.
- Enter `warning-only` to enter a log message and not terminate the session, when the *maximum-number* is reached.
- Enter a `restart` *number* to specify after how many received prefixes, the local router restarts the session.

```
sonic(config-router-bgp-af)# maximum-prefix maximum-number {threshold-value | warning-only |
restart number}
```

Configure BGP neighbor address-family settings example

```
sonic(config)# router bgp 100
sonic(config-router-bgp)# router-id 2.2.2.2
sonic(config-router-bgp)# default ipv4-unicast
sonic(config-router-bgp)# address-family ipv4 unicast
sonic(config-router-bgp-af)# redistribute connected
sonic(config-router-bgp-af)# exit
sonic(config-router-bgp)# address-family ipv6 unicast
sonic(config-router-bgp-af)# redistribute connected
sonic(config-router-bgp-af)# exit

sonic(config-router-bgp)# neighbor 101.2.2.2
sonic(config-router-bgp-neighbor)# remote-as 100
sonic(config-router-bgp-neighbor)# address-family ipv4 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# address-family ipv6 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# exit

sonic(config-router-bgp)# neighbor 1001:2222::2
sonic(config-router-bgp-neighbor)# address-family ipv4 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# address-family ipv6 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# end
sonic#
```

View address-family settings

```
sonic# show ip bgp ipv4 neighbors

BGP neighbor is 101.2.2.2, remote AS 100, local AS 100, internal link
  BGP version 4, remote router ID 1.1.1.1 , local router ID 2.2.2.2
  BGP state = Established, up for 00:20:23
  Last read 00:00:22, Last write 00:00:22
  Hold time is 180 seconds, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    AddPath: advertised and received
    Route refresh: advertised and received
    Multiprotocol Extension: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:               2          2
    Notifications:       2          0
    Updates:             8          8
    Keepalive:           22         22
```

```
    Route Refresh:       0           0
    Capability:          0           0
    Total:               34          32

  For address family: IPv4 Unicast
    Address-family enabled
    Prefixes received 2
  For address family: IPv6 Unicast
    Address-family enabled
    Prefixes received 1
  Connections established 2, dropped 1
  Last reset 00:20:24, Last reset reason BGP Notification send
  Local host: 101.2.2.1, Local port: 55388
  Foreign host: 101.2.2.2, Foreign port: 179
  BGP Connect Retry Timer in Seconds 120

BGP neighbor is 1001:2222::2, remote AS 100, local AS 100, internal link
  BGP version 4, remote router ID 1.1.1.1 , local router ID 2.2.2.2
  BGP state = Established, up for 00:03:55
  Last read 00:00:54, Last write 00:00:54
  Hold time is 180 seconds, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    AddPath: advertised and received
    Route refresh: advertised and received
    Multiprotocol Extension: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent        Rcvd
    Opens:               3           3
    Notifications:       2           2
    Updates:             11          11
    Keepalive:           7           7
    Route Refresh:       0           0
    Capability:          0           0
    Total:               23          23

  For address family: IPv4 Unicast
    Address-family enabled
    Prefixes received 2
  For address family: IPv6 Unicast
    Address-family enabled
    Prefixes received 1
  Connections established 3, dropped 2
  Last reset 00:03:56, Last reset reason Peer closed the session
  Local host: 1001:2222::1, Local port: 40018
  Foreign host: 1001:2222::2, Foreign port: 179
  BGP Connect Retry Timer in Seconds 120
```

```
sonic# show ip bgp ipv6 neighbors

BGP neighbor is 101.2.2.2, remote AS 100, local AS 100, internal link
  BGP version 4, remote router ID 1.1.1.1 , local router ID 2.2.2.2
  BGP state = Established, up for 00:20:13
  Last read 00:00:12, Last write 00:00:12
  Hold time is 180 seconds, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    AddPath: advertised and received
    Route refresh: advertised and received
    Multiprotocol Extension: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent        Rcvd
    Opens:               2           2
    Notifications:       2           0
    Updates:             8           8
    Keepalive:           22          22
```

```
    Route Refresh:        0           0
    Capability:           0           0
    Total:                34          32

  For address family: IPv4 Unicast
    Address-family enabled
    Prefixes received 2
  For address family: IPv6 Unicast
    Address-family enabled
    Prefixes received 1
  Connections established 2, dropped 1
  Last reset 00:20:14, Last reset reason BGP Notification send
  Local host: 101.2.2.1, Local port: 55388
  Foreign host: 101.2.2.2, Foreign port: 179
  BGP Connect Retry Timer in Seconds 120

BGP neighbor is 1001:2222::2, remote AS 100, local AS 100, internal link
  BGP version 4, remote router ID 1.1.1.1 , local router ID 2.2.2.2
  BGP state = Established, up for 00:03:45
  Last read 00:00:45, Last write 00:00:45
  Hold time is 180 seconds, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    AddPath: advertised and received
    Route refresh: advertised and received
    Multiprotocol Extension: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent        Rcvd
    Opens:               3           3
   Notifications:       2           2
    Updates:             11          11
    Keepalive:           7           7
    Route Refresh:       0           0
    Capability:          0           0
    Total:               23          23

  For address family: IPv4 Unicast
    Address-family enabled
    Prefixes received 2
  For address family: IPv6 Unicast
    Address-family enabled
    Prefixes received 1
  Connections established 3, dropped 2
  Last reset 00:03:47, Last reset reason Peer closed the session
  Local host: 1001:2222::1, Local port: 40018
  Foreign host: 1001:2222::2, Foreign port: 179
  BGP Connect Retry Timer in Seconds 120
```

BGP peer groups

Instead of manually configuring each neighbor with the same BGP settings, you can create a peer-group policy and apply it to individual neighbors. BGP neighbors can inherit the same configuration.

Configure peer group

Use BGP peer-group mode to configure peer groups on the router:

1. Enter BGP configuration mode by entering the router's AS number.

```
sonic(config)# router bgp local_asn
```

2. Create a peer group by assigning a name and enter Peer-Group configuration mode.

```
sonic(config-router-bgp)# peer-group peer-group-name
```

3. Configure the AS number of the peer group. Enter `internal` to configure a peer group in the local AS to exchange routing information through internal BGP (iBGP) peer sessions. Enter `external` to configure a peer group in an external or remote AS to exchange routing information through external BGP (eBGP) peer sessions.

```
sonic(config-router-bgp-pg)# remote-as {peer_asn | internal | external}
```

4. (Optional) Enter a text description of the peer group.

```
sonic(config-router-bgp-pg)# description text
```

5. (Optional) In an eBGP session, configure the maximum number of hops allowed to communicate with a neighbor in a remote network (1 to 255; default 1).

```
sonic(config-router-bgp-pg)# ebgp-multihop hop-number
```

6. (Optional) Configure the time (in seconds) between sending keepalive messages to a BGP neighbor (1 to 65535; default 60). Configure the hold-time to wait (in seconds) to receive a keepalive message before considering a BGP neighbor to be dead (3 to 65535; default 180). Enter a connect value (in seconds) for the retry time (1 to 65535; default 30).

```
sonic(config-router-bgp-pg)# timers keepalive holdtime [connect value]
```

7. (Optional) Configure the time (in seconds) between sending BGP route updates to group members (1 to 600; eBGP 30 seconds, iBGP 5 seconds).

```
sonic(config-router-bgp-pg)# advertisement-interval seconds
```

8. Activate the peer group settings . For information about how to add BGP neighbors to the peer group using the `peer-group` command, see Configure BGP neighbors. Each neighbor that you add inherits the BGP configuration for the peer group.

```
sonic(config-router-bgp-pg)# no shutdown
```

Update source

(Optional) Enter the router IP address or interface to configure a BGP router to receive routing updates.

```
sonic(config-router-bgp-pg)# update-source {ip-address | ipv6-address | interface {Ethernet
port-number | PortChannel number | Vlan vlan-id}}
```

Enable BFD

(Optional) Enable bi-directional forwarding detection (BFD) to detect forwarding-path failures in BGP routes. Use BFD to reduce BGP convergence time in case of link failure.

```
sonic(config-router-bgp-pg)# bfd check-control-plane-failure
```

Dynamic BGP peering

(Optional) Enable dynamic BGP peering to add remote neighbors that are defined by a subnet IP address to the peer group. Enter the `extended-nexthop` option to accept and attempt BGP connections to BGP neighbors on remote networks that are not directly connected.

```
sonic(config-router-bgp-pg)# capability {dynamic | extended-nexthop}
```

Disable connected checking

(Optional) Disable the automatic checking for the next longest matching path when the primary path to a directly connected BGP neighbor goes down.

```
sonic(config-router-bgp-pg)# disable-connected-check
```

Disable multipath negotiation

(Optional) Disable the automatic negotiation of multiple paths for the same network prefix.

```
sonic(config-router-bgp-pg)# dont-capability-negotiate
```

Enforce first AS

(Optional) Deny any route updates received from an eBGP neighbor that does not have its AS number at the beginning of the AS_PATH attribute in the route update.

```
sonic(config-router-bgp-pg)# enforce-first-as
```

Enforce multihop

(Optional) Deny any route updates received from an eBGP neighbor that is directly connected.

```
sonic(config-router-bgp-pg)# enforce-multihop
```

Advertise local AS

(Optional) Advertise routes with the local AS number to only BGP neighbors that are part of the local autonomous system.

```
sonic(config-router-bgp-pg)# local-as value
```

Override capability

(Optional) Enable override of the negotiated best path selection.

```
sonic(config-router-bgp-pg)# override-capability
```

Configure passive transport

(Optional) Configure a passive transport connection with the neighbor so that the local router does not initiate a session.

```
sonic(config-router-bgp-pg)# passive
```

Configure password

(Optional) Configure a password for MD5 authentication on the connection with a BGP neighbor. Enter a text string in plain text or encrypted format. If you enter an encrypted password, you must specify the encrypted option.

```
sonic(config-router-bgp-pg)# password text [encrypted]
```

Configure local port

(Optional) Configure the local port on which BGP routing updates are sent and received.

```
sonic(config-router-bgp-pg)# port number
```

Configure solo

(Optional) Configure the capability that prevents routes advertised by a neighbor from being reflected back to the neighbor.

```
sonic(config-router-bgp-pg)# solo
```

Strict capability match

(Optional) Configure the requirement that the configured local BGP capabilities must exactly match the configured BGP capabilities on the neighbor.

```
sonic(config-router-bgp-pg)# strict-capability-match
```

Maximum TTL security hops

(Optional) Configure the maximum number of hops that can separate eBGP neighbors (1 to 254; no default).

```
sonic(config-router-bgp-pg)# ttl-security hops number
```

Configure BGP peer group example

```
sonic(config)# router bgp 100
sonic(config-router-bgp)# router-id 2.2.2.2
sonic(config-router-bgp)# default ipv4-unicast
sonic(config-router-bgp)# address-family ipv4 unicast
sonic(config-router-bgp-af)# redistribute connected
sonic(config-router-bgp-af)# exit
sonic(config-router-bgp)# address-family ipv6 unicast
```

```
sonic(config-router-bgp-af)# redistribute connected
sonic(config-router-bgp-af)# exit

sonic(config-router-bgp)# peer-group pgrp0
sonic(config-router-bgp-pg)# remote-as 100
sonic(config-router-bgp-pg)# exit

sonic(config-router-bgp)# neighbor 101.2.2.2
sonic(config-router-bgp-neighbor)# peer-group pgrp0
sonic(config-router-bgp-neighbor)# address-family ipv4 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# address-family ipv6 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# exit

sonic(config-router-bgp)# neighbor 1001:2222::2
sonic(config-router-bgp-neighbor)# peer-group pgrp0
sonic(config-router-bgp-neighbor)# address-family ipv4 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# address-family ipv6 unicast
sonic(config-router-bgp-neighbor-af)# activate
sonic(config-router-bgp-neighbor-af)# exit
sonic(config-router-bgp-neighbor)# exit
sonic(config-router-bgp)# end
sonic#
```

Display peer groups

```
sonic# show ip bgp peer-group

BGP peer-group pgrp0, remote AS 100
  Configured address-families: IPv4 Unicast;
  Peer-group members:
   1001:2222::2 Established
   101.2.2.2 Established
```

BGP peer group address family

Use BGP peer-group address-family configuration mode to configure global settings for the IPv4 and IPv6 unicast routes, and L2VPN EVPN routes exchanged with routers in a BGP peer group. You can configure various neighbor address-family settings, including:

- Advertise all paths and best path.
- Advertise ORF capabilities.
- Originate the default route.
- Use a filter list or prefix list for inbound and outbound route updates.
- Configure a route reflector and client.
- Soft reconfiguration, maximum number of received prefixes, and default weight.

BGP peer group address family configuration mode

1. Enter BGP configuration mode by entering the local AS number, from 1 to 4294967295.

   ```
   sonic(config)# router bgp 100
   ```

2. Enter the peer group name, and enter Peer-Group configuration mode.

   ```
   sonic(config-router-bgp)# peer-group peer-group-name
   ```

3. Enter BGP peer-group address-family mode by specifying the address-family of routes — IPv4 unicast, IPv6 unicast, or L2VPN EVPN.

   ```
   sonic(config-router-bgp-pg)# address-family {ipv4 unicast | ipv6 unicast | l2vpn evpn}
   sonic(config-router-bgp-pg-af)#
   ```

Activate route exchange

Enable route advertisement for a specified address family with peer-group members. To disable address-family exchange, enter the `no activate` *`address-family`* command.

```
sonic(config-router-bgp-pg-af)# activate
```

Allow AS path

Configure the AS number to be accepted in the route exchange in a BGP peer group.

```
sonic(config-router-bgp-pg-af)# allow as-in AS-count [origin]
```

- Enter an `as-count` value for the number of times that an AS number is allowed in the AS path attribute in BGP route updates.
- Enter `origin` to accept only route updates that originate from the local AS.

Advertise all paths

Enable advertisement of all BGP routes in a BGP peer group.

```
sonic(config-router-bgp-pg-af)# addpath-tx-all-paths
```

Advertise best path

Enable advertisement of only the best path to each AS in a BGP peer group.

```
sonic(config-router-bgp-pg-af)# addpath-tx-bestpath-per-AS
```

Override route exchange

Enable the override of outbound route updates to an AS path that includes the remote AS configured in a BGP peer group `remote-as` command.

```
sonic(config-router-bgp-pg-af)# as-override
```

Do not change BGP attribute

Do not change the AS-path, MED, or next-hop attribute value in routes received in a BGP peer group.

```
sonic(config-router-bgp-pg-af)# attribute-unchanged {as-path | med | next-hop}
```

Advertise ORF capability

Configure an outbound route filter (ORF) capability for sending and receiving routes in a BGP peer group. Enter the prefix list used to filter the routes. Configure the prefix list with the `prefix-list` *`prefix-list-name`* `{in | out | both}` command.

```
sonic(config-router-bgp-pg-af)# capability orf prefix-list {send | receive | both}
```

Originate default route

Configure the default route 0.0.0.0 on the local router to be used as the originating route in route advertisements sent to BGP peer-group members. Configure a route map to select, using matching route entries, the routes in which to advertise the default originating route.

```
sonic(config-router-bgp-pg-af)# default-originate [route-map route-map-name]
```

Use filter list

Configure a BGP filter list to be used on inbound and outbound route updates. Configure the ACL using the `ip access-list` command. Enter `in` or `out` to apply the matching permit and deny entries in the ACL to inbound or outbound route advertisements.

```
sonic(config-router-bgp-pg-af)# filter-list acl-name {in | out}
```

Use prefix list

Configure a prefix filter list to be used on inbound and outbound route updates. Configure the prefix list using the `ip prefix-list` command. Enter `in` or `out` to apply the matching permit and deny entries in the prefix list to inbound or outbound route advertisements.

```
sonic(config-router-bgp-pg-af)# prefix-list prefix-list-name {in | out}
```

Configure next-hop self

Configure the local router to be the next-hop for outgoing routes in a BGP peer group. Enter `force` to set the next hop to the local router for reflected routes. This command is useful in an AS subnet in which all peer-group members do not speak to each other.

```
sonic(config-router-bgp-pg-af)# next-hop-self [force]
```

Remove private AS numbers

Remove private AS numbers from advertised AS route-paths in a BGP peer group. Private AS numbers are from 64512 to 65535.

```
sonic(config-router-bgp-pg-af)# remove-private-AS [all] [replace-AS]
```

- Enter `all` to remove all private AS numbers in advertised AS route paths.
- Enter `replace-AS` to replace advertised AS numbers with the local AS.

Configure route reflector and client

Configure the local router as a route reflector in the AS and the peer-group members as clients. As a route reflector, the router passes on all BGP-learned routes to peer-group clients. Use a route reflector in iBGP sessions when all BGP routers in the AS are not fully meshed.

```
sonic(config-router-bgp-pg-af)# route-reflector-client
```

Configure route server and client

Configure the local router as a route server in the AS and the peer-group members as clients. As a route server, the router sends all BGP-learned routes with a common next-hop, AS-path, and MED values to peer-group clients.

```
sonic(config-router-bgp-pg-af)# route-server-client
```

Send communities attribute

Configure the local router to send the communities attribute value in route updates to peer-group members. Specify whether to send only standard communities, only extended communities, or both. By default, a communities attribute value is not sent in route updates.

```
sonic(config-router-bgp-pg-af)# send-community {standard | extended | both}
```

Configure soft reconfiguration

A soft BGP reconfiguration allows the router to start storing inbound route updates received from peer-group members when a BGP session is established.

```
sonic(config-router-bgp-pg-af)# soft-reconfiguration inbound
```

Unsuppress routes

Reconfigure the advertisement of previously suppressed routes that were aggregated into a single entry in the BGP routing table. Configure a route map to select, using matching route entries, the routes to unsuppress and advertise.

```
sonic(config-router-bgp-pg-af)# unsurpress-map route-map-name
```

Set default weight

Configure the default weight to assign to routes learned from peer-group members, from 0 to 65535; default 0. When multiple routes are available to an eBGP peer, the route with the highest weight is used.

```
sonic(config-router-bgp-pg-af)# weight value
```

Set maximum prefix

Configure the maximum number of prefixes that can be received from peer-group members and stored in the BGP routing table.

- Enter a *threshold-value* to specify the percentage of the *maximum-number* when a warning message is displayed.
- Enter `warning-only` to generate a log message and not terminate the session, when the *maximum-number* is reached.
- Enter a `restart` *number* to specify after how many received prefixes, the local router restarts the session.

```
sonic(config-router-bgp-pg-af)# maximum-prefix maximum-number {threshold-value | warning-only
| restart number}
```

Configure BGP peer-group address-family example

```
sonic(config-router-bgp)# peer-group pgrp0
sonic(config-router-bgp-pg)# remote-as 100
sonic(config-router-bgp-pg)# address-family ipv4 unicast
sonic(config-router-bgp-pg-af)# activate
sonic(config-router-bgp-pg-af)# capability orf prefix-list send
sonic(config-router-bgp-pg)# exit
```

View BGP peer-group address family

```
sonic# show ip bgp ipv4 neighbors

BGP neighbor is 101.2.2.2, remote AS 100, local AS 100, internal link
  BGP version 4, remote router ID 1.1.1.1 , local router ID 2.2.2.2
  BGP state = Established, up for 00:02:34
  Last read 00:00:32, Last write 00:00:32
  Hold time is 180 seconds, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    AddPath: advertised and received
    Route refresh: advertised and received
    Multiprotocol Extension: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:               2          2
    Notifications:       2          0
    Updates:             10         14
    Keepalive:           44         44
    Route Refresh:       0          0
    Capability:          0          0
    Total:               58         60

  For address family: IPv4 Unicast
  Address-family enabled
    Prefixes received 3
  For address family: IPv6 Unicast
  Address-family enabled
    Prefixes received 2
  Connections established 2, dropped 1
  Last reset 00:02:34, Last reset reason BGP Notification send
  Local host: 101.2.2.1, Local port: 35616
  Foreign host: 101.2.2.2, Foreign port: 179
  BGP Connect Retry Timer in Seconds 120
  ...
```

BGP routing policy filters

When you configure connections with BGP neighbors in external networks (remote AS), create routing policies to filter the routes in inbound and outbound route advertisements.

- To filter the prefixes in route advertisements, use IP prefix lists, route maps, autonomous-system-path access lists, and filter lists. Inbound and outbound route updates are managed according to the permit and deny statements they match.
- To filter route advertisements, you can also configure BGP routers to process the Communities attribute. Routes that belong to the same community receive common treatment.

IP prefix lists

To configure IP prefix filtering in BGP route advertisements, create a prefix list with permit and deny statements for matching network prefixes. Enter the prefix list in BGP neighbor and BGP peer-group address family configurations to filter the inbound and outbound route advertisements using the `filter-list list {in | out}` command.

```
sonic(config)# ip prefix-list {deny | permit} ip-address/prefix-length [ge ge-min-prefix-length] [le le-max-prefix-length]
```

```
sonic(config)# ipv6 prefix-list {deny | permit} ipv6-prefix/prefix-length [ge ge-min-prefix-length] [le le-max-prefix-length]
```

- Enter `permit` or `deny` for the action to take on a matching network prefix.
- Enter an IPv4 network prefix by specifying an IP address and bit mask from 1 to 32. Enter an IPv6 address network prefix by specifying an IPv6 prefix and prefix length address and bit mask.
- (Optional) Specify a range of addresses to use by entering the greater than or equal to (*ge-min-prefix-length*) value or the less than or equal to (*le-min-prefix-length*) value.
-

Configure IP prefix list example

```
sonic(config)# ip prefix-list prflst656 permit 156.1.1.0/24
sonic(config)# ip prefix-list prflst657 permit 157.1.1.0/24
sonic(config)# ipv6 prefix-list prflst758 permit 1758:5523::/64 ge 67 le 68
sonic(config)# ipv6 prefix-list prflst759 permit 1759:5567::/64
sonic(config)# end
```

```
sonic# show ip prefix-list
IP prefix list prflst656:
     permit 156.1.1.0/24
IP prefix list prflst657:
     permit 157.1.1.0/24

sonic# show ip prefix-list prflst657
IP prefix list prflst657:
     permit 157.1.1.0/24
```

```
sonic# show ipv6 prefix-list
IPv6 prefix list prflst758:
     permit 1758:5523::/64 ge 67 le 68
IPv6 prefix list prflst759:
     permit 1759:5567::/64

sonic# show ipv6 prefix-list prflst758
IPv6 prefix list prflst758:
     permit 1758:5523::/64 ge 67 le 68
```

BGP communities

By default, the Communities attribute is not sent in BGP route updates. You must configure the router to process the Communities attribute in BGP routes. You must also configure the community names to which routes belong or use the predefined, well-known community names, such as `no-advertise` and `no-export`.

Configure a list of BGP community values to use in the permit/deny statements in route maps. Apply the route maps to a BGP neighbor or BGP peer-group address-family. Enable BGP community exchange with specified neighbors using the `send-community` command in neighbor configuration mode.

Configure standard community list

```
sonic(config)# bgp community-list standard community-name [community-number] [local-AS] [no-advertise] [no-export] [no-peer] [any | all]
```

- Enter a *community-number* in the format *AS:NN*, where valid AS numbers are 0 to 65535 and valid network numbers are 0 to 65535.

- Enter other community attributes to match in route maps, such as `local-AS`, `no-advertise`, `no-export`, and `no-peer`.
- Enter `any` or `all` to specify whether at least one (any) or all of the configured community attributes must match for a route update to be permitted or denied. The default is `any`.
-

For example to configure a standard community list that matches either the `no-advertise` or `no-export` route attributes:

```
sonic(config)# bgp community-list standard comm1 no-advertise no-export any
```

For example, to configure a standard community list that requires that all attributes match routes from network 10 in AS 1000 and routes not received from a peer:

```
sonic(config)# bgp community-list standard comm2 1000:10 no-peer
```

Configure expanded community list

```
bgp community-list expanded community-name regular-expression [any | all]
```

- Enter the *regular-expression* as an ordered list. The regular expression is used to filter communities by matching it with the community attribute value.
- Re-enter the command to configure additional expanded community lists.

For example, in an expanded community list, match routes from networks 1 to 10 in AS 1000:

```
sonic(config)# bgp community-list expanded comm3 1000:[1-10]
```

Configure an extended community list

Use extended community lists to filter the routes in VRF instances. In an extended community, the route target (RT) determines the VRFs and neighbors that receive the routes. The site of origin (SOO) prevents routes learned from a neighbor from being advertised back to the originating router.

```
sonic(config)# bgp extcommunity-list standard extended-community-attribute [rt value] [soo
value] [any | all]
```

- Enter the *extended-community-attribute* in *AA:NN* format, where *AA* is the local-AS number, and *NN* is the network number. Both values are from 0 to 65535.
- Enter the route target in `rt AA:NN` or `rt ip-address:NN` format.
- Enter the site of origin in `soo AA:NN` or `soo ip-address:NN` format.
- Re-enter the command to configure additional standard extended community lists.

To configure an extended expanded community list:

```
sonic(config)# bgp extcommunity-list expanded regular-expression [any | all]
```

- Enter the *regular-expression* as an ordered list. The regular expression is used to filter communities by matching it with the community attribute value.
- Re-enter the command to configure additional expanded extended community lists.

For example, in an extended standard community list, match the routes for both the route target network 10 in AS 64545, and the site-of-origin network 11 in AS 64555:

```
sonic(config)# bgp extcommunity-list standard 2 64545:10 64555:11 all
```

For example, in an extended expanded community list, match the routes for any of the configured *AA:NN* expressions:

```
sonic(config)# bgp extcommunity-list expanded 102:2
sonic(config)# bgp extcommunity-list expanded 103:3
sonic(config)# bgp extcommunity-list expanded 104:4
```

AS path lists

Create filters for the AS paths in route advertisements using regular expressions as match criteria. Apply an AS list filter to the inbound and outbound address families of routes that are advertised to, and received from, BGP neighbors and BGP peer groups using the `filter-list list {in | out}` command.

```
sonic(config)# bgp as-path-list AS-pathlist-name regular-expression
```

- Enter the *regular-expression* as an ordered list in the format *AA:NN*. The regular expression is used to filter routes by matching the AS path in a route as an ASCII string.
- Re-enter the command to configure additional AS path lists or add matching regular expressions to an AS list.

Configure AS path list example

```
sonic(config)# bgp as-path-list expanded ASlist1 1002:202
sonic(config)# bgp as-path-list expanded ASlist1 2002:203
```

Route maps

Create route maps to filter routes using permit and deny statements. Configure match statements to select routes; configure set statements to modify the BGP attributes in a matching route.

Apply a route-map filter to the inbound and outbound address families of routes that are advertised to, and received from, BGP neighbors and BGP peer groups using the `route-map map-name {in | out}` command.

1. Create a route map to match the route parameters listed in the next step. Specify a permit or deny statement to configure how matching routes are handled. Enter the sequence number for the order in which the statement is processed in the map.

   ```
   sonic(config)# route-map map-name {permit | deny} sequence-number
   ```

2. In route-map configuration mode, enter any of these match statements to select routes.

   ```
   sonic(config-route-map)# match as-path acl-name
   sonic(config-route-map)# match community community-list-name
   sonic(config-route-map)# match ext-community extcommunity-list-name
   sonic(config-route-map)# match interface interface
   sonic(config-route-map)# match ip address prefix-list prefix-list-name
   sonic(config-route-map)# match ipv6 address prefix-list prefix-list-name
   sonic(config-route-map)# match metric value
   sonic(config-route-map)# match route-type {internal | external}
   sonic(config-route-map)# match origin {egp | igp | incomplete}
   sonic(config-route-map)# match tag value
   sonic(config-route-map)# match local-preference value
   sonic(config-route-map)# match peer ip-address
   sonic(config-route-map)# match ip next-hop prefix-list prefix-list-name
   sonic(config-route-map)# call route-map-name
   sonic(config-route-map)# match source-protocol {bgp | ospf | ospf3 | static | connected}
   ```

3. In route-map configuration mode, enter any of these set statements to change the specified BGP attribute in matching routes.

   ```
   sonic(config-route-map)# set as-path prepend list
   sonic(config-route-map)# set comm-list comm-list-name {add | del}
   sonic(config-route-map)# set community options
   sonic(config-route-map)# set ext-community options
   sonic(config-route-map)# set ip next-hop number
   sonic(config-route-map)# set ipv6 next-hop number
   sonic(config-route-map)# set local-preference value
   sonic(config-route-map)# set metric value
   sonic(config-route-map)# set origin {igp | egp | incomplete}
   sonic(config-route-map)# set tag value
   ```

Configure route map example

```
sonic(config)# route-map map1 permit 10
sonic(config-route-map)#  match as-path ASlist1
sonic(config-route-map)#  match ext-community comm3
sonic(config-route-map)#  match interface Ethernet4
sonic(config-route-map)#  set metric 100
```

```
sonic(config-route-map)#  set origin egp
sonic(config-route-map)#  set local-preference 10000
```

To remove a configured value in a route map entry, enter the no version of the `match` or `set` command; for example:

```
sonic(config)# route-map map1 permit 10
sonic(config-route-map)#  no match as-path
sonic(config-route-map)#  no set origin
```

Unnumbered BGP

BGP uses TCP for connections with neighbor devices. A router interface that connects to a neighbor requires a unique IP address. Assigning an IP address to each router interface may exhaust the available pool of IP addresses, resulting in an error in operation.

An unnumbered interface does not have a user-configured IP address. BGP unnumbered interfaces use the extended next-hop encoding (ENHE) feature, as defined in RFC 5549, to advertise IPv4 routes with an IPv6 next hop.

After you enable IPv6 on an interface that is connected to a BGP neighbor, an IPv6 link-local address is automatically created. BGP uses the link-local address to set up a BGP session with the neighbor. Unnumbered interfaces use IPv6 router advertisements (RAs) to identify the address of a BGP neighbor.

Using unnumbered BGP, hosts and switches automatically discover neighboring routers. Peer routers that are connected with point-to-point links are discovered by parsing their router advertisements.

Each router periodically generates an RA, which contains its MAC address and link-local address. If you configure an unnumbered interface on a BGP-enabled router, the interface parses the RA information it receives from a peer device and sets up a BGP session with the device.

Configure unnumbered BGP

1. Use the SONiC CLI to enable IPv6 on an interface; for example:

   ```
   admin@sonic:~$ sudo config interface ipv6 enable use-link-local-only Ethernet 0
   ```

2. Use Management Framework CLI commands to configure unnumbered BGP on the interface.

   ```
   sonic(config)# interface Loopback 0
   sonic(conf-if-lo0)# ip address 10.0.1.1/32
   sonic(conf-if-lo0)# end
   sonic# configure terminal
   sonic(config)# router bgp 64601
   sonic(config-router-bgp)# router-id 10.0.1.1
   sonic(config-router-bgp)# neighbor interface Ethernet 0
   sonic(config-router-bgp-neighbor)# remote-as external
   sonic(config-router-bgp-neighbor)# capability extended-nexthop
   sonic(config-router-bgp-neighbor)# no shutdown
   sonic(config-router-bgp-neighbor)# address-family ipv4 unicast
   sonic(config-router-bgp-neighbor-af)# activate
   sonic(config-router-bgp-neighbor-af)# exit
   sonic(config-router-bgp-neighbor)# exit
   sonic(config-router-bgp)# end
   ```

View BGP configuration

You can view BGP configuration and operation using `show` commands.

View BGP routes

```
sonic# show ip bgp [vrf vrf-name] {ipv4 | ipv6}
```

```
sonic# show ip bgp
BGP routing table information for VRF default
Router identifier 20.0.0.1, local AS number 100
Route status codes: * - valid, > - active, e - ECMP
Origin codes: i - IGP, e - EGP, ? - incomplete
AS Path Attributes: Or-ID - Originator ID, C-LST - Cluster List,
LL Nexthop - Link Local Nexthop
```

```
       Network      Next Hop  Metric   LocPref   Weight   Path
* >    3.0.0.1/32   1.1.1.2   0        100       0        300 i
*      3.0.0.1/32   1.0.0.2   0        100       0        200 ?
* >    3.0.0.2/32   1.1.1.2   0        100       0        300 i
*      3.0.0.2/32   1.0.0.2   0        100       0        200 ?
* >    3.0.0.3/32   1.1.1.2   0        100       0        300 i
```

```
sonic# show ip bgp [vrf vrf-name] {ipv4 | ipv6} prefix
```

```
sonic# show ip bgp 30.0.0.0/24

BGP routing table entry for 30.0.0.0/24, version 35
Paths: (3 available, best #2, table default)
Multipath: eBGP
Flag: 0x860
  Advertised to update-groups:
     1
  200
    50.0.0.1 from 50.0.0.1 (20.0.0.1)
      Origin incomplete, localpref 100, valid, external, backup/repair
      Only allowed to recurse through connected route
  200
    60.0.0.1 from 60.0.0.1 (20.0.0.1)
      Origin incomplete, localpref 100, weight 100, valid, external, best
      Only allowed to recurse through connected route
  200
    70.0.0.1 from 70.0.0.1 (40.0.0.1)
      Origin incomplete, localpref 100, valid, external,
      Only allowed to recurse through connected route
```

View BGP neighbors

```
sonic# show ip bgp [vrf vrf-name] {ipv4 | ipv6} summary
```

```
sonic#show ip bgp summary
BGP summary information for VRF default
Router identifier 20.0.0.1, local AS number 100
Neighbor   V AS  MsgRcvd MsgSent InQ OutQ Up/Down  State  PfxRcd PfxAcc
10.1.0.100 4 200 1075    1083    0    0   00:04:04 Connect
10.2.0.101 4 200 1079    1088    0    0   00:04:14 Connect
```

```
sonic# show ip bgp [vrf vrf-name] neighbors [nbr-ip] {routers | received-routes | advertised-
routes}
```

```
sonic# show ip bgp neighbors 10.3.0.103 advertised-routes
BGP routing table information for VRF default
Router identifier 10.0.0.102, local AS number 64500
Route status codes: s - suppressed, * - valid, > - active, # - not installed,
                    E - ECMP head, e - ECMP, S - Stale,
                    c - Contributing to ECMP, b - backup, L - labeled-unicast
Origin codes: i - IGP, e - EGP, ? - incomplete
AS Path Attributes: Or-ID - Originator ID, C-LST - Cluster List,
                    LL Nexthop - Link Local Nexthop

      Network          Next Hop   Metric  LocPref Weight Path
* >   10.1.0.0/24      10.3.0.102   -      100     -      i
* >   10.2.0.0/24      10.3.0.102   -      100     -      i
* >   10.3.0.0/24      10.3.0.102   -      100     -      i
* >   10.100.0.0/24    10.1.0.100   200    100     -    64496 i
* >   10.100.1.0/24    10.1.0.100   -      100     -    64496 64497 65536 i
* >   10.100.2.0/24    10.1.0.100   42     100     -    64496 ?
* >   10.101.0.0/24    10.2.0.101   -      100     -    64510 i
* >   10.101.1.0/24    10.2.0.101   -      100     -    64510 i
* >   10.101.2.0/24    10.2.0.101   -      100     -    64510 i
```

View BGP peer groups

```
sonic# show ip bgp [vrf vrf-name] peer-group [peer-group-name]
```

```
sonic# show ip bgp peer-group
BGP peer-group is EXTERNAL
  BGP version 4
  Static peer-group members:
    VRF default:
      10.1.0.100, state: Connect
        Negotiated MP Capabilities:
            IPv4 Unicast: No
            IPv6 Unicast: No
      10.2.0.101, state: Connect
        Negotiated MP Capabilities:
            IPv4 Unicast: No
            IPv6 Unicast: No

BGP peer-group is INTERNAL
  BGP version 4
  Listen-range subnets:
    VRF default:
      10.3.0.0/24, remote AS 64500
  Dynamic peer-group members:
    VRF default:
```

View BGP route maps

```
sonic# show route-map
Route map map1:
   permit, sequence 10
    Match clauses:
    Set clauses:
      local preference 10
    Call clauses:
    Actions:
      Exit routemap
Route map map2:
   permit, sequence 2
    Match clauses:
      med 10
    Set clauses:
    Call clauses:
    Actions:
      Exit routemap
```

View BGP prefix lists

```
sonic# show ip prefix-list
IP prefix list pref1:
     permit 20.0.0.0/8
```

```
sonic# show ipv6 prefix-list
IPv6 prefix list pref2:
     permit 2222::/64 ge 65 le 65
     permit 2223::/64 ge 65 le 128
```

View BGP community lists

```
sonic# show bgp community-list
Standard community list com1:  match: ANY
     local-AS
Expanded community list com2:   match: ANY
     Extended1
```

```
sonic# show bgp ext-community-list
Standard extended community list extcom1:  match: ANY
     rt:2:2
```

```
Expanded extended community list extcom2:  match: ANY
     extcom
```

View AS path lists maps

```
sonic# show bgp as-path-access-list
AS path list aspath1:
   members: 1:1+
```

17

Virtual routing and forwarding

Virtual routing and forwarding (VRF) partitions a physical router into multiple virtual routers. The control and data plane are isolated in each virtual router; traffic does not flow across virtual routers. VRF allows multiple instances of independent routing tables to co-exist within the same router simultaneously.

SONiC supports a management VRF instance, a default VRF instance, and nondefault VRF instances. Use the default and nondefault VRF instances to configure routing.

By default, the system initially assigns all physical interfaces and all logical interfaces to the default VRF instance. You can move the management interface from the default to management VRF instance. You must create the management VRF instance because it does not exist in the system by default.

In VRF configuration mode, you can configure various features for a specific VRF instance. This table shows the features and services that are supported in the management, default, and nondefault VRF instances.

Table 3. Supported VRF features

Feature	Management VRF	Default VRF	Nondefault VRF
BGP	No	Yes	Yes
ICMP/ping	Yes	Yes	Yes
NTP client	Yes	Yes	Yes
NTP server	Yes	Yes	Yes
SCP client	Yes	Yes	Yes
Traceroute	Yes	Yes	Yes

Topics:

Management VRF

The default VRF routing table is used by default on all data-plane switch ports. Although the management interface is assigned by default to the default VRF instance, you can move it to its own VRF instance.

Create management VRF

The management VRF provides a separate `mgmt` routing table for an out-of-band management network that is different from the default in-band data-plane network. A dedicated management VRF provides an isolated routing table for the management interface when accessing the switch using SSH. The management VRF allows you to create a separate IP network that is "always on" for switch management.

```
sonic(config)# ip vrf management
```

- NTP uses the management VRF by default; if the management VRF is not configured, NTP uses the default VRF.
- The management VRF does not support IP services, such as DNS, TFTP, and FTP.
- Use `no ip vrf management` to delete the management VRF.
- SONiC does not support the use of a static IP or IPv6 route to direct incoming traffic in a remote management session, such as SSH, to the management interface.

View management VRF

```
sonic#show ip vrf management
VRF-Name                  Interfaces
```

```
-----------------------------------
management                  eth0
```

Configure nondefault VRF instances

Besides a management VRF and the default VRF instance, you can configure multiple instances of routing and forwarding tables on the switch. Multiple VRF instances are necessary when you must maintain separate traffic for different customers or different departments in a large enterprise.

When you create a nondefault VRF, you assign L3 interfaces to the VRF. This is called binding a L3 interface to a VRF. You can bind physical Ethernet, loopback, port channel, and VLAN interfaces to a nondefault VRF. You can create a maximum of 128 nondefault VRF instances.

Create VRF

```
sonic(config)# ip vrf Vrf_red
sonic(conf-vrf)#
```

- *vrf-name* is up to 32 alphanumeric characters
- Use `no ip vrf vrf-name` to delete a VRF.

Bind L3 interface to VRF

```
sonic(config)# interface Ethernet 12
sonic(conf-if-Ethernet12)# ip vrf forwarding Vrf_red
sonic(conf-if-Ethernet12)#
sonic(conf-if-Ethernet12)#
sonic(conf-if-Ethernet12)# exit
sonic(config)#
sonic(config)# interface Ethernet 16
sonic(conf-if-Ethernet16)# ip vrf forwarding Vrf_1
%Error: No instance found for 'Vrf_1'
sonic(conf-if-Ethernet16)#
```

Unbind L3 interface to VRF

```
sonic(config)# interface Ethernet 100
sonic(conf-if-Ethernet100)# no ip vrf forwarding Vrf_red
Success
```

View VRF configuration

```
sonic# show ip vrf
VRF-NAME            INTERFACES
------------------------------
mgmt                eth0
Vrf_blue            Ethernet8
                    Loopback20
                    PortChannel20
                    Vlan20
Vrf_red             Ethernet4
                    Loopback10
                    PortChannel10
                    Vlan10
```

```
sonic# show ip vrf management
VRF-Name            Interfaces
------------------------------
mgmt                eth0
```

```
sonic# show ip vrf Vrf_blue
VRF-NAME            INTERFACES
------------------------------
Vrf_blue            Ethernet8
                    Loopback20
                    PortChannel20
                    Vlan20
```

```
sonic# show ip vrf Vrf_red
VRF-NAME           INTERFACES
-----------------------------
Vrf_red            Ethernet4
                   Loopback10
                   PortChannel0
                   Vlan10
```

18

Network Address Translation

Network address translation (NAT) enables the process that assigns a public IP address to devices that access resources outside the network. NAT conserves IP address usage in the local network.

NAT is not required within the network to route traffic between private IP addresses. A NAT gateway translates the private IP addresses of local network devices to a globally unique, public IP address when they communicate with remote devices.

Topics:

Enable NAT

To enable NAT and enter NAT configuration commands in NAT Configuration mode:

```
sonic(config)# nat
sonic(config-nat)# enable
sonic(config-nat)#
```

To disable NAT, use `no enable`.

NAT configuration

After you enable NAT, perform these steps in any order.

Add static NAT entry

To communicate outside your network over the Internet, you can manually configure a static NAT entry to replace a local IP address with a globally unique IP address.

```
sonic(config-nat)# static basic global-ip local-ip [snat | dnat] [twice_nat_id value]
```

- Source NAT (`snat`) translates a source IP address in the local network to a global IP address sent to an external network. Remote devices in outside networks use the global address to access the local device.
- Destination NAT (`dnat`) translates a destination IP address in packets, which are received from an external network and traverse the local network, into a local IP address used in the local network. `dnat` is the default.
- `twice_nat_id` *`value`* performs address translation on both source and destination IP addresses for static entries which have the same *`value`* .
- To remove a static NAT entry, use `no static basic` *`global-ip local-ip`*.

```
sonic(config-nat)# static basic 125.4.4.4 12.1.1.1
sonic(config-nat)# static basic 100.100.100.100 15.15.15.15 snat twice-nat-id 5
sonic(config-nat)# static basic 200.200.200.5 17.17.17.17 dnat twice-nat-id 5
```

Add static NAT entry with L4 port

You can also configure a static NAT entry to translate a local IP address and TCP or UDP port number into a global IP addresses with TCP or UDP port number.

```
sonic(config-nat)# static {tcp | udp} global-ip global-port local-ip local-port
[snat | dnat] [twice_nat_id value]
```

- Source NAT (`snat`) translates a source IP address and TCP/UDP port in the local network into a global IP address and TCP/UDP port sent to an external network. Remote devices in outside networks use the global address and L4 port to access the local device.

- Destination NAT (`dnat`) translates a destination IP address and TCP/UDP port in packets, which are received from an external network and traverse the local network, into a local IP address and TCP/UD port used in the local network. `dnat` is the default.
- `twice_nat_id value` performs address translation on both source and destination IP addresses for static entries which have the same ID *value* .

```
sonic(config-nat)# static udp 148.56.7.7 8991 10.11.1.12 2000
sonic(config-nat)# static tcp 123.3.4.1 901 11.11.1.1 1000
sonic(config-nat)# static tcp 65.55.46.6 106 20.0.0.6 206 dnat twice-nat-id 200
sonic(config-nat)# static tcp 65.55.45.5 100 20.0.0.5 200 snat twice-nat-id 200
```

Create NAT address pool

If you configure dynamic NAT, a local address is replaced using a pool of global addresses. Dynamic translation is useful when multiple users on a private network access the Internet. Configure a pool of available global addresses by defining the global IP address range, and optionally the TCP/UDP port range used for local address translation.

```
pool pool-name global-ip-range [global-port-range]
```

- Enter the global IP address range in the format *ipaddress-ipaddress*. Valid values are from `65.55.45.10` to `65.55.45.15`.
- Enter the TCP/UDP port-number range in the format *portnumber-portnumber*. Valid values are from `1024` to `65535`.
- To delete a NAT address pool, use the `no pool pool-name` command. To delete all NAT address pools, use `no pools`.

```
sonic(config-nat)# pool Pool1 19.19.19.19
sonic(config-nat)# pool Pool2 20.0.0.7 1024-65535
sonic(config-nat)# pool Pool3 65.55.45.10-65.55.45.15 500-1000
```

Use ACLs to configure NAT binding

You can use an access control list (ACL) to determine the IP addresses in a global NAT address pool. By default, if you specify an ACL, traffic from all IP hosts is allowed. A permit statement allows an IP address. A deny statement denies the address. Using the address pool:

- `snat` translates a source IP address to a global IP address in the pool.
- `dnat` translates a destination IP address to a global IP address in the pool. `dnat` is the default.
- `twice_nat_id value` performs address translation on both source, and destination IP addresses using the address pool for static entries which have the same ID *value*.

```
binding binding-name pool-name [acl-name]
[snat | dnat] [twice_nat_id value]
```

```
sonic(config-nat)# binding Bind1 Pool1 10_ACL_IPV4
sonic(config-nat)# binding Bind2 Pool2 12_ACL_IPV4 snat twice-nat-id 25
sonic(config-nat)# binding Bind3 Pool3 15_ACL_IPV4 dnat twice-nat-id 25
```

To remove the ACL-pool binding, enter `no binding binding-name`. To remove all ACL-pool bindings, enter `no bindings`.

Configure NAT zones

Configure a NAT zone on L3 interfaces so that NAT address translation is performed on packets when a packet traverses a zone on configured interfaces. You can configure a NAT zone on any Ethernet, VLAN, port channel, or loopback interface that is configured with an IP address. The range of NAT zone numbers is from 0 to 3.

```
sonic(config)# interface Ethernet 4
sonic(conf-if-Vlan5)# ip address 20.20.20.20/24
sonic(conf-if-Ethernet4)# nat-zone 1
sonic(conf-if-Ethernet4)# exit
sonic(config)# interface Vlan 5
sonic(conf-if-Vlan5)# ip address 23.23.23.23/24
sonic(conf-if-Vlan5)# nat-zone 1
sonic(conf-if-Vlan5)# exit
sonic(config)# interface PortChannel 2
sonic(conf-if-po2)# ip address 25.25.25.25/24
sonic(conf-if-po2)# nat-zone 1
sonic(conf-if-po2)# exit
sonic(config)# interface Loopback 1
sonic(conf-if-lo1)# ip address 10.10.10.10/32
sonic(conf-if-lo1)# nat-zone 2
```

The source zone of a packet is the zone of the inbound interface on which the packet is received. The destination zone of the packet is the zone of the L3 next-hop interface from the L3 route lookup of the destination address.

- Inbound traffic entering on a source interface is L3 forwarded using static DNAT translation.
- Outbound traffic being sent on a destination interface is dynamically SNAT translated.

To remove the NAT configuration on all interfaces, enter `no nat interfaces`.

Configure dynamic NAT timeout

By default, only dynamic address translation configurations time out after 10 minutes (600 seconds). There is no timeout for static NAT entries. To change the dynamic NAT timeout value, specify a new value in seconds (300 to 432000).

```
sonic (config)# nat
sonic (config-nat)# timeout 1200
```

You can also change the NAT entry timeouts for L4 UDP and TCP NAT entries. To change the TCP timeout for address translation, enter a new timeout value in seconds (300 to 432000; default 86400). To change the UDP timeout for address translation, enter a new value in seconds (120 to 600; default 300).

```
sonic (config)# nat
sonic (config-nat)# udp-timeout 600
sonic (config-nat)# tcp-timeout 66460
```

View NAT configuration

Use these commands to display the NAT configuration and operation, and NAT table entries.

View NAT translation table

```
sonic# show nat translations

                                        Translated       Translated
Protocol Source         Destination     Source           Destination
-------- ------         -----------     ----------       -----------
all      10.0.0.1       ---             65.55.42.2       ---
all      ---            65.55.42.2      ---              10.0.0.1
all      10.0.0.2       ---             65.55.42.3       ---
all      ---            65.55.42.3      ---              10.0.0.2
tcp      20.0.0.1:4500  ---             65.55.42.1:2000  ---
tcp      ---            65.55.42.1:2000 ---              20.0.0.1:4500
udp      20.0.0.1:4000  ---             65.55.42.1:1030  ---
udp      ---            65.55.42.1:1030 ---              20.0.0.1:4000
tcp      20.0.0.1:6000  ---             65.55.42.1:1024  ---
tcp      ---            65.55.42.1:1024 ---              20.0.0.1:6000
tcp      20.0.0.1:5000  65.55.42.1:2000 65.55.42.1:1025  20.0.0.1:4500
tcp      20.0.0.1:4500  65.55.42.1:1025 65.55.42.1:2000  20.0.0.1:5000
```

To clear the entries in the NAT translation table, enter `clear nat translations`.

View NAT translation statistics

```
sonic# show nat statistics

Protocol Source              Destination         Packets   Bytes
-------- ---------           -----------         -------   -------
all      100.100.100.100     200.200.200.5       15        12785
all      17.17.17.17         15.15.15.15         10        12754
all      12.12.12.14         ---                 0         0
all      ---                 138.76.28.1         12        12500
tcp      12.12.15.15:1200    ---                 0         0
tcp      ---                 138.76.29.2:250     8         85120
tcp      100.100.101.101:251 200.200.201.6:276   21        21654
tcp      17.17.18.18:1251    15.15.16.16:1201    18        21765
```

To clear the NAT statistics, enter `clear nat statistics`.

View NAT and NAPT configuration

```
sonic# show nat config static

                                Global L4  Local IP    Local L4  Twice-Nat
Nat Type  IP Protocol Global IP   Port       Port        Port      Id
--------  ----------- ----------  ---------  ----------  --------  ---------
dnat      all         65.55.45.5  ---        10.0.0.1    ---       ---
dnat      all         65.55.45.6  ---        10.0.0.2    ---       ---
dnat      tcp         65.55.45.7  2000       20.0.0.1    4500       1
snat      tcp         20.0.0.2    4000       65.55.45.8  1030       1
```

View NAT pools

```
sonic# show nat config pool

Pool Name      Global IP Range             Global L4 Port Range
------------   -------------------------   --------------------
Pool1          65.55.45.5                  1024-65535
Pool2          65.55.45.6-65.55.45.8       ---
Pool3          65.55.45.10-65.55.45.15     500-1000
```

View NAT binding configuration

```
sonic# show nat config bindings

Binding Name   Pool Name      Access-List    Nat Type  Twice-Nat Id
------------   ------------   ------------   --------  ------------
Bind1          Pool1          ---            snat      ---
Bind2          Pool2          1              snat      1
Bind3          Pool3          2              snat      --
```

View global NAT configuration

```
sonic# show nat config globalvalues

Admin Mode     : enabled
Global Timeout : 600 secs
TCP Timeout    : 86400 secs
UDP Timeout    : 300 secs
```

View L3 interface zones

```
sonic# show nat config zones
Port                Zone
-------------------------------------------------
Ethernet0           1
Loopback0           1
Vlan5               0
PortChannel12       2
```

View NAT translation counts

```
sonic# show nat translations count

Static NAT Entries        ..................  4
Static NAPT Entries       ..................  2
Dynamic NAT Entries       ..................  0
Dynamic NAPT Entries      ..................  4
Static Twice NAT Entries  ..................  0
Static Twice NAPT Entries ..................  4
Dynamic Twice NAT Entries  .................  0
Dynamic Twice NAPT Entries .................  0
Total SNAT/SNAPT Entries   .................  9
Total DNAT/DNAPT Entries   .................  9
Total Entries              ................. 14
```

19

Access control lists

Access control lists (ACLs) filter inbound and outbound traffic passing through the switch using match criteria. For example, you can specify source IP addresses to permit or deny packets, and add upper-level TCP or UDP protocol rules for additional filtering.

ACLs are identified by name, and ACLs process in sequence. If a packet does not match the criterion in the first filter statement, the second filter applies. You can also use ACLs with other features, such as quality of service (QoS), routing table updates, and network address translation (NAT).

Use ACLs to filter traffic

1. Configure an IP access list with permit and deny match criteria.
2. Apply the ACL to an inbound or outbound L3 IP interface.

Topics:

Create ACL

Before you start

- By default, an ACL denies all traffic. If a packet does not match any of the permit criteria, the packet is dropped.
- An ACL filters packets using the permit and deny criteria in the order in which you enter them. The order is determined by the sequence number that is assigned to each permit/deny rule. The ACL stops filtering packet criteria when a permit/deny match is found.
- Add a `permit any any` rule to the end of an ACL to permit all packets that are not denied by the earlier criteria.
- To delete an ACL rule, enter `no seq` *`sequence-number`* in the ACL list.
- To delete an ACL, enter `no ip access-list` *`acl-name`* in Configuration mode.

ACL configuration

1. Create a named ACL (up to 63 characters).

```
ip access-list acl-name
```

2. Add permit and deny rules for IP traffic, and optionally, for packets containing TCP, UDP and ICMP protocol values. Re-enter the command to add more rules. The rules are used to match packets in the sequentially numbered order.

   Enter the *source-ip-prefix* or *dest-ip-prefix* in dotted decimal format *A.B.C.D/mask*.

   The supported protocol numbers are:

   - Internet control message protocol (ICMP) — `1`
   - Internet group membership protocol (IGMP) — `2`
   - Transmission control protocol (TCP) — `6`
   - User datagram protocol (UDP) — `17`
   - Resource reservation protocol (RSVP) — `46`
   - Generic routing encapsulation (GRE) — `47`
   - Authentication header (AUTH) — `51`
   - Protocol-independent multicast (PIM) — `103`
   - Layer two tunneling protocol v.3 (L2TP) — `115`

   For IPv4 permit/deny rules, enter the source or destination IP address, and an optional DSCP value (0 to 63). You can enter `any` to specify all source or destination IP prefixes.

```
seq sequence-number {permit | deny} [protocol-number] [source-ip-prefix | any]
[dest-ip-prefix | any] [dscp value]
```

For TCP permit/deny rules, enter the TCP source and destination IP prefix, an optional TCP port number (0 to 65535), an optional DSCP value (0 to 63), and one or more TCP flags. The TCP flags are for acknowledgment, finish, push, reset, synchronize, and urgent. You enter more than one TCP flag.

```
seq sequence-number {permit | deny} tcp {{source-ip-prefix | any} [src-eq src-port]}
{{dest-ip-prefix | any} [dest-eq dest-port]} [dscp value] {[ack] | [fin] | [psh] | [rst] |
[syn] | [urg]}
```

For UDP permit/deny rules, enter the UDP source and destination IP prefix, an optional UDP port number (0 to 65535), and an optional DSCP value (0 to 63).

```
seq sequence-number {permit | deny} udp {{source-ip-prefix | any} [src-eq src-port]}
{{dest-ip-prefix | any} [dest-eq dest-port]} [dscp value]
```

For ICMP or protocol-number permit/deny rules, enter only the source or destination IP address. The valid protocol numbers are: 1 (ICMP), 2 (IGMP), 6 (TCP), 17 (UDP), 46 (RSVP), 47 (GRE), 51 (AUTH), 103 (PIM), and 115 (L2TP).

```
seq sequence-number {permit | deny} {icmp | protocol-number} [source-ip-prefix | any]
[dest-ip-prefix | any] [dscp value]
```

3. View the configured ACLs.

```
show ip access-lists [acl-name]
```

IP ACL configuration example

```
sonic(config)# ip access-list ACL4
sonic(config-ipv4-acl)# seq 1 permit tcp 4.4.4.4/24 5.5.5.5/24 fin ack dst-eq 100
sonic(config-ipv4-acl)# seq 2 permit tcp any any dst-eq 24
sonic(config-ipv4-acl)# seq 3 permit udp any any
sonic(config-ipv4-acl)# seq 4 permit icmp 3.3.3.3/24 5.5.5.5/24 dscp 10
sonic(config-ipv4-acl)# seq 5 deny 103 any any
```

View IP ACL configuration

```
sonic# show ip access-lists
ip access-list ACL1
     1 permit icmp 11.1.1.1/32 21.1.1.1/32 dscp 1 (0 matches)
     2 deny tcp 11.1.1.2/32 eq 102 21.1.1.2/32 eq 202 fin dscp 2(5 matches)
     3 permit pim 11.1.1.3/32 21.1.1.3/32 dscp 3 (10 matches)
     4 deny tcp 11.1.1.4/32 21.1.1.4/32 dscp 4 (12 matches)
     5 permit tcp 11.1.1.5/32 21.1.1.5/32 dscp 5 (21 matches)
ip access-list ACL2
     1 permit tcp 12.1.1.1/32 eq 101 22.1.1.1/32 eq 201 dscp 1 (4 matches)
     2 permit tcp 12.1.1.2/32 eq 102 22.1.1.2/32 eq 202 dscp 2 (6 matches)
ip access-list ACL3
     1 permit tcp 0.0.0.0/0 0.0.0.0/0 eq 24 (0 matches)
```

Apply ACL to interface

You can apply only one access list for a specified protocol on an inbound or outbound L3 interface. An inbound ACL filters incoming traffic. An outbound ACL filters outgoing traffic. Inbound and outbound ACLs do not filter traffic that originates on the switch.

- On an inbound interface, incoming packets that pass the permit criteria in the inbound ACL are routed to an outbound interface.
- On an outbound interface, outgoing packets are processed by the outbound ACL. If packets match a permit rule, they are transmitted. If packets match a deny rule, they are dropped.

1. Enter Interface configuration mode. Specify the physical port number on the front of the switch.

```
interface ethernet port-number
```

2. Configure an IP address on the interface to place it in L3 mode.

```
ip address ip-address
```

3. Apply an ACL for incoming or outgoing traffic on the L3 interface. Re-enter the command to apply an ACL on traffic in the opposite direction.

```
ip access-group acl-name {in | out}
```

To remove an ACL, enter the `no ip access-group acl-name {in | out}` command.

Apply ACL rules to L3 interface

```
sonic(config)# interface Ethernet 4
sonic(conf-if-Ethernet4)# ip access-group ip-acl1 in
sonic(conf-if-Ethernet4)# ip access-group test5-out1 out
```

View inbound and outbound ACLs

```
# show ip access-group
Ingress IP access-list ACL1 on Ethernet0
Egress IP access-list ACL3 on Ethernet4
```

20

Inband flow analyzer

The Inband flow analyzer (IFA) is a packet and flow monitoring telemetry solution. Configure IFA sessions to use inband telemetry on sampled traffic that is sent to collectors.

The IFA monitors and analyzes packets when they enter and exit the network. It records the path that packets and flows take through the network, the rate at which packets arrive at each hop, and how much time log packets spend at each hop. Use inband IFA to collect network data that out-of-band management methods cannot measure.

IFA configuration

You can configure telemetry and monitoring on switches that act as ingress, intermediate, and egress devices in an IFA session. A switch's role is per-flow. The same switch can operate as the ingress device in one flow, and the intermediate device in another flow.

1. Configure the Telemetry and Monitoring (TAM) device identifier (0 to 65535; default 0). The TAM ID uniquely identifies a switch for data flow monitoring. To delete a TAM ID, use the `no device-id` command.

```
sonic(config)# tam
sonic(config-tam)# device-id
```

2. Configure the collector to which TAM data is sent. Re-enter the command to configure additional collectors. The configured port number identifies the port interface that sends and receives TAM data. To delete a TAM collector, use the `no collector-name` command.

```
sonic(config-tam)# collector collector-name type {ipv4 | ipv6} ip ip-address port port-
number
```

3. Enable IFA data collection.

```
sonic(config-tam)# int-ifa
sonic(config-tam-int-ifa)# feature {enable | disable}
```

4. Specify the ACL flow criteria to match against incoming flow and tag as IFA data. Specify a sampling rate that determines the size of the flow from which one packet is sampled. An IFA data report is sent to the configured collectors. Re-enter the command to configure additional sampling rates on other collectors. To delete a flow configuration, use the `no flow flow-name` command.

```
sonic(config-tam-int-ifa)# flow flow-name acl-table table-name acl-rule acl-name [sampling-
rate rate] [collector-name name]
```

   - For *acl-rule* and `acl-table`, enter the names that are configured for the ACL rule and table.
   - Enter *sampling-rate* to specify the number of packets for the flow size from which one packet is sampled (1 to 10000; default 1)
   - Enter the *collector-name* configured in Step 2.

Configure TAM session

```
sonic(config)# tam
sonic(config-tam)# device-id 10
sonic(config-tam)# collector cnew type ipv4 ip 1.1.1.1 port 10
sonic(config-tam)# int-ifa
sonic(config-tam-int-ifa)# feature enable
sonic(config-tam-int-ifa)# flow fnew acl-table acl1 acl-rule rule1 sampling-rate 10 collector-
name cnew
sonic(config-tam-int-ifa)# end
sonic#
```

View TAM configuration

```
sonic# show tam device
----------------------
TAM Device Information
```

```
----------------------
device-id: 10
```

```
sonic#  show tam collector
-------------------------------------------------
NAME            IP TYPE         IP ADDRESS      PORT
-------------------------------------------------
cnew            ipv4            1.1.1.1         10

sonic# show tam collector cnew
-------------------------------------------------
NAME            IP TYPE         IP ADDRESS      PORT
-------------------------------------------------
cnew            ipv4            1.1.1.1         10
```

```
sonic# show tam int-ifa supported
---------------------------
TAM IFA Feature Information
---------------------------
IFA Feature Supported: True
```

```
sonic# show tam int-ifa flow
------------------------------------------------------------------------
FLOW            ACL TABLE       ACL RULE        SAMPLING RATE  COLLECTOR
------------------------------------------------------------------------
fnew            acl1            rule1           10             cnew

sonic# show tam int-ifa flow fnew
------------------------------------------------------------------------
FLOW            ACL TABLE       ACL RULE        SAMPLING RATE  COLLECTOR
------------------------------------------------------------------------
fnew            acl1            rule1           10             cnew

sonic# show tam int-ifa flow all
------------------------------------------------------------------------
FLOW            ACL TABLE       ACL RULE        SAMPLING RATE  COLLECTOR
------------------------------------------------------------------------
fnew            acl1            rule1           10             cnew
```

21

sFlow

sFlow monitors network traffic in high-speed networks with many switches and routers. The collected data on inbound and outbound traffic is sent to an sFlow data collector.

- SONiC supports sFlow version 5.
- sFlow data collection is only supported on data ports.
- You can configure a maximum of two sFlow collectors.

sFlow uses two types of sampling:

- Statistical packet-based sampling of switched or routed packet flows
- Time-based sampling of interface counters

sFlow monitoring consists of an sFlow agent that is embedded in a switch and an sFlow collector:

- The sFlow agent resides anywhere within the path of the packet. The agent combines the flow samples and interface counters into sFlow datagrams and forwards them to the sFlow collector at regular intervals. The datagrams consist of information about, but not limited to, the packet header, ingress and egress interfaces, sampling parameters, and interface counters. ASICs handle packet sampling.
- The sFlow collector analyses the datagrams that are received from different devices and produces a network-wide view of traffic flows.

Topics:

Configure sFlow

Configure sFlow globally on a switch. By default, the sFlow agent is disabled. You must enable sFlow globally to sample traffic on all data interfaces before you can reconfigure the default settings.

sFlow defaults

- sFlow polling interval: 20 seconds
- sFlow collector port: 6343
- sFlow sampling rates:
  - 1G link: 1 packet in 1000
  - 10G link: 1 packet in 10,000
  - 40G link: 1 packet in 40,000
  - 50G link: 1 packet in 50,000
  - 100G link: 1 packet in 100,000

sFlow configuration

1. Enable sFlow globally on all inbound and outbound interfaces.

```
sonic(config)# sflow enable
```

You can disable sFlow on a per-interface basis in Interface Configuration mode.

```
sonic(config)# interface Ethernet0
sonic(conf-if-Ethernet0)# no sflow enable
```

2. Configure an sFlow collector by entering its IPv4 or IPv6 address. Configure the destination collector-port number for sFlow data traffic (0 to 65535; default 6343).

```
sonic(config)# sflow collector name {ip-address | ipv6-address} [collector-port-number]
```

```
sonic(config)# sflow collector col2 1.1.1.2 4451
```

To remove an sFlow collector, enter `no sflow collector name`.

3. Configure an sFlow agent interface. The interface name provides the IPv4 or IPv6 address for the collector to uniquely identify the source of the packets it receives.

```
sonic(config)# sflow agent-id interface-name
```

```
sonic(config)# sflow agent-id Ethernet0
```

To return the sFlow agent interface to the default, enter `no sflow agent-id`.

4. Configure the sFlow polling interval. The polling interval is the time (in seconds) when traffic samples/counters are collected (5 to 300; default 20). Enter 0 to disable sFlow traffic polling.

```
sonic(config)# sflow polling-interval seconds
```

```
sonic(config)# sflow polling-interval 44
```

To return the sFlow polling interval to the default, enter `no sflow polling-interval`.

5. (Optional) Reconfigure the sFlow sampling rate for packets only in exceptional cases. The sampling rate collects one packet in the specified number of packets (256 to 8388608). The default detects a new flow of 10% of the link bandwidth in less than one second and depends on the interface speed — see **sFlow defaults**. It is recommended that you do not change the default setting.

```
sonic(config)# sflow sampling-rate number
```

```
sonic(config)# sflow sampling-rate 4400
```

To return the sampling rate to the default value for an interface, enter `no sflow sampling-rate`.

View sFlow statistics

Use the show commands to view sFlow configuration and counters.

View global sFlow configuration

```
sonic# show sflow
-----------------------------------
Global sFlow Information
-----------------------------------
        admin state:       up
        polling-interval:  20
        agent-id:          default
```

View sFlow interface status

```
sonic# show sflow interface
------------------------------------------------------------
sFlow interface configurations
   Interface   Admin State     Sampling Rate
   Ethernet0       up              4000
   Ethernet1       up              4000
   Ethernet2       up              4000
   Ethernet3       up              4000
   Ethernet4       up              4000
   Ethernet5       up              4000
   Ethernet6       up              4000
   Ethernet7       up              4000
   Ethernet8       up              4000
   ...
```

```
   Ethernet24     up              4000
   Ethernet25     up              4000
   Ethernet26     up              4000
   Ethernet27     up              4000
   Ethernet28     up              4000
   Ethernet29     up              4000
```

sFlow configuration example

sFlow provides a flow-based sampling method to monitor network traffic. Various use cases for sFlow include network security monitoring in large enterprise data centers, monitoring traffic for different tenants in a logical network, monitoring traffic on interfaces of interest, and quality of service (QoS) operations.

In this example, an sFlow collector has IP address 190.167.1.1/24 with these configuration steps:

1. Enable sFlow globally on all interfaces.
2. Configure the sFlow collector.
3. Configure nondefault polling interval. Keep the default sampling rate.
4. Verify the sFlow configuration.

```
sonic(config)# sflow enable
sonic(config)# sflow collector collector_1 190.167.1.1
sonic(config)# sflow polling-interval 44
sonic(config)# exit

sonic # show sflow
------------------------
Global sFlow Information
------------------------
admin state:            up
polling-interval:       44
agent-id:               default
configured collectors:  1
   collector_1    190.167.1.1        6343

sonic# show sflow interface
-----------------------------------------------
sFlow interface configurations
   Interface      Admin State      Sampling Rate
   Ethernet0      up               4000
   Ethernet1      up               4000
   Ethernet2      up               4000
   Ethernet3      up               4000
   Ethernet4      up               4000
   Ethernet5      up               4000
   Ethernet6      up               4000
   ...
```

To disable sFlow on an interface, use the `no sflow enable` command in Interface configuration mode.

```
sonic(config)# interface Ethernet 0
sonic(conf-if-Ethernet0)# no sflow enable
sonic(conf-if-Ethernet0)# exit
sonic(config)# interface Ethernet 4
sonic(conf-if-Ethernet4)# no sflow enable
sonic(conf-if-Ethernet4)# exit
```

22

REST API

The REST API allows applications and scripts to have complete, model-driven programmatic control over the SONiC OS, using a standard data model and easy-to-use syntax. REST API requests use Uniform Resource Identifiers (URIs) to define the switch resources to be configured or retrieved. The REST API is enabled by default and complies with RFC 8040.

To view REST API end-points and operations, use a web browser to access the REST API Explorer page at `https://switch-ip-address/ui`. All available YANG and SONiC data models are displayed. Use REST API client tools, such as Swagger or Postman, to generate code from these data models for HTTP requests.

Read and write operations in HTTP requests

To configure and monitor a switch, use REST API client tools, such as Postman or Swagger, to run HTTP web requests — GET, PUT, POST, DELETE, and PATCH. These REST API operations act on SONiC REST resources:

- Configuration and operational data the REST API client accesses in `/restconf/data`.
- Protocol-specific data model operations in `/restconf/operations` that SONiC advertises.

The REST API supports the following HTTP requests for read and write operations. REST API operations are performed on a switch resource that is identified by a Universal Resource Identifier (URI). Examples of curl commands that access a switch using the username `admin` and the password `adminpw` for the URI of a collection and a member resource are:

```
curl -k -u admin:adminpw https://switch-ip-address/restconf/data/openconfig-
interfaces:interfaces/

curl -k -u admin:adminpw https://switch-ip-address/restconf/data/openconfig-
interfaces:interfaces/interface=Ethernet0
```

GET	On a collection resource: Retrieves the URIs of member resources in the collection resource in the response body. On a member resource: Retrieves the representation of the member resource in the response body.
POST	On a collection resource: Creates a member resource in the collection resource using the instructions in the request body. The URI of the created member resource is automatically assigned and returned in the Location header field in the response. On a member resource: Creates a member resource in the member resource using the instructions in the request body. The URI of the created member resource is automatically assigned and returned in the Location header field in the response.
PUT	On a collection resource: Replaces all representations of the member resources in the collection resource with the representation in the request body, or creates the collection resource if it does not exist. On a member resource: Replaces all representations of the member resource or creates the member resource if it does not exist, with the representation in the request body.
PATCH	On a collection resource: Updates all representations of the member resources in the collection resource using the instructions in the request body, or creates the collection resource if it does not exist. On a member resource: Updates all representations of the member resource, or creates the member resource if it does not exist, using the instructions in the request body.
DELETE	On a collection resource: Deletes all representations of the member in the collection resource. On a member resource: Deletes all representations of the member resource.

NOTE: The GET method is a safe, read-only operation — applying it to a resource does not result in a state change of the resource. The GET, PUT and DELETE methods are idempotent - applying them multiple times to a resource results in the same state change of the resource as applying them once, although the response may differ.

REST error codes

REST supports only standard HTTP error codes.

Topics:

REST API authentication

For user authentication, the REST API uses HTTP basic password authentication, server certificates, and JSON Web Token (JWT)-coded tokens with `username` and `password` credentials to authenticate requests. User credentials are sent as an HTTP Authorization header in Base64 format; for example: `"Authorization: Basic YWRtaW46c29uaWNhZG1pbg=="`.

By default, HTTP password and JWT are enabled for REST API authentication on a switch. To verify the currently enabled REST API authentication modes:

```
sonic# show authentication rest
---------------------------------------
REST Client Authentication Modes
---------------------------------------
client_auth: password,jwt
```

To reconfigure the REST API authentication modes:

```
sonic(config)# authentication rest auth-mode
```

Where *auth-mode* is one or more of these values separated by commas:

- `password` — Enable HTTP password authentication.
- `jwt` — Enable JWT token-based authentication.
- `cert` — Enable certificate-based authentication.
- `none` — Remove the configured authentication modes, and restore the defaults: HTTP password and JWT authentication.

Enter multiple values for *auth-mode* by separating them with a comma; for example:

```
sonic(config)# authentication rest password,jwt,cert
```

Use HTTP password authentication

A curl command encodes the user credentials in the `--user` option with a *username*`:`*password* value; for example:

```
curl --user admin:sonicadmin -k https://switch-ip-address/restconf/data/openconfig-
interfaces:interfaces/interface=Ethernet0
```

Use JWT token authentication

A JWT token is valid for 1 hour, with a refresh interval of 30 seconds. This means that you can only refresh the token at most 30 seconds before it expires. If the token expires, you must re-authenticate your REST API session.

To generate a JWT token, send the following curl command to the switch from a remote device:

```
curl -k -X POST https://switch-ip-address/authenticate -d '{"username": "admin", "password":
"sonicadmin"}'
```

The switch returns a response that contains the JWT access code to use instead of username and password to authenticate REST API calls on the switch:

```
{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicm9sZXMiOlsi
YWRtaW4iLCJzdWRvIiwiZG9ja2VyIl0sImV4cCI6MTU4MjI0NTA0M30.3wRyN5FfN3LIg2hTUERm3qT5NQEoCNPxQxrRz3
PcWDg","token_type":"Bearer","expires_in":3600}
```

To use the new JWT token to access the REST API and retrieve data about switch resources; for example, openconfig-interfaces:

```
curl -k -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicm9sZXMiOlsiYWRtaW4iLCJzdWRvI
iwiZG9ja2VyIl0sImV4cCI6MTU4MjI0NTA0M30.3wRyN5FfN3LIg2hTUERm3qT5NQEoCNPxQxrRz3PcWDg"  https://
switch-ip-address/restconf/data/openconfig-interfaces:interfaces/interface=Ethernet0
```

You can refresh a JWT token only valid during the 30-second refresh interval before the one-hour expiration time ends. To refresh the token, copy the current access code into the `refresh` curl command syntax:

```
curl -k -X POST https://switch-ip-address/refresh -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicm9sZXMiOlsiYWRtaW4iLCJzdWRvI
iwiZG9ja2VyIl0sImV4cCI6MTU4MjI0NTA0M30.3wRyN5FfN3LIg2hTUERm3qT5NQEoCNPxQxrRz3PcWDg"
```

Use certificate authentication

REST API authentication using a certificate requires a client certificate to be sent by the client. The certificate must be signed by a certificate authority (CA) and contain the common name (CN) field set to the name of the user.

NOTE: gRPC and REST API servers that perform certificate authentication require that your remote device has a certificate and private key pair.

Create a certificate-key pair on the switch, and store them in the local directory `/etc/sonic/server_cert/`. Then configure the REST server on the switch to accept password, JWT, and certificate authentication. Restart the Management Framework service to enable the configuration for the REST API:

```
admin@sonic:~$ sudo cp -r /home/admin/.cert /etc/sonic/server_cert
admin@sonic:~$ redis-cli -n 4 hmset "DEVICE_METADATA|x509" server_crt "/etc/sonic/server_cert/
certificate.pem"
OK
admin@sonic:~$ redis-cli -n 4 hmset "DEVICE_METADATA|x509" server_key "/etc/sonic/server_cert/
key.pem"
OK
admin@sonic:~$ redis-cli -n 4 hmset "DEVICE_METADATA|x509" ca_crt "/host/cli-ca/cert.pem"
OK
admin@sonic:~$ sudo systemctl restart mgmt-framework.service
```

From the remote device, access the REST API by specifying the certificate-key pair in a curl command:

```
curl  -H  "accept: application/yang-data+json" "https://switch-ip-address/restconf/data/
openconfig-system:system/state" -k --key key.pem  --cert certificate.pem
```

A successful REST call with approved certificate authentication returns this response:

```
{"openconfig-system:state":{"boot-time":"1582791592","current-datetime":"2020-02-28T02:59:29Z
+00:00","hostname":"st-sjc-z9264f-19"}}
```

REST API requests using curl

Using the REST API, you can provision switches using HTTPS requests. The examples in this section show how to access the REST API using curl commands. Curl is a Linux shell command that generates HTTPS requests and is run on an external server.

Curl commands

- `-X` specifies the HTTPS request type; for example, `POST` , `PATCH`, or `GET`.
- `-u` specifies the username and password to use for server authentication.
- `-k` runs a REST request in insecure mode. Insecure mode does not verify a server certificate. Although an encrypted HTTPS connection is used, the server certificate is self-signed and not verified.
- `-H` specifies an extra header to include in the request when sending HTTPS to a server. You can enter multiple extra headers.
- `-d` sends the specified data in an HTTPS request.

Use `%2F` to represent a forward slash (`/`); for example, enter the IP address and prefix length `10.1.2.3/24` as `10.1.2.3%2F24`.

For more information, see the curl Manpage.

REST API operations

POST	Creates data only if the data does not exist.
PUT	Replaces data if it exists, or creates data if the data does not exist.
PATCH	Creates or updates data, but does not delete it.
DELETE	Deletes data if it exists.
GET	Retrieves configuration and operational data.

Usage information

Take into account the following items when you access the REST API using `curl` commands:

- In an HTTPS request, it is recommended that you use the most specific URI that is available to avoid unnecessary data and delay in the response.
- When a REST query is in progress, you cannot configure any CLI commands until a REST query is complete.

REST API examples

Some common REST API operations include retrieving interface information and configuring interface settings, such as MTU. These examples use curl commands to send the HTTPS request. It is recommended that you use the Swagger user interface to show all REST endpoints and then test them in a web browser.

NOTE: In each REST API request, use the IP address of the Management interface. The following examples use the Management IP address `10.11.56.39`.

Get information on all interfaces

```
curl -k -u "admin:sonicadmin" -X GET "https://10.11.56.39/restconf/data/openconfig-
interfaces:interfaces" -H "accept: application/yang-data+json"
```

Get information on a specified interface

```
curl -k -u "admin:sonicadmin" -X GET "https://10.11.56.39/restconf/data/openconfig-
interfaces:interfaces/interface=Ethernet8" -H "accept: application/yang-data+json"
```

Configure interface MTU

```
curl -k -u "admin:sonicadmin" -X PATCH "https://10.11.56.39/restconf/data/openconfig-
interfaces:interfaces/interface=Ethernet0/config/mtu" -H  "accept: application/yang-data
+json" -H  "Content-Type: application/yang-data+json" -d "{  \"openconfig-interfaces:mtu\":
9105}"
```

Create an ACL

```
curl -k -u "admin:sonicadmin" -X POST "https://10.11.56.39/restconf/data/openconfig-acl:acl/
acl-sets" -H "accept: application/yang-data+json" -H "Content-Type: application/yang-data
+json" -d "{\"openconfig-acl:acl-set\":[{\"name\": \"MyACL1\",\"type\": \"ACL_IPV4\", \"config
\": {\"name\": \"MyACL1\",\"type\": \"ACL_IPV4\",\"description\": \"Description for
MyACL1\"}}]}"
```

Get ACL information

A POST request does not return a response. You must send a GET request to verify the configuration change; for example, to receive information for all ACLs:

```
curl -k -u "admin:sonicadmin" -X GET "https://10.11.56.39/restconf/data/openconfig-acl:acl/
acl-sets" -H "accept: application/yang-data+json"
{
  "openconfig-acl:acl-sets": {
    "acl-set": [
      {
        "config": {
          "description": "Description for MyACL1",
          "name": "MyACL1",
          "type": "openconfig-acl:ACL_IPV4"
        },
        "name": "MyACL1",
        "state": {
          "description": "Description for MyACL1",
          "name": "MyACL1",
          "type": "openconfig-acl:ACL_IPV4"
        },
        "type": "openconfig-acl:ACL_IPV4"
      }
    ]
  }
}
```

To receive information about a specific ACL:

```
curl -k -u "admin:sonicadmin" -X GET "https://10.11.56.39/restconf/data/openconfig-acl:acl/
acl-sets/acl-set=MyACL1,ACL_IPV4" -H "accept: application/yang-data+json"
{
  "openconfig-acl:acl-set": [
    {
      "config": {
        "description": "Description for MyACL1",
        "name": "MyACL1",
        "type": "openconfig-acl:ACL_IPV4"
      },
      "name": "MyACL1",
      "state": {
        "description": "Description for MyACL1",
        "name": "MyACL1",
        "type": "openconfig-acl:ACL_IPV4"
      },
      "type": "openconfig-acl:ACL_IPV4"
    }
  ]
}
```

Create VLAN10

```
curl -k -u "admin:sonicadmin" -X PATCH "https://10.11.56.39/restconf/data/openconfig-
interfaces:interfaces/interface=Vlan10/config" -H "accept: application/yang-data+json" -H
"Content-Type: application/yang-data+json" -d "{ \"openconfig-interfaces:config\": { \"name
\": \"Vlan10\" }}"
```

Delete VLAN10

```
curl -k -u "admin:sonicadmin" -X DELETE "https://10.11.56.39/restconf/data/openconfig-
interfaces:interfaces/interface=Vlan10" -H "accept: application/yang-data+json"
```

Add Ethernet36 interface to VLANs 2, 4, and 5

```
curl -k -u "admin:sonicadmin" -X PATCH "https://10.11.56.39/restconf/data/openconfig-
interfaces:interfaces/interface=Ethernet36/openconfig-if-ethernet:ethernet/openconfig-
vlan:switched-vlan/config/trunk-vlans" -H "accept: application/yang-data+json" -H "Content-
Type: application/yang-data+json" -d "{ \"openconfig-vlan:trunk-vlans\": [ 2,4,5 ]}"
```

Remove Ethernet36 interface from VLAN4

```
curl -k -u "admin:sonicadmin" -X DELETE "https://10.11.56.39/restconf/data/openconfig-
interfaces:interfaces/interface=Ethernet36/openconfig-if-ethernet:ethernet/openconfig-
vlan:switched-vlan/config/trunk-vlans=4" -H "accept: application/yang-data+json"
```

Get power supply status

```
curl -kX GET "https://10.11.56.39/restconf/data/openconfig-platform:components/component=PSU
%201/power-supply/state" -H "accept: application/yang-data+json" -H "authorization: Basic
YWRtaW46c29uaWMxMjM="
```

The GET response contains power supply output voltage in encoded form; for example:

```
REST GET Command:
curl -kX GET "https://10.11.177.14/restconf/data/openconfig-platform:components/component=PSU
%201/power-supply/state/openconfig-platform-psu:output-voltage" -H "accept:  yang-data+json" -
H "authorization: Basic YWRtaW46c29uaWMxMjM="

REST GET response:
{"openconfig-platform-psu:output-voltage":"QUMKPQ=="}
```

The **QUMKPQ==** value of the PSU output voltage is base64-encoded. To decode the response, use base64 decode, and copy the decoded 4-bytes into a buffer in Big-Endian (Network) order. The following Python example shows how to decode the voltage value `12.1899995803833`:

```
admin@sonic:~$ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> import struct
>>> struct.unpack('>f', base64.b64decode(b'QUMKPQ=='))
(12.1899995803833,)
```

NOTE: To view power supply status from the command-line interface, use the `show platform psusummary` command.

23

gRPC Network Management Interface

The gRPC Network Management Interface (gNMI) allows you to configure and monitor switches using remote procedure calls (RPCs). gNMI supports both read/write configuration, and telemetry streaming of configuration and operational data using gNOI RPCs. gNMI uses JavaScript Object Notation (JSON) to code data for YANG data objects and complies with RFC 7951.

To create RPCs, you can use examples of gNMI RPC code — such as gnmi_get, gnmi_set, and gnoi_client — in scripts or write your own client software in various programming languages, such as C, C++, C#, Java, JavaScript, PERL, and Python. To access these sample gRPC scripts, go to project-arlo / sonic-telemetry on GitHub. Use the gNMI/gRPC libraries and the protobuf files that are available at OpenConfig gNMI.

gNMI supports four types of requests:

Get	Retrieves configuration and operational data from YANG objects on the switch.A GetRequest is sent from a remote device to the target switch. The switch sends a GetResponse in response to the request.
Set	Updates, replaces, or deletes configuration values in the data tree of a YANG model. An update request modifies the configuration parameters with the values that are specified in the JSON request. A replace request resets the configuration parameters to their default settings. A delete request removes the configured value and resets it to the default. The update and replace requests create an object if it does not exist.
Capabilities	Returns the gNMI version and a list of supported YANG models and encodings.
Subscribe	Allows a client to subscribe for updates to one or more paths.

gNMI authentication

There are three types of authentication which you can include in gNMI requests:

- Username and password — The username and password are sent in the metadata in the request,
- JSON web token (JWT) — JWT requires you to first authenticate using a gNOI RPC call by providing a username and password. The RPC returns a token for the specified user that is valid for one hour, and sends it in the metadata. You must refresh the token within 30 seconds of the 1 hour expiration time, otherwise you must reauthenticate.
- Certificates — Certificate authentication requires the use of a valid certificate, signed by the certificate authority (CA) specified in the switch, and must contain the username in the common name (CN) field. For more information, see gNMI certificate authentication.

By default, HTTP password and JWT are enabled for gNMI authentication on a switch. To verify the currently enabled gNMI authentication modes:

```
sonic# show authentication telemetry
-----------------------------------------
Telemetry Client Authentication Modes
-----------------------------------------
client_auth: password,jwt
```

To reconfigure the gNMI authentication modes:

```
sonic(config)# authentication telemetry auth-mode
```

Where *auth-mode* is one or more of these values separated by commas:

- `password` — Enable HTTP password authentication.
- `jwt` — Enable JWT token-based authentication.
- `cert` — Enable certificate-based authentication.
- `none` — Remove the configured authentication modes, and restore the defaults: HTTP password and JWT authentication.

Enter multiple values for *auth-mode* by separating them with a comma; for example:

```
sonic(config)# authentication telemetry password,jwt,cert
```

NOTE: Username and certificate authentication require a re-authentication of the user and role for each request. To avoid this additional overhead, use JWT authentication. Using a token, JWT-based authentication authenticates the user and role one time for the first request in the entire one-hour session.

JSON encoding

RFC 7951 specifies the rules for using JSON to encode YANG data values, such as leafs, containers, and nodes. If you manually build the JSON payload, use the REST swagger user interface to obtain a JSON template and enter the template. You can also follow these methods:

- Fill out the `protobuf` object using a supported programming language, and then serialize it to JSON.
- Send a gNMI Get request on the path you want to set, then modify the gNMI Get response and send the modified output in a Set request.

NOTE: A gNMI Set request may not return output, such as when you create an object as an IP address. If you manually construct the JSON payload, use a tool like PYANG to print the YANG model tree and use the data tree to build the JSON content. For example, to add a new IPv4 address, the JSON coding is:

gNMI Set request path	`openconfig-interfaces:interfaces/interface[name=Ethernet8]/subinterfaces/subinterface[index=0]`
JSON content	`{ "openconfig-if-ip:ipv4": { "addresses": { "address": [ { "ip": "9.9.9.9", "config": { "ip": "9.9.9.9", "prefix-length": 24 } } ] } } }`

Topics:

gNMI certificate authentication

gNMI remote procedure calls require that you authenticate your access to a switch using one of the methods described in gNMI Network Management Interface. To use certificate authentication:

1. From the Linux shell, create a certificate-key pair and store them in the local directory `/etc/sonic/server_cert/`. Create a CA certificate on the switch and store it in the `/host/cli-ca/` directory. Then configure the gNMI Telemetry service to accept password, JWT, and certificate authentication. Restart the Telemetry service to enable the configuration for the gRPC server:

```
admin@sonic:~$ sudo cp -r /home/admin/.cert /etc/sonic/server_cert
admin@sonic:~$ redis-cli -n 4 hmset "DEVICE_METADATA|x509" server_crt "/etc/sonic/
server_cert/certificate.pem"
OK
admin@sonic:~$ redis-cli -n 4 hmset "DEVICE_METADATA|x509" server_key "/etc/sonic/
server_cert/key.pem"
OK
admin@sonic:~$ redis-cli -n 4 hmset "DEVICE_METADATA|x509" ca_crt "/host/cli-ca/cert.pem"
OK
admin@sonic:~$ redis-cli -n 4 hmset "TELEMETRY|gnmi" client_auth "password,jwt,cert"
```

```
OK
admin@sonic:~$ sudo systemctl restart telemetry.service
```

NOTE: gRPC and REST API servers that perform certificate authentication require that your remote device has a certificate and private key pair.

2. From the remote device, access the gRPC service on a switch by specifying the certificate-key pair and CA certificate file in a gNMI RPC; for example:

   NOTE: If you do not specify a CA certificate or if you are using self-signed certificates, specify the `-insecure` option to disable certificate validation.

```
./gnmi_get -cert certificate.pem -key key.pem -xpath /openconfig-interfaces:interfaces/
interface[name=Ethernet0]/ -target_addr switch-ip-address:8080 -ca cert.pem  -target_name
admin
== getRequest:
prefix: <
>
path: <
  elem: <
    name: "openconfig-interfaces:interfaces"
  >
  elem: <
    name: "interface"
    key: <
      key: "name"
      value: "Ethernet0"
    >
  >
>
encoding: JSON_IETF
```

A successful login and curl command execution returns a response; for example:

```
== getResponse:
notification: <
  timestamp: 1582569532183928802
  prefix: <
  >
  update: <
    path: <
      elem: <
        name: "openconfig-interfaces:interfaces"
      >
      elem: <
        name: "interface"
        key: <
          key: "name"
          value: "Ethernet0"
        >
      >
```

gNMI JWT authentication

A JSON web token is used to authenticate a username and role when accessing the gNMI interface to configure or monitor a switch. You must first generate a token and then use it in a gNMI request. A JWT token expires after the expiration period stated in the response.

Generate the JWT token

```
./gnoi_client -module Sonic -rpc authenticate -jsonin '{"username": "admin", "password":
"sonicadmin"}' -insecure
Sonic Authenticate
{"Token":
{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicm9sZXMiOlsi
YWRtaW4iLCJzdWRvIiwiZG9ja2VyIl0sImV4cCI6MTU4MTUzNDk4NX0.ThySin-
fSEoH86R7nitx9kv6DyPO0Mp2zV8MDZX_10A","type":"Bearer","expires_in":3600}}
```

Use the JWT token in a gNMI Get request

```
./gnmi_get -xpath /openconfig-interfaces:interfaces/interface[name=Ethernet0]/ -insecure -
target_addr localhost:8080 -jwt_token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwicm9sZXMiOlsiYWRtaW4iLCJzdWRvI
iwiZG9ja2VyIl0sImV4cCI6MTU4MTUzNDk4NX0.ThySin-fSEoH86R7nitx9kv6DyPO0Mp2zV8MDZX_10A
== getRequest:
prefix: <
>
path: <
  elem: <
    name: "openconfig-interfaces:interfaces"
  >
  elem: <
    name: "interface"
    key: <
      key: "name"
      value: "Ethernet0"
    >
  >
>
encoding: JSON_IETF

== getResponse:
notification: <
  timestamp: 1581531461243037727
  prefix: <
  >
  update: <
    path: <
      elem: <
        name: "openconfig-interfaces:interfaces"
      >
      elem: <
        name: "interface"
        key: <
          key: "name"
          value: "Ethernet0"
        >
      >
    >
    val: <
      json_ietf_val: "{\"openconfig-interfaces:interface\":[{\"config\":{\"enabled
\":true,\"mtu\":9100,\"name\":\"Ethernet0\"},\"name\":\"Ethernet0\",\"openconfig-if-
ethernet:ethernet\":{\"state\":{\"port-speed\":\"openconfig-if-ethernet:SPEED_25GB\"}},\"state
\":{\"admin-status\":\"UP\",\"description\":\"\",\"enabled\":true,\"mtu\":9100,\"name
\":\"Ethernet0\",\"oper-status\":\"DOWN\"},\"subinterfaces\":{\"subinterface\":[{\"index
\":0}]}}]}"
    >
  >
>
```

gNMI password authentication

Password authentication for gNMI requests requires you to provide a username and password to access a switch.

gNMI password authentication for get interface counters

```
# ./gnmi_get -insecure -xpath /openconfig-interfaces:interfaces/interface[name=eth0]/state/
counters -target_addr switch-ip-address:8080 -username admin -password sonicadmin
```

```
== getRequest:
prefix: <
>
path: <
  elem: <
    name: "openconfig-interfaces:interfaces"
  >
  elem: <
    name: "interface"
```

```
    key: <
      key: "name"
      value: "eth0"
    >
  >
  elem: <
    name: "state"
  >
  elem: <
    name: "counters"
  >
>
encoding: JSON_IETF
```

```
== getResponse:
notification: <
  timestamp: 1580320160838720664
  prefix: <
  >
  update: <
    path: <
      elem: <
        name: "openconfig-interfaces:interfaces"
      >
      elem: <
        name: "interface"
        key: <
          key: "name"
          value: "eth0"
        >
      >
      elem: <
        name: "state"
      >
      elem: <
        name: "counters"
      >
    >
    val: <
      json_ietf_val: "{\"openconfig-interfaces:counters\":{\"in-discards\":\"0\",
\"in-errors\":\"0\",\"in-multicast-pkts\":\"0\",\"in-octets\":\"489787\",\"in-pkts\":\"7013\",
\"out-discards\":\"0\",\"out-errors\":\"0\",\"out-octets\":\"3073273\",\"out-pkts
\":\"3292\"}}"
    >
  >
>
```

gNMI request examples

To create RPCs used in gNMI requests, you can download sample gRPC code — such as `gnmi_get` and `gnmi_set` — from project-arlo / sonic-telemetry on GitHub. See gRPC Network Management Interface for more information.

Create user

```
# ./gnmi_set -insecure -username admin -password sonicadmin -update /openconfig-
system:system/aaa/authentication/users:@./user.json -target_addr switch-ip-address:8080
```

```
== getRequest user.json:
{
  "openconfig-system:user": [
    {
      "username": "testadmin",Emerald44
      "config": {
        "username": "testadmin",
        "password": "test",
        "password-hashed": "test",
        "ssh-key": "string",
        "role": "admin"
      }
```

```
    }
  ]
}
```

Delete user

```
gnmi_set -insecure -username admin -password sonicadmin -delete /openconfig-system:system/aaa/
authentication/users/user[username=testuser] -target_addr switch-ip-address:8080
```

Get interface counters

```
# ./gnmi_get -insecure -xpath /openconfig-interfaces:interfaces/interface[name=eth0]/state/
counters -target_addr switch-ip-address:8080 -username admin -password sonicadmin
```

```
== getRequest:
prefix: <
>
path: <
  elem: <
    name: "openconfig-interfaces:interfaces"
  >
  elem: <
    name: "interface"
    key: <
      key: "name"
      value: "eth0"
    >
  >
  elem: <
    name: "state"
  >
  elem: <
    name: "counters"
  >
>
encoding: JSON_IETF
```

```
== getResponse:
notification: <
  timestamp: 1580320160838720664
  prefix: <
  >
  update: <
    path: <
      elem: <
        name: "openconfig-interfaces:interfaces"
      >
      elem: <
        name: "interface"
        key: <
          key: "name"
          value: "eth0"
        >
      >
      elem: <
        name: "state"
      >
      elem: <
        name: "counters"
      >
    >
    val: <
      json_ietf_val: "{\"openconfig-interfaces:counters\":{\"in-discards\":\"0\",
\"in-errors\":\"0\",\"in-multicast-pkts\":\"0\",\"in-octets\":\"489787\",\"in-pkts\":\"7013\",
\"out-discards\":\"0\",\"out-errors\":\"0\",\"out-octets\":\"3073273\",\"out-pkts
\":\"3292\"}}"
    >
  >
>
```

Set interface MTU

In the `gnmi_set` command, `mtu.json` specifies a local file that contains the text describing the MTU to be set on the Ethernet0 interface; for example: `{"mtu": 9000}`.

```
# gnmi_set -update /openconfig-interfaces:interfaces/interface[name=Ethernet0]/config/
mtu:@mtu.json -target_addr switch-ip-address:8080  -username admin -password sonicadmin -
insecure
```

```
== setRequest:
prefix: <
>
update: <
  path: <
    elem: <
      name: "sonic-interfaces:interfaces"
    >
    elem: <
      name: "interface"
      key: <
        key: "name"
        value: "Ethernet0"
      >
    >
    elem: <
      name: "config"
    >
    elem: <
      name: "mtu"
    >
  >
  val: <
    json_ietf_val: "{\"mtu\": 9100}"
  >
>
```

```
== getResponse:
prefix: <
>
response: <
  path: <
    elem: <
      name: "sonic-interfaces:interfaces"
    >
    elem: <
      name: "interface"
      key: <
        key: "name"
        value: "Ethernet0"
      >
    >
    elem: <
      name: "config"
    >
    elem: <
      name: "mtu"
    >
  >
  op: UPDATE
>
```

Enable interface

In the `gnmi_set` command, `enabled.json` specifies a local file that contains the text describing the data to be set; for example: `{"enabled": true}`.

```
# gnmi_set -update /openconfig-interfaces:interfaces/interface[name=Ethernet0]/config/
enabled:@enabled.json -target_addr switch-ip-address:8080  -username admin -password
sonicadmin -insecure
```

```
== setRequest:
prefix: <
>
update: <
  path: <
    elem: <
      name: "sonic-interfaces:interfaces"
    >
    elem: <
      name: "interface"
      key: <
        key: "name"
        value: "Ethernet0"
      >
    >
    elem: <
      name: "config"
    >
    elem: <
      name: "enabled"
    >
  >
  val: <
    json_ietf_val: "{\"enabled\": true}"
  >
>
```

```
== getResponse:
prefix: <
>
response: <
  path: <
    elem: <
      name: "sonic-interfaces:interfaces"
    >
    elem: <
      name: "interface"
      key: <
        key: "name"
        value: "Ethernet0"
      >
    >
    elem: <
      name: "config"
    >
    elem: <
      name: "enabled"
    >
  >
  op: UPDATE
>
```

Create VLAN

In the `gnmi_set` command, `vlan.json` specifies a local file that contains the text describing the data to be set; for example:

```
{"openconfig-interfaces:interface":[{"config":{"name ":"Vlan2"},"name":"Vlan2","state":
{"admin-status":"UP","enabled":true,"mtu":9100,"name":"Vlan2"},"subinterfaces":
{"subinterface":[{"index":0}]}}]}
```

or

```
{
  "openconfig-interfaces:interface": [
    {
      "config": {
        "name ": "Vlan2"
      },
      "name": "Vlan2",
      "state": {
        "admin-status": "UP",
        "enabled": true,
        "mtu": 9100,
        "name": "Vlan2"
      },
      "subinterfaces": {
        "subinterface": [
          {
            "index": 0
          }
        ]
      }
    }
  ]
}
```

.

```
# gnmi_set -update /openconfig-interfaces:interfaces/interface/:@vlan.json -target_addr
switch-ip-address:8080 -username admin -password sonicadmin -insecure
```

```
== setRequest:
prefix: <
>
update: <
  path: <
    elem: <
      name: "openconfig-interfaces:interfaces"
    >
    elem: <
      name: "interface"
    >
  >
  val: <
    json_ietf_val: "{\"openconfig-interfaces:interface\":[{\"config\":{\"name
\":\"Vlan2\"},\"name\":\"Vlan2\",\"state\":{\"admin-status\":\"UP\",\"enabled\":true,\"mtu
\":9100,\"name\":\"Vlan2\"},\"subinterfaces\":{\"subinterface\":[{\"index\":0}]}}]}"
  >
>
```

```
== getResponse:
prefix: <
>
response: <
  path: <
    elem: <
      name: "openconfig-interfaces:interfaces"
    >
    elem: <
      name: "interface"
    >
  >
  op: UPDATE
>
```

Subscribe to interface oper-status

```
# ./gnmi_cli -insecure -logtostderr -address switch-ip-address:8080 -query_type s -
streaming_type
TARGET_DEFINED -q /openconfig-interfaces:interfaces/interface[name=Ethernet0]/state/oper-
```

```
status -target OC-YANG -with_user_pass
username: admin
password:
{
  "OC-YANG": {
    "openconfig-interfaces:interfaces": {
      "interface": {
        "Ethernet0": {
          "state": {
            "oper-status": "{\"openconfig-interfaces:oper-status\":\"DOWN\"}"
          }
        }
      }
    }
  }
}
```

Capabilities request

```
./gnmi_cli -capabilities -insecure -address switch-ip-address:8080 -with_user_pass
username: admin
password:

supported_models: <
  name: "openconfig-acl"
  organization: "OpenConfig working group"
  version: "1.0.2"
>
supported_models: <
  name: "openconfig-routing-policy"
  organization: "OpenConfig working group"
  version: "3.1.1"
>
supported_models: <
  name: "openconfig-mclag"
  organization: "OpenConfig working group"
  version: "1.0.2"
>
supported_models: <
  name: "openconfig-acl"
  organization: "OpenConfig working group"
  version: "1.0.2"
>
...
```

Subscribe to interface LLDP updates in On_Change mode

```
./gnmi_cli -insecure -logtostderr -address 127.0.0.1:8080 -query_type s -streaming_type
ON_CHANGE
-q /openconfig-lldp:lldp/interfaces/interface[name=Ethernet84] -target OC-YANG -with_user_pass
username: admin
password:
```

Subscribe to ACL updates in Sample mode

```
# gnmi_cli -insecure -logtostderr -address 127.0.0.1:8080 -query_type s -
streaming_sample_interval 20 -streaming_type SAMPLE -q /openconfig-acl:acl/ -v 0 -target OC-
YANG -with_user_pass
username: admin
password: {
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
```

Subscribe to ACL updates in Polling mode

```
# gnmi_cli -insecure -logtostderr -address 127.0.0.1:8080 -query_type p -polling_interval 1s -
count 5 -q /openconfig-acl:acl/ -v 0 -target OC-YANG -with_user_pass
username: admin
password: {
```

```
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
{
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
{
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
^CE0206 01:04:23.625717     407 gnmi_cli.go:185] sendQueryAndDisplay(ctx, {Addrs:
[127.0.0.1:8080] AddressChains:[] Target:OC-YANG Replica:0 UpdatesOnly:false Queries:
[[openconfig-acl:acl]] Type:poll Timeout:30s NotificationHandler:<nil> ProtoHandler:<nil>
Credentials:0xc00012ece0 TLS:0xd6f6c0 Extra:map[] SubReq:<nil> Streaming_type:TARGET_DEFINED
Streaming_sample_int:0 Heartbeat_int:0 Suppress_redundant:false}, &{PollingInterval:1s
StreamingDuration:0s Count:1 countExhausted:false Delimiter:/ Display:0x833930 DisplayPrefix:
DisplayIndent:   DisplayType:group DisplayPeer:false Timestamp: DisplaySize:false
Latency:false ClientTypes:[gnmi]}):
  client.Poll(): client.Poll(): EOF
```

Subscribe to ACL updates in Once mode

```
# gnmi_cli -insecure -logtostderr -address 127.0.0.1:8080 -query_type o -q /openconfig-
acl:acl/ -v 0 -target OC-YANG -with_user_pass
username: admin
password: {
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
```

gNMI for streaming telemetry

Network health relies on performance monitoring and data collection for analysis and troubleshooting. Network data is often collected with SNMP and CLI commands using pull mode. In pull mode, a management device sends a get request and pulls data from a client.

As the number of objects in the network and the metrics grow, traditional methods limit network scaling and efficiency. Using multiple management systems further limits network scaling. The pull model increases the processing load on a switch by collecting all data even when there is no change.

Streaming telemetry provides an alternative method where data is continuously transmitted from network devices with efficient, incremental updates. Operators subscribe to the specific data they need using well-defined sensor identifiers.

While SNMP management systems poll for data even if there is no change, streaming telemetry enables access to near real-time, model-driven, and analytics-ready data. It supports more effective network automation, traffic optimization, and preventative troubleshooting. For example, streaming telemetry reports packet drops or high utilization on links in real time. A network automation application can use this information to provision new paths and optimize traffic transmission across the network. The data is encoded using standard IETF JSON and streamed using Google Protocol RPC (gRPC) transport.

You can use gNMI telemetry in a gRPC framework to stream data to:

- Open-source external collectors, such as Telegraph
- Proprietary network collectors that you implement

gNMI modes

A gNMI-based telemetry session can stream data in two modes:

- Dial-in mode — The switch initiates a session with one or more collector devices according to the sensor paths and destinations in a subscription.
- Dial-out mode — A collector initiates a session with the switch.

NOTE: Enterprise SONiC supports gNMI-based streaming telemetry only in dial-in mode. Only dial-in mode is supported for YANG-based streaming.

Subscribe requests

A gNMI client subscribes for updates to one or more paths in one of the following modes:

- `once` — The target creates the relevant update messages, transmits them, and closes the remote procedure call (RPC).
- `poll` — A client sends a Subscribe Request message to the target that contains a poll field with an empty poll message. The target responds with updates to the empty fields.
- `stream` — Operates in `ON_CHANGE`, `SAMPLE` or `TARGET_DEFINED` mode. In a Subscribe stream request, only some fields support `ON_CHANGE`; all fields support `SAMPLE`.
  - `ON_CHANGE` — The target sends an update when a field value changes.
  - `SAMPLE` — The target sends periodic updates. In `SAMPLE` mode, the sampling period is specified in the Subscribe Request message. Each field has a minimum default sampling interval.
  - `TARGET_DEFINED` — The target sends `ON_CHANGE` updates if they are supported by the gNMI path. If not, the target sends `SAMPLE` updates.

gNMI for telemetry examples

- Use gNMI to specify a sampling interval for streaming telemetry data:

```
# gnmi_cli -insecure -logtostderr -address 127.0.0.1:8080 -query_type s -
streaming_sample_interval 20 -streaming_type SAMPLE -q /openconfig-acl:acl/ -v 0 -target
OC-YANG -with_user_pass
username: admin
password: {
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
```

- Use gNMI to specify a polling interval for retrieving telemetry data:

```
# gnmi_cli -insecure -logtostderr -address 127.0.0.1:8080 -query_type p -polling_interval
1s -count 5 -q /openconfig-acl:acl/ -v 0 -target OC-YANG -with_user_pass
username: admin
password: {
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
{
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
{
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
^CE0206 01:04:23.625717     407 gnmi_cli.go:185] sendQueryAndDisplay(ctx, {Addrs:
[127.0.0.1:8080] AddressChains:[] Target:OC-YANG Replica:0 UpdatesOnly:false Queries:
[[openconfig-acl:acl]] Type:poll Timeout:30s NotificationHandler:<nil> ProtoHandler:<nil>
Credentials:0xc00012ece0 TLS:0xd6f6c0 Extra:map[] SubReq:<nil>
Streaming_type:TARGET_DEFINED Streaming_sample_int:0 Heartbeat_int:0
Suppress_redundant:false}, &{PollingInterval:1s StreamingDuration:0s Count:1
countExhausted:false Delimiter:/ Display:0x833930 DisplayPrefix: DisplayIndent:
DisplayType:group DisplayPeer:false Timestamp: DisplaySize:false Latency:false ClientTypes:
[gnmi]}):
  client.Poll(): client.Poll(): EOF
```

- Use gNMI to specify a single retrieval of telemetry data:

```
# gnmi_cli -insecure -logtostderr -address 127.0.0.1:8080 -query_type o -q /openconfig-
acl:acl/ -v 0 -target OC-YANG -with_user_pass
username: admin
password: {
  "OC-YANG": {
    "openconfig-acl:acl": "{}"
  }
}
```

gRPC Network Operations Interface

The gRPC Network Operations Interface (gNOI) defines a set of gRPC-based services to configure, monitor, and stream data from a switch.

To create RPCs used in gNOI requests, you can download sample gRPC code — such as `gnoi_client` — from project-arlo / sonic-telemetry on GitHub. See gRPC Network Management Interface for more information.

gNOI RPC for show tech support command

```
./gnoi_client -module Sonic -rpc showtechsupport
-jsonin "{\"sonic-show-techsupport:input\":{\"date\":\"2020-01-01T00:00:00.000Z\"}}" -insecure
```

gNOI RPC for copy config command

```
admin@sonic:~$ ./gnoi_client -module Sonic -rpc copyConfig
-jsonin '{"sonic-config-mgmt:input": {"source": "running-configuration", "destination":
"startup-configuration"}}' -insecure
Sonic CopyConfig
{"sonic-config-mgmt:output":{"status-detail":"SUCCESS."}}
```

24

Basic troubleshooting

This information describes basic troubleshooting

Best practices

- View traffic end-to-end from the application's view point
- Deploy network management infrastructure rapidly, where needed, when needed, and on-demand
- Extend analysis beyond the network and watch traffic to and from your host
- Focus on real-time assessment and use trend analysis to backup your conclusions
- Emphasize *effective* over *absolute* — leverage management solutions that resolve your most common, most expensive problem quickly
- Address networking performance issues before you focus on the application performance
- Use methodologies and technologies that fit your network and needs
- Continuously monitor performance and availability as a baseline for system performance and setup up time to quickly separate network issues from application issues

Diagnostic tools

show techsupport

You can generate a collection of information about switch configuration, operation, and logs for troubleshooting purposes. This information is helpful to analyze and diagnose problems that occur during switch operation, and proactively monitor network operation to minimize downtime.

Enterprise SONiC continually gathers diagnostic information about system hardware, operation, and software configuration by default. Use the `show techsupport` command to store the collected system information in a compressed .tar file.

```
/var/dump/sonic_dump_sonic_date_time.tar.gz
```

```
/var/dump/sonic_dump_sonic_20191118_221625.tar.gz
```

After you decompress and extract files from the compressed .tar file, most of the extracted files are in readable format. Larger extracted files such as log files, core files, and other files that contain a large amount of output (dump of all BGP tables) are compressed in gzip format. These larger files have a .gz file extension.

Gather diagnostic information

Use the `show techsupport` command to store the collected system information in a .tar file. To reduce the amount of collected information, specify the starting time from which information is collected.

```
admin@sonic:~$ show techsupport [since date time]
```

Enter the date in the format *YYYY-MM-DD*, where:

- *YYYY* is the year, such as `2020`
- *MM* is the number of the month, from 01 to 12
- *DD* is the number of the day, from 01 to 31

Enter the time in the format `THH:MM:SS[.ddd...]{Z | +hh:mm | -hh:mm}`, where:

- Enter `T` to identify that a time parameter follows
- *HH* is the hour (01 to 24)
- *MM* is the number of minutes (00 to 59)
- *SS* is the second (01 to 60)
- *.ddd...* is an optional decimal of the specified second (example, `.234`.

- `Z` indicates that there is no offset from the specified time)
- *`+hh:mm`* indicates the hours and minutes to be added to the specified time and date
- *`-hh:mm`* indicates the hours and minutes to be subtracted from the specified time and date

```
admin@sonic:~$ show techsupport --since="2020-03-05T07:10:00Z"
```

```
admin@sonic:~$ show techsupport --since="2020-03-05T07:10:00-07:20"
```

Send tech support data to a remote server and view extracted file contents

To transfer the `show tech-support` .tar output file to a remote server and view its contents:

1. Log in to the server and access a directory on the server to which you have write access. The directory must have at least 50 MB of available space.

   ```
   admin@sonic:~$ mkdir dump
   admin@sonic:~$ cd dump
   ```

2. Copy the `show tech-support` .tar file to the directory using a supported file transfer method. When successful, the .tar file and its file size are displayed.

   ```
   admin@sonic:~$ scp admin@switch-ip-address:/var/dump/
   sonic_dump_sonic_20200113_232351.tar.gz ./ admin@password: *******

   sonic_dump_sonic_20200113_232351.tar.gz       100% 2183KB   2.1MB/s   00:00
   ```

3. Extract the contents of the tar.gz file to the server directory using the `tar xvzf` command (example, to the `dump` directory).

   ```
   admin@sonic:~$ tar xvzf sonic_dump_sonic_20200113_232351.tar.gz
   sonic_dump_sonic_20200113_232351/
   sonic_dump_sonic_20200113_232351/generate_dump
   sonic_dump_sonic_20200113_232351/proc/
   sonic_dump_sonic_20200113_232351/proc/vmstat
   sonic_dump_sonic_20200113_232351/proc/ioports
   sonic_dump_sonic_20200113_232351/proc/partitions
   sonic_dump_sonic_20200113_232351/proc/net/
   sonic_dump_sonic_20200113_232351/proc/net/ip6_tables_matches
   sonic_dump_sonic_20200113_232351/proc/net/unix
   ...
   ```

   The `show tech-support` .tar files are extracted in a directory tree. The tree is organized according to the type of information contained in the files. Some examples of the file categories for which subdirectories are created in the output file tree are:

   - Log files — `log` directory
   - Linux configuration files — `etc` directory
   - Generic application dump output — `dump` directory
   - Network hardware driver information — `sai` directory
   - Detailed information about various processes — `proc` directory

   NOTE: Use this command to extract the .tar file contents to a different directory.

   ```
   admin@sonic:~$ tar xvzf filename.tar.gz -C /destination-directory-path
   ```

4. Display the contents of the directory in which you extracted the .tar file. Switch to the top level of the extracted directory tree. Display the subdirectories in the directory tree; for example, `debubgsh`, `dump`, `log`, and so on.

   ```
   admin@sonic:~$ ls -ld *
   drwxr-sr-x 8 userid ncore    4096 Jan 13 15:23 sonic_dump_sonic_20200113_232351
   -rw-r--r-- 1 userid ncore 2235129 Jan 13 15:32 sonic_dump_sonic_20200113_232351.tar.gz

   admin@sonic:~$ cd sonic_dump_sonic_20200113_232351
   userid@xenlogin-eqx-05:~/sonic/showtech/dump/sonic_dump_sonic_20200113_232351

   admin@sonic:~$ ls -d *
   debugsh  dump  etc  generate_dump  log  proc  sai
   ```

5. View the contents of a subdirectory in the directory tree; for example, `log`. The subdirectory contains compressed .gz files.

```
admin@sonic:~$ ls -d log/*
log/auth.log.gz                    log/iccpd.log.gz            log/stpd.log.gz
log/bgpd.log.gz                    log/kern.log.1.gz           log/swss.rec.gz
log/btmp.gz                        log/kern.log.gz             log/syslog.1.gz
log/cron.log.gz                    log/mcelog.gz               log/syslog.gz
log/daemon.log.1.gz                log/messages.1.gz           log/system.journal.gz
log/daemon.log.gz                  log/messages.gz             log/telemetry.log.gz
log/debug.1.gz                     log/natorch_debug.log.gz    log/udldd.log.gz
log/debug.gz                       log/neighorch_debug.log.gz  log/user.log.gz
log/dpkg.log.gz                    log/routeorch_debug.log.gz  log/wtmp.gz
log/dropmonitororch_debug.log.gz   log/sairedis.rec.1.gz       log/ztp.log.gz
log/fdborch_debug.log.gz           log/sairedis.rec.gz
```

6. Extract the contents of a .gz file (for example, `log/iccpd.log.gz`) using the `gunzip` command. Use a text editor or the `cat` command to view the contents of an extracted file; for example, `iccpd.log`.

```
admin@sonic:~$ cd log

admin@sonic:~$ gunzip iccpd.log.gz

admin@sonic:~$ ls -d *iccp*
iccpd.log

admin@sonic:~$ cat iccpd.log
Jan 13 23:02:37.468023 sonic NOTICE iccpd#iccpd: [ICCP_FSM.NOTICE] Start ICCP: warm reboot
no
Jan 13 23:10:34.580879 sonic NOTICE iccpd#iccpd: [ICCP_FSM.NOTICE] Start ICCP: warm reboot
no
```

syslog

System logs and event messages from all Docker containers are captured using `rsyslog` and saved in */var/log/syslog*. System logs are enabled by default. To filter the syslog content, use the `grep` option.

Syslog alerts are generated while monitoring memory, disk partition, and CPU thresholds, and core system services. For complete information about configuring and managing a remote syslog server, see *Syslog Server configuration and show* chapter in the *Enterprise SONiC Distribution by Dell Technologies, Management Framework CLI Reference Guide*.

Console logs can be viewed using `show logging`, and the command enters the information in the console syslog. You can follow the log live as entries are written to it by specifying the `-f` or `--follow` flag.

```
admin@sonic:~$ show logging -f
```

```
admin@sonic:~$ show logging
Jan 13 05:46:34.186922 sonic INFO liblogging-stdlog: message repeated 2 times: [ [origin
software="rsyslogd"
swVersion="8.24.0" x-pid="24743" x-info="http://www.rsyslog.com"] rsyslogd was HUPed]
Jan 13 05:46:34.294418 sonic DEBUG admin: admin [22328]: sudo python /etc/spytest/remote-
spytest-helper.py
--init-clean NO,NO,NO
Jan 13 05:46:36.164816 sonic DEBUG admin: admin [22328]: /sbin/ifconfig eth0
Jan 13 05:46:37.306888 sonic DEBUG admin: admin [22328]: stty cols 80
Jan 13 05:46:55.985175 sonic INFO dhclient[24602]: XMT: Solicit on eth0, interval 120990ms.
```

tcpdump

`tcpdump` is a common packet analyzer and is used to display TCP/IP and other packets being transmitted or received over a network.

Topics:

- Port up or down troubleshooting
- Investigating packet drops
- Physical link signal
- Isolate SONiC switch from network

- NAT troubleshooting
- Kernel dump

Port up or down troubleshooting

This information describes how to determine if your network ports are up or down.

All port-related configuration that is done using the CLI or Config_DB is saved in the redis configuration database. This configuration is handled by modules, and the result is stored in the application database (APP_DB). If the modules complete their operation and if the result must be programmed into the ASIC, the result will be synchronized by the syncd service and stored in the ASIC_DB

Verify databases to debug or troubleshoot an issue

1. Check the configuration in CONFIG_DB, and the status using `show` commands.
2. Check the application status of the application in the APP_DB.
3. Check the ASIC-related programming state, and the status in the ASIC_DB.
4. Check the actual ASIC.

Check configuration and port status

1. Check the interface status.

```
admin@sonic:~$ show interface status Ethernet62
Interface Lanes Speed MTU Alias Oper Admin
----------- -------- ------- ----- -------------- ------ -------
Ethernet62 101,102 40G 9100 fiftyGigE1/16/2 up up
```

2. Check the interface transceiver presence.

```
admin@sonic:~$ show interfaces transceiver presence Ethernet4
Port Presence
----------- ----------
Ethernet4 Present
```

3. Dump the port configuration from the ConfigDB.

```
admin@sonic:~$ redis-dump -d 4 -k "PORT|Ethernet4" -y
{
  "PORT|Ethernet4": {
    "type": "hash",
    "value": {
      "admin_status": "up",
      "alias": "fiftyGigE1/2/1",
      "description": "Servers1:eth0",
      "index": "2",
      "lanes": "53,54",
      "mtu": "9100",
      "pfc_asym": "off",
      "speed": "50000"
    }
  }
```

4. Check the port status in the APP_DB.

```
admin@sonic:~$ redis-dump -d 0 -k *PORT_TABLE:Ethernet62* -y
{
  "PORT_TABLE:Ethernet62": {
    "type": "hash",
    "value": {
      "admin_status": "down",
      "alias": "fiftyGigE1/16/2",
      "description": "fiftyGigE1/16/2",
      "index": "16",
      "lanes": "95,96",
      "mtu": "9100",
      "oper_status": "down",
      "pfc_asym": "off",
      "speed": "50000"
    }
  }
```

5. Check the port status in the ASIC_DB.

```
admin@sonic:~$ redis-dump -d 1 -k *ASIC_STATE* -y
"ASIC_STATE:SAI_OBJECT_TYPE_PORT:oid:0x1000000000014" -y
{
  "ASIC_STATE:SAI_OBJECT_TYPE_PORT:oid:0x1000000000014": {
    "type": "hash",
    "value": {
      "NULL": "NULL",
      "SAI_PORT_ATTR_ADMIN_STATE": "true",
      "SAI_PORT_ATTR_INGRESS_ACL": "oid:0xb000000000a61",
      "SAI_PORT_ATTR_MTU": "9122",
      "SAI_PORT_ATTR_PORT_VLAN_ID": "1000",
      "SAI_PORT_ATTR_PRIORITY_FLOW_CONTROL": "24",
      "SAI_PORT_ATTR_QOS_DSCP_TO_TC_MAP": "oid:0x14000000000a34",
      "SAI_PORT_ATTR_QOS_PFC_PRIORITY_TO_QUEUE_MAP": "oid:0x14000000000a35",
      "SAI_PORT_ATTR_QOS_TC_TO_PRIORITY_GROUP_MAP": "oid:0x14000000000a38",
      "SAI_PORT_ATTR_QOS_TC_TO_QUEUE_MAP": "oid:0x14000000000a39",
      "SAI_PORT_ATTR_SPEED": "50000"
    }
  }
```

6. Check the port status for the Broadcom ASIC; from the CLI, enter into the Broadcom shell, then use Ctrl C to exit.

```
BCM : bcmcmd ps
      port     ena/link Lanes Speed Duplex LinkScan AutoNeg? STPstate pause discrd LrnOps
Int
      xe0( 50) down     2     50G   FD     SW       No       Forward        None    FA
KR2
      xe1( 51) down     2     50G   FD     SW       No       Forward        None    FA
KR2
      xe2( 54) up       2     50G   FD     SW       No       Forward        None    FA
KR2
```

Investigating packet drops

This information describes how to investigate packet drops using the `show interfaces counters` command.

- `RX_ERR/TX_ERR` — includes all physical layer (L2) related drops such as FCS error and RUNT frames. If there is an `RX_ERR` or `TX_ERR`, it indicates some physical layer link issues.
- `RX_DRP` — includes all L2, L3, ACL-related drops in the switch ingress pipeline, and drops due to insufficient ingress buffer.
- `TX_DRP` — includes mainly the egress buffer related drop due to congestion including WRED drop.
- `RX_OVR/TX_OVR` — counts the oversized packets.

```
admin@sonic:~$ show interfaces counters
     Iface           RX_OK     RX_RATE   RX_UTIL   RX_ERR   RX_DRP   RX_OVR
TX_OK     TX
---------- --------------- ----------- --------- -------- -------- -------- ---------------
------
 Ethernet0 471,729,839,997 653.87 MB/s    12.77%        0   18,682        0 409,682,385,925
556.84
 Ethernet4 453,838,006,636 632.97 MB/s    12.36%        0    1,636        0 388,299,875,056
529.34
 Ethernet8 549,034,764,539 761.15 MB/s    14.87%        0   18,274        0 457,603,227,659
615.20
Ethernet12 458,052,204,029 636.84 MB/s    12.44%        0   17,614        0 388,341,776,615
527.37
Ethernet16  16,679,692,972  13.83 MB/s     0.27%        0   17,605        0  18,206,586,265
17.51
Ethernet20  47,983,339,172  35.89 MB/s     0.70%        0    2,174        0  58,986,354,359
51.83
Ethernet24  33,543,533,441  36.59 MB/s     0.71%        0    1,613        0  43,066,076,370
49.92
```

Physical link signal

This information describes how to determine the optical signal strength.

NOTE: Not all link types display signal strength values. For example, AOC cables have power values, but DAC cables do not have them.

Optical power should be greater than -10dBm.

```
root@sonic:/# redis-cli -n 6 hgetall "TRANSCEIVER_DOM_SENSOR|Ethernet97"
 1) "temperature"
 2) "0.0000"
 3) "voltage"
 4) "6.5280"
 5) "rx1power"
 6) "-28.2391"
 7) "rx2power"
 8) "N/A"
 9) "rx3power"
10) "N/A"
11) "rx4power"
12) "N/A"
13) "tx1bias"
14) "8.7200"
15) "tx2bias"
16) "N/A"
17) "tx3bias"
18) "N/A"
19) "tx4bias"
20) "N/A"
21) "tx1power"
22) "-28.2391"
23) "tx2power"
24) "N/A"
25) "tx3power"
26) "N/A"
27) "tx4power"
28) "N/A"
29) "temphighalarm"
30) "75.0000"
31) "temphighwarning"
32) "70.0000"
33) "templowalarm"
34) "5.0000"
35) "templowwarning"
36) "10.0000"
37) "vcchighalarm"
38) "3.6300"
39) "vcchighwarning"
40) "3.4650"
41) "vcclowalarm"
42) "2.9700"
43) "vcclowwarning"
44) "3.1350"
45) "txpowerhighalarm"
46) "N/A"
47) "txpowerlowalarm"
48) "N/A"
49) "txpowerhighwarning"
50) "N/A"
51) "txpowerlowwarning"
52) "N/A"
53) "rxpowerhighalarm"
54) "4.9969"
55) "rxpowerlowalarm"
56) "-11.8977"
57) "rxpowerhighwarning"
58) "4.0000"
59) "rxpowerlowwarning"
60) "-7.8995"
61) "txbiashighalarm"
62) "10.0000"
```

```
63) "txbiaslowalarm"
64) "0.5000"
65) "txbiashighwarning"
66) "9.5000"
67) "txbiaslowwarning"
68) "1.0000"
```

Isolate SONiC switch from network

If a SONiC switch is dropping traffic and behaving abnormally, you may want to isolate the device from the network for troubleshooting purposes.

Before isolating a device from the network, use `show techsupport` first (see Basic troubleshooting). You can then shut down BGP sessions to neighbors with `config bgp shutdown`.

1. Generate troubleshooting information.

   ```
   admin@sonic:~$ show techsupport [since dateTime]
   ```

2. Shut down BGP session with neighbor by neighbor's hostname.

   ```
   admin@sonic:~$ sudo config bgp shutdown neighbor SONIC02SPINE
   ```

   Shut down BGP session with neighbor by neighbor's IP address.

   ```
   admin@sonic:~$ sudo config bgp shutdown neighbor 192.168.1.124
   ```

   Shut down BGP session with all neighbors.

   ```
   admin@sonic:~$ sudo config bgp shutdown all
   ```

NAT troubleshooting

This information describes the available commands for network address translation (NAT). All NAT-related configuration is saved in the REDIS database (CONFIG_DB).

When you have IP connectivity problems in a NAT environment, it is often difficult to determine the cause of the problem. Many times NAT is mistakenly blamed, when in reality there is an underlying problem.

Debug NAT issues

- Check if NAT global mode is enabled to display all NAT configuration.

```
admin@sonic:~$ show nat config

Global Values

Admin Mode     : enabled
Global Timeout : 600 secs
TCP Timeout    : 86400 secs
UDP Timeout    : 300 secs

Static Entries

Nat Type   IP Protocol   Global IP   Global Port   Local IP   Local Port Twice-NAT Id
---------- ------------ ----------- ------------- ---------- ---------- ------------
snat       all          112.0.0.2   ---           111.0.0.3  ---             1

Pool Entries

Pool Name   Global IP Range   Global Port Range
----------- ----------------- -------------------
nat1        2.0.0.5           10-200

NAT Bindings

Binding Name   Pool Name   Access-List   Nat Type   Twice-NAT Id
-------------- ----------- ------------- ---------- --------------
```

```
bind1          nat1                      snat       ---

NAT Zones

Port        Zone
--------- ------
Ethernet0      1
Ethernet2      2
```

- Check if the ingress and egress L3 interfaces are configured in different NAT zones.

```
admin@sonic:~$ show nat config zones

Port                Zone
--------------------------------------------------
Ethernet0           1
Loopback0           1
Vlan5               0
PortChannel12       2
```

- Check if the NAT miss packets are reaching the Linux stack to monitor the ingress L3 interface.

```
admin@sonic:~$ tcpdump
```

- Check if the NAT translation entries are learned by the Linux kernel. Use `conntrack -j -L and iptables -t nat -nv -L`.
- Check if the NAT translation entry is created, then view the statistics per NAT entry.

```
admin@sonic:~$ show nat translations

                                        Translated       Translated
Protocol Source         Destination     Source           Destination
-------- ------         -----------     ----------       -----------
all      10.0.0.1       ---             65.55.42.2       ---
all      ---            65.55.42.2      ---              10.0.0.1
all      10.0.0.2       ---             65.55.42.3       ---
all      ---            65.55.42.3      ---              10.0.0.2
tcp      20.0.0.1:4500  ---             65.55.42.1:2000  ---
tcp      ---            65.55.42.1:2000 ---              20.0.0.1:4500
udp      20.0.0.1:4000  ---             65.55.42.1:1030  ---
udp      ---            65.55.42.1:1030 ---              20.0.0.1:4000
tcp      20.0.0.1:6000  ---             65.55.42.1:1024  ---
tcp      ---            65.55.42.1:1024 ---              20.0.0.1:6000
tcp      20.0.0.1:5000  65.55.42.1:2000 65.55.42.1:1025  20.0.0.1:4500
tcp      20.0.0.1:4500  65.55.42.1:1025 65.55.42.1:2000  20.0.0.1:5000
```

```
admin@sonic:~$ show nat statistics

Protocol Source               Destination         Packets   Bytes
-------- ---------            -----------         -------   -------
all      100.100.100.100      200.200.200.5       15        12785
all      17.17.17.17          15.15.15.15         10        12754
all      12.12.12.14          ---                 0         0
all      ---                  138.76.28.1         12        12500
tcp      12.12.15.15:1200     ---                 0         0
tcp      ---                  138.76.29.2:250     8         85120
tcp      100.100.101.101:251  200.200.201.6:276   21        21654
tcp      17.17.18.18:1251     15.15.16.16:1201    18        21765
```

- Check if the NAT translation entries are installed in the switching ASIC. Use `bcmcmd l3 nat_ingress show` and `l3 nat_egress show` Broadcom shell commands.
- Check if the packet counters for the installed NAT entries are incrementing to monitor the packet and byte counter per entry.

```
admin@sonic:~$ show nat statistics

Protocol Source               Destination         Packets   Bytes
-------- ---------            -----------         -------   -------
all      100.100.100.100      200.200.200.5       15        12785
all      17.17.17.17          15.15.15.15         10        12754
all      12.12.12.14          ---                 0         0
all      ---                  138.76.28.1         12        12500
tcp      12.12.15.15:1200     ---                 0         0
```

```
tcp      ---                  138.76.29.2:250     8          85120
tcp      100.100.101.101:251  200.200.201.6:276   21         21654
tcp      17.17.18.18:1251     15.15.16.16:1201    18         21765
```

- Verify that the translated source IP address for the outbound traffic belongs to one of the L3 interfaces
- Check if the translated destination IP address for the inbound traffic is reachable using one of the L3 interfaces

NAT clear commands

- Clear all NAT statistics

```
admin@sonic:~$ clear nat statistics

NAT statistics are cleared.
```

- Clear all dynamic NAT translations

```
admin@sonic:~$ clear nat translations

Dynamic NAT entries are cleared.
```

Kernel dump

To debug a kernel crash, you can generate core files with a dump of the crash. You can analyze the line of code that caused the crash, then view the kernel logs of the events that led to the crash.

The dump files are in `show techsupport` output. Using the `show techsupport` .tar output file, you can export the kernel core files to a remote server for offline analysis (see *Diagnostic tools* in Basic troubleshooting). Use Debian and Linux kernel-provided native tools to view kernel cores.

When you configure the kdump feature, you can change the default and specify the amount of memory to reserve for the kernel core file. Be sure to leave enough available memory for protocol applications to operate. You can also configure the maximum number and size of the core files.

Configure and enable kdump

1. Enable kdump (disabled by default).

```
admin@sonic:~$ sudo config kdump enable
KDUMP configuration has been updated in the startup configuration
Kdump configuration changes will be applied after the system reboots
```

   To disable kdump, use `kdump disable`, then reboot the switch.

```
admin@sonic:~$ sudo config kdump disable

KDUMP configuration has been updated in the startup configuration
ALERT! A system reboot is highly recommended.
Kdump configuration changes will be applied after the system reboots
```

2. (Optional) Configure the amount of memory reserved for the kernel crash dump files. Specify the reserved memory in MB in the format `M`. The `M` parameter is mandatory. The amount of memory that is reserved by default for kernel crash dump files depends on the RAM size.
   - 0 M to 2 GB RAM reserves 256M
   - 2 GB to 4 GB reserves 320M
   - 4 GB to 8 GB reserves 384M
   - 8 GB reserves 448M

```
admin@sonic:~$ sudo config kdump memory 512
% Error: Illegal parameter.

admin@sonic:~$ sudo config kdump memory 512M
KDUMP configuration has been updated in the startup configuration
kdump updated memory will be only operational after the system reboots
```

To reset the amount of memory that is reserved for kernel core dump files to the default value, use `no kdump memory`, then reboot the switch.

3. (Optional) Configure the maximum number of kernel core dump files that can be stored locally. Valid numbers are from 1 and 9; default 3. If a new kernel core dump is generated that exceeds the configured number of files, the oldest stored file is deleted. Save the configuration change.

```
admin@sonic:~$ sudo kdump num_dumps 4
sonic(config)# end
sonic# write memory
```

To reset the maximum number of kernel core files that are stored locally to the default value, use `no kdump num_dumps`, then reboot the switch.

4. Verify the kdump configuration.

```
admin@sonic:~$ show kdump status

Kdump Administrative Mode:  Enabled
Kdump Operational State:    Ready after Reboot
Memory Reserved: 512M
Maximum number of Kernel Core files Stored: 4
No kernel core dump files
```

```
admin@sonic:~$ show kdump memory
Memory Reserved: 512M
```

```
admin@sonic:~$ show kdump num_dumps
Maximum number of Kernel Core files Stored: 4
```

5. From the Linux shell, reboot the switch to apply the kdump configuration settings.

```
admin@sonic:~$ sudo reboot
```

View kernel core dump file after crash

```
admin@sonic:~$ show kdump status

Kdump Administrative Mode:  Enabled
Kdump Operational State:    Ready
Memory Reserved: 512M
Maximum number of Kernel Core files Stored: 4
Record Key           Filename
-----------------------------------------------------------
     1 202003170138 /var/crash/202003170138/dmesg.202003170138
                    /var/crash/202003170138/kdump.202003170138
```

```
admin@sonic:~$ show kdump files

Record  Key           Filename
-----------------------------------------------------------------
     1  202003170138 /var/crash/202003170138/dmesg.202003170138
                     /var/crash/202003170138/kdump.202003170138
```

View kernel core file

To display the last 75 records in the kernel dump log `dmesg`, use `show kdump log` *`record_number`*. Enter the record number of the kernel crash log displayed in `show kdump files` output.

```
admin@sonic:~$ show kdump log 1

File: /var/crash/202003170138/dmesg.202003170138
[  126.213108] i2c i2c-31: new_device: Instantiated device sff8436 at 0x50
[  149.986934] i2c i2c-32: delete_device: Deleting device optoe1 at 0x50
[  150.993025] optoe 32-0050: 32896 byte sff8436 EEPROM, read/write
[  150.994168] i2c i2c-32: new_device: Instantiated device sff8436 at 0x50
[  179.414447] i2c i2c-33: delete_device: Deleting device optoe1 at 0x50
[  180.422898] optoe 33-0050: 32896 byte sff8436 EEPROM, read/write
[  180.422915] i2c i2c-33: new_device: Instantiated device sff8436 at 0x50
[  212.799170] i2c i2c-34: delete_device: Deleting device optoe1 at 0x50
```

```
[  213.805941] optoe 34-0050: 32896 byte sff8436 EEPROM, read/write
[  213.805958] i2c i2c-34: new_device: Instantiated device sff8436 at 0x50
[  250.860500] i2c i2c-35: delete_device: Deleting device optoe1 at 0x50
[  251.866754] optoe 35-0050: 32896 byte sff8436 EEPROM, read/write
...
```