JUNIPER NETWORKS

DAY ONE: EX SERIES UP AND RUNNING

Covers the new ELS-capable EX Series.

Configure your Juniper Networks® EX Series Ethernet Switch with all the new ELS capabilities! This *Day One* will get you up and running in no time.

By Scott Reisinger, J.T. Wilson, Salman Syed

DAY ONE: EX SERIES UP AND RUNNING

Juniper Networks® EX Series Ethernet Switches deliver high-performance, scalable solutions for campus, branch office, and data center environments. You can deploy cost-effective Junos switching solutions that deliver carrier-class reliability, security risk management, network virtualization, application control, and reduced total cost of ownership. *Day One: EX Series Up and Running* contains all the new Enhanced Layer 2 Software (ELS) support developed to provide more programming capabilities and feature support for EX Series devices. The book delivers with hundreds of config examples, tips, and links into the *Juniper TechLibrary*.

"Feature-packed with information and references for network engineers of all levels, from the brand new and uninitiated to the seasoned veteran looking for straight-up insights. Everything inside is a testament to the dedication that Juniper has to their customers."

William C. Etheridge IV, TSgt, USAF, Instructor, Cyber Transport Systems

"This is the perfect book to acquire a better understanding of Juniper EX Series Switches with everything you need to know in order to configure your device from beginning to end in a clear and concise manner."

Stan Carver, Senior Sales Engineer, Raytheon

"The information in this book goes well beyond "Day One" – it's a comprehensive, much-needed reference that includes the updated Junos ELS syntax. Keep this one in your backpack."

Kerri Gillespie, Resident Engineer, Juniper Networks

IT'S DAY ONE AND YOU HAVE A JOB TO DO, SO LEARN HOW TO:

- Know the difference between ELS and non-ELS provisioning
- Identify the different switching hardware models
- Upgrade your system to ELS software and ELS configuration
- Identify the new ELS commands
- Understand the new syntax for trunk and port modes in ELS
- Configure and monitor VLANS and TRUNKS
- Configure inter-VLAN routing
- Configure NTP, RADIUS, TACACS, SNMPv3, DHCP and SYSLOG
- Configure and Monitor port security features
- Secure the EX connection with MACSec

ISBN 978-1-941441-53-4

Day One: EX Series Up and Running

by Scott Reisinger, JT Wilson, and Salman Syed

JUNIPER NETWORKS

Published by Juniper Networks Books
Authors: Scott Reisinger, JT Wilson, and Salman Syed
Technical Reviewers: Scott Crosby, Vaughn Willy
Editor in Chief: Patrick Ames
Copyeditor and Proofer: Nancy Koerbel
Illustrator: Karen Joice

ISBN: 978-1-941441-53-4 (print)
Printed in the USA by Vervante Corporation.

ISBN: 978-1-941441-54-1 (ebook)

Version History: v2, October 2017
2 3 4 5 6 7 8 9 10

http://www.juniper.net/dayone

About the Authors
Scott Reisinger is a Senior Systems Engineer for the USAF at Juniper Networks and GOTO SE for Automation. He has over fifteen years of experience working with devices running the Junos OS and has spent the last seven years with Juniper Networks. Scott is also a 15-year veteran of the USAF and has supported DoD Networks for over thirty years. Scott is the author of *Day One: Deploying Zero Touch Provisioning (ZTP)* by Juniper Networks Books.

JT Wilson is a Consulting Systems Engineer at Juniper Networks and GOTO SE for Switching. He has over sixteen years of networking experience and has been working for Juniper for the last ten years. JT has spent his career at Juniper supporting mission-critical DoD customer networks.

Salman Syed is a Senior Technical Marketing Engineer at Juniper Networks. He's spent seven years with Juniper and has over thirteen years of experience in designing data center and campus networks. He holds a CCIE in routing and switching.

Technical Reviewers
Scott Crosby is a Senior Systems Engineer with Juniper Networks and supports the Federal team out of South Korea. Prior to working for Juniper Networks, he worked for eighteen years with various vendor's equipment including Juniper, Cisco, Brocade and Avaya. Scott also has his own blog where he discusses Juniper technology especially virtualized products. Check it out here: https://netbinenet.wordpress.com/.

Vaughn Willy is a Resident Engineer (RE) with Advanced Services for DoD. Resident Engineers have one of the most important and toughest jobs in the business as they sit on the customer site and provide engineering and operational support for a specific customer.

Welcome to Day One

This book is part of the *Day One* library, produced and published by Juniper Networks Books.

Day One books were conceived to help you get just the information that you need on day one. The series covers Junos OS and Juniper Networks networking essentials with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow.

The *Day One* library also includes a slightly more comprehensive and longer suite of *This Week* books, whose concepts and test bed examples are more similar to a week long seminar.

You can obtain publications from either series in multiple formats:

- Download a free PDF edition at http://www.juniper.net/dayone.
- Get the ebook edition for iPhones and iPads from the iTunes/iBooks Store. Search for *Juniper Networks Books* or the title of this book.
- Get the ebook edition for any device that runs the Kindle app (Android, Kindle, iPad, PC, or Mac) by opening your device's Kindle app and going to the Kindle Store. Search for *Juniper Networks Books* or the title of this book.
- Purchase the paper edition at either Vervante Corporation (www.vervante.com) for between $12-$28, depending on page length.
- Note that most mobile devices can also view PDF files.

Audience

This book is intended for Enterprise network administrators and provides field-tested, common network deployment scenarios, as well as brief background information needed to understand and deploy these solutions in your own environment.

This book's chapters have a logical sequence to guide you through setting up a brand new Juniper Networks® EX Series Ethernet Switch, from typical configuration and management through applying many advanced features.

What You Need to Know Before Reading This Book

It would be ideal to have a basic understanding of routing and switching before reading this book. However, the authors have tried to include everything you need, from the most basic to the more complex configurations, and if you need more knowledge on a subject than what is provided, they have included literally dozens of links to the documentation that expands on that particular topic. Here are some prerequisites that would be helpful:

- Basic networking skills
- Understanding of IP addresses, MAC addresses, and ports and protocols
- Basic familiarity with Junos
- Comprehension of VLANs, Trunks, and Routing

What You Will Learn by Reading This Book

After reading this book you will be able to:

- Know the difference between ELS and non-ELS provisioning
- Identify the different switching hardware models that support ELS
- Use the ELS translator tool provided by Juniper Networks
- Upgrade your system to ELS software and ELS configuration
- Identify the new ELS commands
- Understand the new syntax for trunk and port modes in ELS
- Configure and monitor VLANS and TRUNKS in ELS
- Configure inter-VLAN routing in ELS
- Configure Junos Fusion Enterprise (JFE)
- Configure NTP, RADIUS, TACACS, SNMPv3, DHCP, and SYSLOG
- Configure and Monitor port security features
- Secure the EX connection with MACSec
- Configure and monitor High-Availability (HA)
- Deploy virtual QFX with Vagrant

This Book's Lab Topology

The overall system architecture required for this *Day One* project is pretty simple. Figure 1 depicts the Out of Band (OOB) network topology.

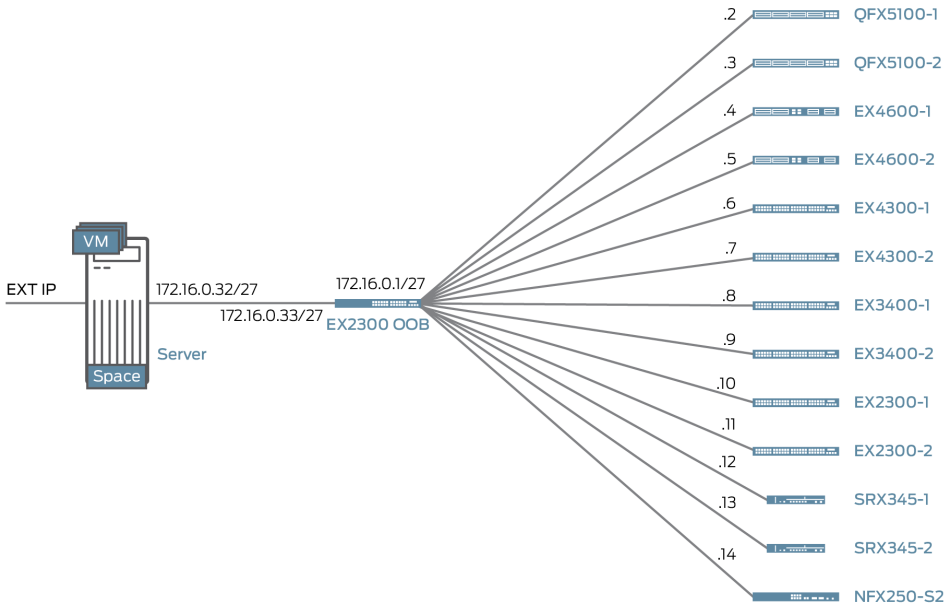


Figure P.1 *This Book's OOB Lab Topology*

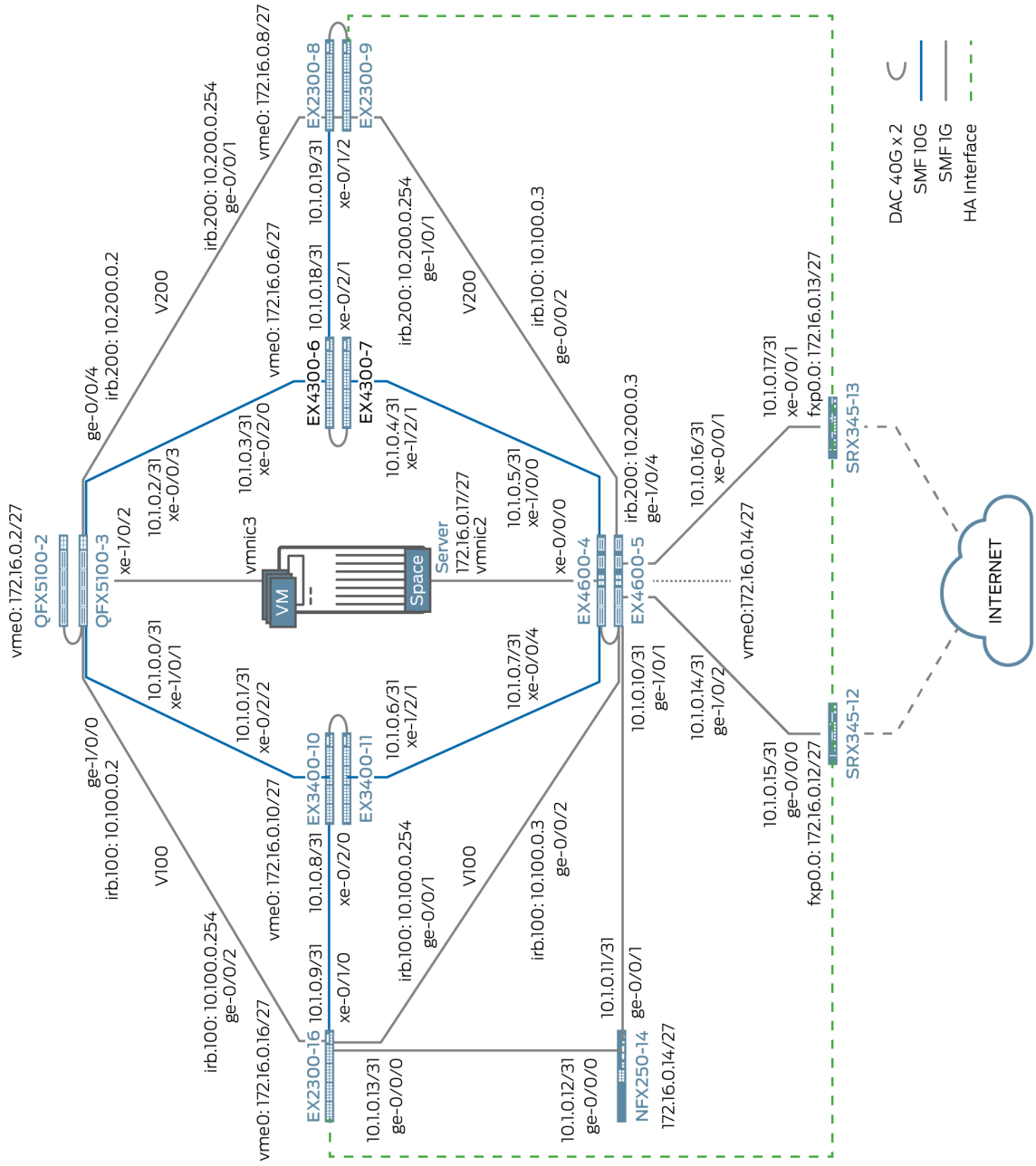


Figure P.2 *This Book's ELS Lab Topology*

For your testing purposes you can use a virtual QFX (vQFX) available from: https://www.juniper.net/us/en/dm/free-vqfx-trial/.

While the QFX Series and the EX Series vary in hardware, the syntax is pretty much the same and this is a FREE and UNLIMITED virtual switch that you can use as long as you like. We'll walk through the setup of the QFX trial and start a 2-node topology so you can see how it is done.

MORE? If at any time you do not understand the terminology used in this document, please refer to the Juniper Technical Documentation at http://www.juniper.net/documentation. Alternatively, you might want to review a few *Day One* books, in its *Fundamentals Series*, such as *Day One: Exploring The Junos CLI, 2nd Edition*. See http://www.juniper.net/dayone for the complete *Day One* library.

Acronyms

Acronyms used throughout this book are:

AAA	Authentication Authorization Accounting
AD	Aggregation Device
API	Application Programmable Interface
ARP	Address Resolution Protocol
ASN.1	Abstract Syntax Notation 1
BFD	Bi-Directional Forwarding Detection
BGP	Border Gateway Protocol
BPDU	Bridge Protocol Data Unit
BUM	Broadcast Unknown-unicast Multicast
CA	Certificate Authority
CAK	Connectivity Association Key
CIDR	Classless Inter-Domain Routing
CIST	Common Internal Spanning Tree
CKN	Connectivity Association Name
CLI	Command Line Interface
CM	Change Management
COS	Class of Service
CSR	Certificate Signing Request
CST	Common Spanning Tree
DAC	Direct Attach Cable
DAI	Dynamic ARP Inspection
DHCP	Dynamic Host Connect Protocol
DNS	Domain Name Service
DOS	Denial of Service
DOT1X	802.1X Protocol
DSCP	Diff-Serve Code Point
EEE	Energy Efficient Ethernet
ELS	Enhanced Layer-2 Software
FPC	Flexible PIC Concentrator
FTP	File Transfer Protocol
GUI	Graphical User Interface
GRE	Generic Route Encapsulation
GRES	Graceful Routing Engine Switchover
HTTP	Hyper Text Transfer Protocol
ICCP	Inter-Chassis Control Protocol
ICL	Inter-Chassis Link
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IRB	Integrated Routing and Bridging Interface
JFE	Junos Fusion Enterprise
JTAC	Juniper Networks Technical Assistance Center
LAG	Link Aggregation Group
LAN	Local Area Network
LCD	Liquid Crystal Display
LLDP	Link Layer Discovery Protocol
LSA	Link State Advertisement
L2	Layer-2
L3	Layer-3
MAC	Machine Address Code
MGMT	Management Interface
MC-LAG	Multi-Chassis Link Aggregation Group

MIB	Management Information Base
MKA	MACSec Key Agreement
MSTP	Multiple Spanning Tree Protocol
MSTI	Multiples Spanning Tree Instance
MTU	Maximum Transmittable Unit
NOC	Network Operation Center
NMS	Network Management System
NSB	Non-Stop Bridging
NSR	Non-Stop Routing
NSSU	Non-Stop Service Upgrade
NTP	Network Time Protocol
OID	Object ID
OOB	Out of Ban
OSPF	Open Shortest Path First
PEM	Privacy Enhanced Mail
PIC	Physical Interface Card
PIM	Protocol Independent Multicast
PFE	Packet Forwarding Engine
POC	Proof of Concept
POE	Power Over Ethernet
PPS	Pulse Policy Secure
PVST+	Per VLAN Spanning Tree Plus
QOS	Quality of Service
QSFP+	Quad Small Form-factor Pluggable
RE	Routing Engine
RFC	Request for Comments
RPC	Remote Procedure Call
RTG	Redundant Trunk Group
RSTP	Rapid Spanning Tree Protocol
SAK	Secure Association Key
SD	Satellite Device
SCP	Secure Copy
SFP	Small Form-factor Pluggable (optic)
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
STP	Spanning Tree Protocol
TCAM	Ternary Content Addressable Memory
TLV	Type Length Value
TOR	Top of Rack
USB	Universal Serial Bus
UTP	Unshielded Twisted Pair
VC	Virtual Chassis
VCE	Virtual Chassis Extended
VCP	Virtual Chassis Port
VIP	Virtual IP
VLAN	Virtual Local Area network
VOIP	Voice over Internet Protocol
VM	Virtual Machine
VPN	Virtual private Network
VRF	Virtual Routing Forwarding
VRRP	Virtual Router Redundancy Protocol
VSTP	VLAN Spanning Tree Protocol
ZTP	Zero Touch Provisioning

Chapter 1

ELS Overview

What is ELS?

Enhanced Layer 2 Software (ELS) support was developed to provide more programming capabilities and feature support in Juniper Networks growing line of EX Series Ethernet Switches and to bring them in line with the rest of the Junos® OS devices that already have switching capabilities such as the MX Series of routers and the data center switching QFX Series.

While ELS does not change the purpose of the device, it does bring the configuration syntax in line with the one Junos operating system for all devices.

The EX2200-C through EX4200 will never have support for ELS and the original Junos syntax will be used. Here is an equivalent hardware replacement path for those products to migrate to ELS-supported hardware:

Non-ELS	*ELS*
EX2200-C	EX2300-C
EX2200	EX2300
EX3200	EX3400
EX3300	EX3400
EX4200	EX4300

The good news is that you can identify the hardware that is ELS capable and the Junos version when ELS was enabled. Table 1.1 lists the hardware device and the initial ELS release supported on that platform. This information is also available online at: https://www.juniper.net/techpubs/en_US/junos/topics/task/configuration/getting-started-els.html.

Table 1.1 ELS Capable Hardware and Software

Device	Initial ELS Release
EX2300	15.X53-D50
EX3400	15.1X53-D50
EX4300 switches	13.2X50-D10
EX4600 switches	13.2X51-D25
EX9200 switches	12.3R2
QFX3500 switches	13.2X50-D15
QFX3600 switches	13.2X50-D15
QFX5100 switches	13.2X51-D10
QFX10000 switches	15.1X53-D10

Upgrading to ELS

If you have purchased any of the new ELS capable hardware to replace the older models, then there is a nifty ELS translator that can take your non-ELS configurations and convert them for you. The ELS translator is a web-based tool that converts Junos Layer 2 configurations to Enhanced Layer 2 Software (ELS) configurations. This conversion tool supports all Juniper Networks EX Series, MX Series, and QFX Series devices with ELS installed. The ELS translator is hosted on the Juniper Networks Customer Support website for EX Series switches, MX Series routers, and QFX Series switches, and is available to registered users, internal users, partners, and premium service contract customers. You need to log in using your Juniper Networks username and password to access the ELS translator, shown open in Figure 1.1: https://www.juniper.net/customers/support/configtools/elstranslator/index.jsp.

As you can see the translator is very intuitive and allows you to copy, paste, or upload a file from your directory. You can also choose the configuration style to reflect your device as well. Once that is accomplished, simply click on the Translate button to submit your configuration.

Here is a capture of the last part of the output on a test box. As you can see, the updated configuration is highlighted in blue and the Download ELS Output button will download the entire ELS converted configuration to your laptop.

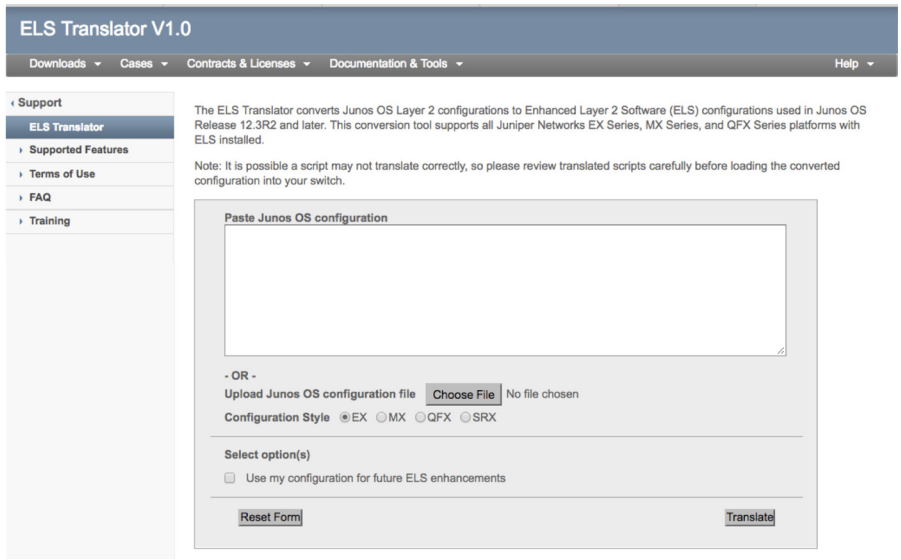


Figure 1.1 ELS Translator V1.0

Figure 1.2 Output from ELS Translator V1.0

Upgrading to ELS Supported Junos OS

Reference Table 1.1 and identify the required Junos OS release that is ELS capable. (Note: The releases in Table 1.1 are the earliest releases and all releases for the hardware after the specified release will be ELS).

To access the Juniper download site: http://www.juniper.net/support/downloads/group/?f=junos.

You must have a valid support agreement and a valid account to log into the portal. You will need your service contract or your chassis serial number that is under support. User registration is located at: https://userregistration.juniper.net/entitlement/loginAssistance.do.

Figure 1.3 is an example of the download site for the EX2300-C.

Take note of the JTAC Recommended box and the green highlighted fields in the table of Figure 1.3. Also, make sure that you use the selection box under Version to select the correct code. Clicking the hyperlink under the install package will start the download process. You will be asked to accept the EULA and then it will download to your default directory.

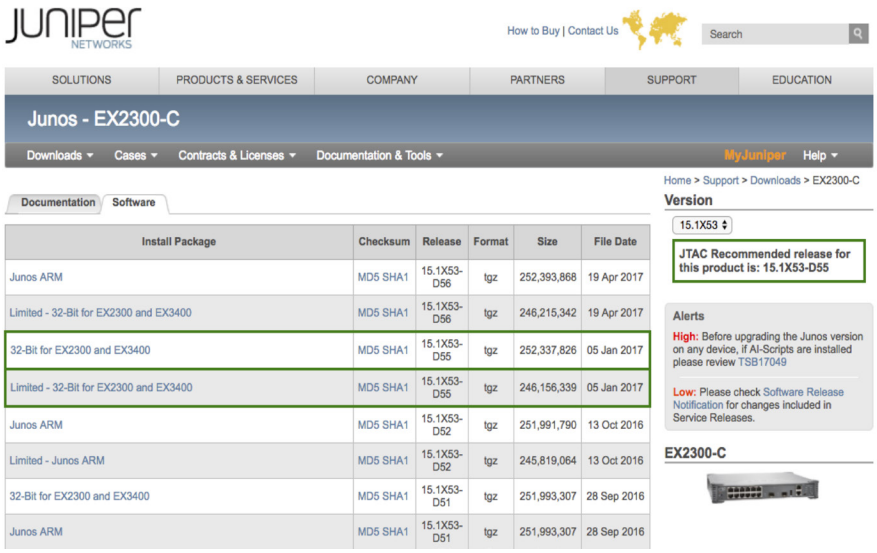


Figure 1.3 *Junos Download page for EX2300-C*

Once you have obtained the version of code for your hardware (Note: It's always best to go with the JTAC recommended code for your platform unless told otherwise by your Juniper Representative.) Then you can Secure Copy (SCP) that code to the device you are going to upgrade.

1. SCP your install package to the device. In this example, upgrading an EX2300-C.

You can also use FTP and HTTP but SCP is the preferred method. You enable SCP by enabling SSH on the device:

```
# set system services ssh
```

Then you can log in using normal credentials:

```
$ scp junos-arm-32-15.1X53-D55.5.tgz username@192.168.2.14:/var/tmp/
```

2. Once your software is staged on the device you can perform the ELS software upgrade. (Note: Check Table 1.1 to make sure your device is ELS-capable before continuing the upgrade process.)

- For standalone devices follow this procedure: https://www.juniper.net/techpubs/en_US/junos/topics/task/installation/ex-series-software-installing-single-routing-engine-cli.html.
- And use the following procedure for upgrading on a virtual chassis: https://www.juniper.net/techpubs/en_US/junos/topics/task/installation/ex-series-software-upgrading-nssu-fixed-virtual-chassis-cli.html.

If your upgrade is successful, you should boot into your new code and see the `login:` prompt.

Applying the ELS Configuration

To apply the new configuration, it is recommended that you follow these steps:

1. Log in to your device using the console port (otherwise you will lose connectivity).

2. Copy the entire existing configuration to another file, on-box, off-box, or both:

- On-box: `switch1 > show configuration | save /var/tmp/Switch.conf`
- Off-box: `scp username@192.168.2.14:/var/tmp/Switch.conf .`

NOTE The dot "." at the end of the SCP statement is required and means copy to the current working directory. You can replace the dot with a fully qualified path if you need.

3. Use the translator tool to convert your configuration saved on your laptop from the previous off-box step.

4. SCP your new ELS translated configuration to the device:

```
$ scp NEW_ELS.conf username@192.168.2.14:/var/tmp/
```

5. Log in to the device and verify the new ELS configuration is on the device:

```
switch1> file list /var/tmp
```

6. Assume configuration/edit mode:

```
switch1> edit exclusive
switch1#
```

7. Delete the original configuration and load the new configuration. Note that the ELS translator produces the configuration in "set mode":

```
switch1# delete
```

8. You have to confirm the deletion by typing `yes`:

```
switch1# load set /var/tmp/NEW_ELS.conf
switch1# commit confirm 3
```

9. `Commit confirm` will load the original configuration if there are any issues causing you not to be able to issue another commit in three minutes:

```
switch1# commit
```

NOTE This has to be issued within the `commit confirm` window. Default is 10 minutes, and 3 minutes was specified here.

10. Verify that all of your connections are up and that the switch functions as required.

New ELS Commands Versus Original Syntax

You might be wondering what changes were made between the original Junos syntax and the improved ELS syntax. As a good network engineer, you should always be checking the vendor technical documentation for updated releases, improvements, and features. Juniper Networks provides all the software documentation for every device within the *TechLibrary*. For example, all EX Series Ethernet Switches Hardware and Software Documentation can be found at the following URL: http://www.juniper.net/techpubs/en_US/release-independent/junos/information-products/pathway-pages/ex-series/product/.

NOTE The Pathway Pages or TechLibrary can provide you with a wealth of information, and you can drill down into your specific hardware or software topic to get right to the point without wading through tons of irrelevant web pages. We highly recommend that use this Day One book in conjuction with the TechLibrary to facilitate your learning experience and get detailed information on specific topics that you need more detail or further clarification on.

An itemized list and side by side comparison of old versus new syntax can be

found here: https://www.juniper.net/techpubs/en_US/junos/topics/task/configuration/getting-started-els.html#jd0e640.

The following section provides a list of existing commands that were moved to new hierarchy levels or changed on EX Series switches as part of this CLI enhancement effort. These sections are provided as a high-level reference only. For detailed information about these commands, use the links to the configuration statements provided or see the technical documentation.

- Changes to the ethernet-switching-options Hierarchy Level
- Changes to the Port Mirroring Hierarchy Level
- Changes to the Layer 2 Control Protocol Hierarchy Level
- Changes to the dot1q-tunneling Statement
- Changes to the L2 Learning Protocol
- Changes to Nonstop Bridging
- Changes to Port Security and DHCP Snooping
- Changes to Configuring VLANs
- Changes to Storm Control Profiles
- Changes to the Interfaces Hierarchy
- Changes to IGMP Snooping

Summary

After reading this chapter you should know what ELS is, how to identify if your hardware is capable of supporting ELS, and what ELS version was first introduced on your particular hardware platform. You should know how to use the upgrade procedure as well as using the ELS translator to convert old style configurations for use with new ELS capable Junos.

In addition, you should have done some research with the provided links on upgrade topics as well as the actual syntax changes to the 11 different areas of hierarchy within the configuration. This should get you on the right path when dealing with ELS now, and in the future!

Okay. Let's get into the lab.

Chapter 2

EX Series Management Overview

Managing via Console

There are several ways to manage EX Series switches, but probably none more important than the console. On all EX switches you will have a console, labeled *CON*, as well as a dedicated out-of-band Ethernet management interface, labeled *MGMT*. Some devices have these on the front, like the compact EX2200-C and EX2300-C, while most others are located on the back of the device, but there should be no problem identifying them as they are clearly marked.

Figure 2.1 is an example of an EX2300-DC rear view with the CON and MGMT interfaces.

Figure 2.1 *EX2300-DC Rear View*

Figure 2.2 is an EX3400-AC with the CON and MGMT interfaces.

Figure 2.2 *EX3400-AC Rear View*

Figure 2.3 shows that EX4600-AC has an SFP port for the MGMT interface as well as a UTP port. Only one of the ports can be used at a time, and you have to specify in the configuration which port you are using.

Figure 2.3 *EX4600-AC Rear View*

To get breakouts of interfaces and field replaceable units in any of the EX Series switches, you can reference the EX Series Ethernet Switches Hardware and Software Documentation in the TechLibrary, and drill down into the model you are interested in: http://www.juniper.net/techpubs/en_US/release-independent/junos/information-products/pathway-pages/ex-series/product/.

Figure 2.4 is a pick-up view of an EX Series marketing document showing the current EX products.

EX2200-C /EX2200	EX2300-C /EX2300	EX3300	EX3400	EX4200	EX4300	EX4550	EX4600	EX9200
Access	Access	Access	Access	Access & Aggregation	Access & Aggregation	Aggregation	Aggregation	Core & Aggregation
Up to 28/104Gbps	Up to 176Gbps	Up to 196Gbps	Up to 336Gbps	Up to 136Gbps	Up to 496Gbps	960Gbps	Up to 1.44Tbps	Up to 13.2Tbps
12-24-48 x GE 2-4 x GE SFP	12-24-48 x GE 2-4 x 10GE SFP+	24-48 x GE 4 x 10GE SFP+	24-48 x GE 4 x 10GE SFP+ 2 x 40GE QSFP+	24-48 x GE 2 x 10GE SFP+	24-48 x GE 4 x 10GE 4 x 40GE QSFP+	48 x 10GE	72 x 10GE SFP+ 12 x 40GE QSFP+	Up to 320 x 10GE Up to 60 x 40GE Up to 20 x 100GE
PoE / PoE+	PoE / PoE+	PoE / PoE+	PoE / PoE+	PoE / PoE+	PoE / PoE+ / SFP	N/A	N/A	N/A
Virtual Chassis Capable								MC-LAG
	Fusion SD		Fusion SD		Fusion SD			Fusion AD

Figure 2.4 *EX Series Table*

The console port is an RS-232 serial interface and uses a standard RJ-45 connector. You can connect to the console using your favorite terminal emulator. The only requirement is that you set the serial parameters to the following:

- BAUD: 9600
- DATA BITS: 8
- PARITY: NONE
- STOP BIT: 1
- FLOW CONTROL: NONE

TIP Laptop manufacturers are no longer including a true serial port on their devices, so make sure that when you are using an USB to SERIAL, that you install the proper drivers on your choice of OS to get it to work. Here is a link to one of our Knowledge Base (KB) articles that will help you find the correct drivers for your OS: https://kb.juniper.net/InfoCenter/index?page=content&id=KB31671&cat=&actp=LIST.

Managing Via Dedicated OOB Ethernet

The dedicated Out-of-Band (OOB) Ethernet interface, labeled as MGMT, is a dedicated RJ-45 port and is used to provide Ethernet access to the device and supports 10/100/1000 BASE-T.

MORE? The dedicated OOB MGMT interface, as well as the default VLAN, are enabled for DHCP by default. In addition, all EX Series switches are ready for Zero Touch Provisioning (ZTP) from the factory. See the *Day One: Deploying Zero Touch Provisioning* book for more information on ZTP URL: https://www.juniper.net/us/en/training/jnbooks/day-one/networking-technologies-series/deploying-zero-touch-provisioning/.

Utilizing and configuring this port matters when you are using the switch as a standalone device, or if you are managing the switch as a Virtual Chassis (VC). When using the MGMT interface on the EX Series switches (with the exception of the EX9200), you will see two types of interfaces. As a standalone switch, you will configure interface me0 unit 0 as the logical interface and when the switch is part of a VC the interface name will be vme unit 0. For example:

```
# set interfaces me0 unit 0 family inet address 192.168.2.10/24
```

Alternative syntax:

```
# set interfaces me0.0 family inet address 192.168.2.10/24
```

When part of a VC:

```
{master:0}[edit]
# set interfaces vme.0 family inet address 192.168.2.10/24
```

TIP Instead of typing out unit to specify the logical interface, Junos allows you to shorthand and use a "." *dot* in place of the keyword unit. This saves some time, but be careful when using that notation on wildcard statements, as some will require the keyword unit to be there.

NOTE You should also note that, unlike other vendors who make you type out the entire mask in dotted decimal format, i.e., 255.255.255.0, Junos uses Classless Inter-Domain Routing (CIDR) notation. Instead of typing out the mask in dotted decimals, you merely indicate how many bits are used for the mask using /24, or /16, or /20; you get the picture!

NOTE When the vme.0 interface is configured, and you have a virtual chassis, all of the switches in the virtual chassis will use the address from the vme.0 interface on the master routing engine. So, if you plug into a line card on the MGMT interface, you will be given the actual logical interface of the master routing engine. To get to a specific line card you can use `request session member<member-id>` to log into that device.

Once you have entered the IP address using the `set interfaces` command you still have to commit the configuration. When you enter edit or configuration mode in Junos, you are working on a *candidate configuration*. You have checked out a copy of the active configuration and the changes you are making are a candidate configuration. Your copy does not get implemented until you issue the `commit` command.

What is really nice about Junos is the `commit confirmed` command. After making your candidate active, the `commit confirmed` will roll back the configuration by default within 10 minutes. If 10 minutes is too long, you can set the rollback time by specifying it at the end, hence `# commit confirmed 3`. Now, *you must* issue an additional `commit` within that 3-minute window, to let Junos know you still have positive control over the system, or it will roll back at the 3-minute mark.

You can also do a `commit check` before doing a `commit` or `commit confirmed`. The `commit check` will tell you if your candidate configuration is okay or it will tell you where you need to correct a command. It does not commit the candidate configuration, it only performs a check.

If you have performed a `commit check` and it tells you all is well and then you issued a `commit confirmed 3`, don't forget to issue another `commit` within 3 minutes or it will automatically roll back to the original active configuration removing your changes. At this point if you have committed the interface setting, you should be able to ping the gateway on that MGMT interface.

NOTE The MGMT interface does not route traffic. Its only purpose in life is to allow management traffic from your OOB management network to PING, SSH, SCP, SNMP, send traps, and logs. It is not a revenue port.

Reference the software documentation for your version of the Junos OS if you need further information on the commit command.

Managing Via In-Band Connection

Unfortunately, in-band management is ubiquitous. An SSH connection or a few pings may not seem like much, but when you add on all of the logging, SNMP polling, NTP updates, and RADIUS or TACACS authentication, that's a lot of traffic that you are sending over your network and all the bandwidth being used to manage your device is taking up precious bandwidth for your customers.

The best solution is to use the dedicated OOB Management Port and offload all that management traffic from taking up precious network resources. However, if that is just not an option, then you are stuck with the in-band method.

When managing the devices in-band it is best to set a loopback interface with a host address that is reachable from your Network Management System (NMS) LAN and then lock that loopback access down to only your NMS address space. There will be more about how to do this later. For now, to set a loopback address, issue the following command:

```
# set interfaces lo0.0 family inet address 1.1.1.1
# show | compare
# commit check
# commit confirmed 2
# commit
# show interfaces lo0
```

Let's take a look at the commands just issued. Just like the me0 interface, you used the "." notation to represent unit, but wait, you didn't specify the CIDR notation for the mask? You should make note that when you *do not include* CIDR notation it will default to `/32` or a host address by default. Therefore, if you do not include the mask on your uplinks it will default to a host address and not work!

You also have a new command `show | compare` (read show pipe compare). This command will show you what changes have been made between your candidate configuration and the active configuration. It's a really good step to make you look at the changes you are getting ready to commit as the active configuration.

Next, there is `commit check`. It's been mentioned already but this command does exactly what you think it does. It just checks to see if the commit will work and reports back to you. If there is an error, it will point to the area where the error was found so you can fix it.

You should use `commit confirmed`, and if you do, you must issue another `commit` within the time window or it will roll back to the point you last committed *or* to the active configuration you checked out when you entered edit mode.

As far as in-band management, you have configured a loopback and you just assumed you already had a revenue-facing port, configured as a Layer 3 interface, or an access port with an Integrated Routing Bridging (IRB) interface.

ELS NOTE The old Junos OS syntax was to specify the Layer 3 interface on a VLAN as VLAN unit 100 or `vlan.100`, but the new ELS syntax uses IRB interfaces, so the command would be:

```
# set vlans V100 vlan-id 100
# set vlans V100 l3-interface irb.100
# set interfaces irb.100 family inet address 192.168.2.10/24
# show | compare
# commit check
# commit confirmed 3
# commit
# show interfaces irb
```

Using the loopback address as the in-band management interface allows you more control over what ports and protocols will be allowed to reach that interface. You want to protect the loopback, as it is the routing engine's IP address. More about firewall filters and policers and protecting the lo0 interface later on.

Managing Via J-WEB

The J-WEB interface is a graphical user interface (GUI) that you can utilize to manage the device. Almost everything that you can do on the command line interface (CLI) you can also do via the GUI.

If you have a brand new out-of-the-box EX2300, or EX4300 Series, with the factory load you can connect an Ethernet cable from your PC to ge-0/0/0 on the switch. Open a browser to http://192.168.1.1 and you should be presented with the login screen. For this exercise, web services was loaded on an EX2300-C in the book's lab.

To enable the J-WEB services:

```
# set system services web-management http interface irb.2
# commit
```

That's all there is to it, and irb.2 was used because that is the in-band management for this switch on VLAN 2. Now you can point your browser to http://192.168.2.6, and you should be presented with something similar to Figure 2.5.

Figure 2.5 *Juniper Networks J-WEB Login Page*

On a factory load you would use `root` as the username and leave the password blank. Once you press the Login button you will be placed into the configure section of the GUI. Figure 2.6 is a quick look inside the J-WEB interface.

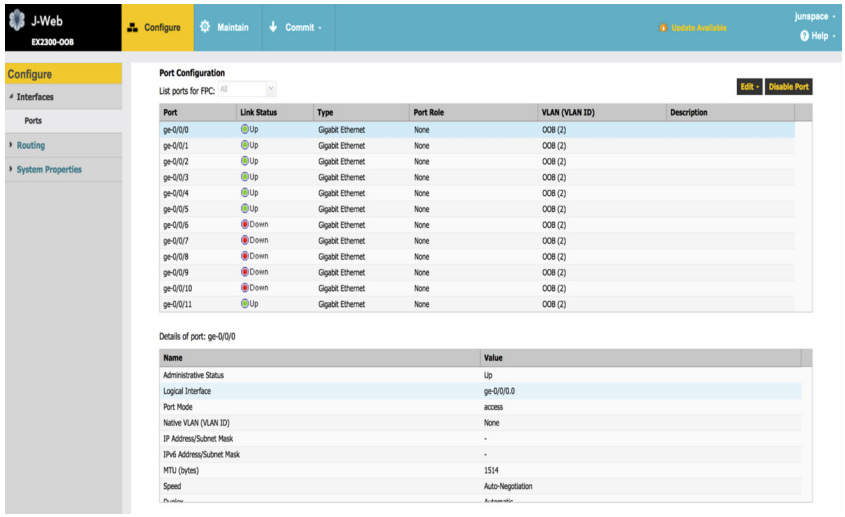


Figure 2.6 *Juniper Networks J-Web Configure Page*

The GUI has come a long way. You can see that there is a Configure section, a Maintain section, and a Commit section on the top task bar. The left pane provides you with all the widgets available in each of those areas.

MORE? That is as far as we will explore the GUI for this book. It is pretty intuitive for the most part and if you need further information on how to configure web services for your particular switch model please refer to: https://www.juniper.net/techpubs/en_US/release-independent/junos/topics/task/configuration/ex-series-initial-configuration-setting-up-j-web.html.

Some entities do not allow web services from their network and are required to turn the web services off. To turn off the web service access, go into edit mode from the CLI and delete the web services statement and then issue a `commit`. To disable web services:

```
# delete system services web-management
# show | compare
[edit system services]
-   web-management {
-       http {
-           interface [ me0.0 irb.2 ];
-       }
-   }
# commit
```

At this point you will no longer be able to get to the GUI because the services are disabled.

Managing Via LCD

Those EX Series switches equipped with an LCD interface can be used to manage the device. The Maintenance menu and Status Menu are enabled by default and you can configure, maintain, and troubleshoot the switch. The LCD can be disabled from the CLI and a custom message can also be displayed in the panel. The LCD can display two lines of up to 16 characters per line.

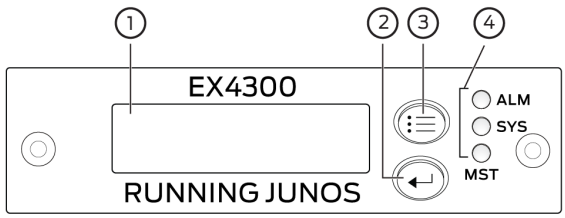


Figure 2.7 LCD Panel in EX4300 Switches

The callouts as illustrated in Figure 2.7 are:

- 1=LCD panel
- 3=LCD panel Menu button
- 2=LCD panel Enter button
- 4= Chassis status LEDs

NOTE When participating in a Virtual Chassis (VC), the LCD will show the member number, the role of the switch, and the host name. You will be able to see the FPC member ID and the role.

From the CLI, you can see what the LCD is displaying, and also the `menu` hierarchy. Let's take a look:

```
EX4300-6_7-VC> show chassis lcd menu
fpc0:
--------------------------------------------------------
status-menu
status-menu vcp-status1-menu
status-menu vcp-status2-menu
status-menu power-status
status-menu environ-menu
status-menu show-version
maintenance-menu
maintenance-menu halt-menu
maintenance-menu system-reboot
maintenance-menu rescue-config
maintenance-menu vc-uplink-config
maintenance-menu factory-default
fpc1:
--------------------------------------------------------
status-menu
status-menu vcp-status1-menu
status-menu vcp-status2-menu
status-menu power-status
status-menu environ-menu
status-menu show-version
maintenance-menu
maintenance-menu halt-menu
maintenance-menu system-reboot
maintenance-menu rescue-config
maintenance-menu vc-uplink-config
maintenance-menu factory-default
```

You can see from the CLI what the LCD is actually displaying the EX4300 in our lab diagram:

```
EX4300-6_7-VC# run show chassis lcd
Front panel contents for slot: 0
---------------------------------
LCD screen:
    00:BK EX4300-6_7
    LED:SPD ALARM 00
```

```
LEDs status:
    Alarms LED: Off
    System LED: Green
    Master LED: Green Blink
Interface  LED(ADM/SPD/DPX/POE)
--------------------------------------
ge-0/0/0        Off
ge-0/0/1        Off
… Truncated for brevity
ge-0/0/20       Off
ge-0/0/21       Off
ge-0/0/22       Off
ge-0/0/23       Off
Front panel contents for slot: 1
--------------------------------
LCD screen:
    01:RE EX4300-6_7
    LED:SPD ALARM 01
LEDs status:
    Alarms LED: Yellow
    System LED: Green
    Master LED: Green
Interface  LED(ADM/SPD/DPX/POE)
--------------------------------------
ge-1/0/0        Off
ge-1/0/1        Off
ge-1/0/2        Off
… Truncated for brevity
ge-1/0/20       Off
ge-1/0/21       Off
ge-1/0/22       Off
ge-1/0/23       Off
```

Only specific models of EX have the LCDs:

- EX3200
- EX3300
- EX4200
- EX4300
- EX4500
- EX4550
- EX6200
- EX8200
- XRE200

You can disable the maintenance capability from the LCD menu and only allow the status menu, or you can disable specific commands from each. In a VC, you will need to specify this for all members. Let's disable the maintenance capability from our VC members 0 and 1:

```
EX4300-6_7-VC# set chassis lcd-menu fpc 0 menu-item maintenance-menu disable
EX4300-6_7-VC# set chassis lcd-menu fpc 1 menu-item maintenance-menu disable
EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
```

And you can see that the maintenance menu is no longer available to anyone using the LCD:

```
EX4300-6_7-VC# run show chassis lcd menu
fpc0:
--------------------------------------------------------------
status-menu
status-menu vcp-status1-menu
status-menu vcp-status2-menu
status-menu power-status
status-menu environ-menu
status-menu show-version
fpc1:
--------------------------------------------------------------
status-menu
status-menu vcp-status1-menu
status-menu vcp-status2-menu
status-menu power-status
status-menu environ-menu
status-menu show-version
```

You can also set a display message on the LCD or even a specific switch in a VC. Let's assume you have an end-of-row VC in your data center and you need another administrator (JJ) to connect a server to port ge-1/0/4 in that VC. Now the other administrator may not know which switch this is, but you can set a message on that switch that says "PORT 4 JJ". That way your staff can locate the exact switch:

```
EX4300-6_7-VC> set chassis display message "PORT 4 JJ" fpc-slot 1 permanent
message sent
EX4300-6_7-VC> show chassis lcd fpc-slot 1
Front panel contents for slot: 1
---------------------------------
LCD screen:
    01:RE EX4300-6_7
    PORT 4 JJ
LEDs status:
    Alarms LED: Yellow
    System LED: Green
    Master LED: Green
Interface  LED(ADM/SPD/DPX/POE)
--------------------------------------
ge-1/0/0        Off
ge-1/0/1        Off
ge-1/0/2        Off
…
```

And the LCD menu is displaying the message.

NOTE When a custom display message is configured on the LCD, the LCD buttons are disabled.

Those are just a few of the commands and features available on the LCD panel. If you want to know more about how to configure the device and the different modes displayed on the LCD, visit the TechLibrary at: https://www.juniper.net/documentation/en_US/release-independent/junos/topics/reference/general/lcd-ex4300-panel.html.

Managing Via SNMP

Simple Network Management Protocol (SNMP) has been around since 1988 as SNMPv1. Improvements were made to it in 1993, and it became known as SNMPv2c. In 1999, a full revision of SNMP was completed and that was adopted as SNMPv3.

- RFC3411 is the IETF Internet Standards track for SNMPv3.
- RFC3418 is the IETF Internet Standards for Management Information Base (MIB).

SNMP has reached the full maturity model in IAW IETF standards. This is a pretty big deal in the networking industry and network engineers should know how to use it.

First, let's take a look at a Management Information Base (MIB). According to RFC3418, an MIB is a *virtual information store*. MIB objects, "virtual pieces of information," are accessed through SNMP. Now those virtual pieces of information, or objects in the MIB, are defined via the Structure of Management Information (SMI) or RFC2578. The Object Identifiers (OIDs) are written in Abstract Syntax Notation One or ASN.1.

That was a lot of information and acronyms to consume in one paragraph. Let's break it down into network terms. SNMP is a protocol to access specific OIDs that are part of a database called a MIB. Each vendor has a unique MIB database and shares a common standard MIB structure. Hopefully, that helps, but let's break it down further so you can see an actual MIB and an OID, and then see SNMP in action.

Next, go hunting for a Juniper MIB. A quick Google search for Junos OS Enterprise MIBs will produce a plethora of links, and the best one is right on top: https://www.juniper.net/techpubs/en_US/release-independent/junos/mibs/mibs.html.

You need to find your Junos OS version; this exercise uses 15.1X53, so go ahead and download the tar or zip version. You are going to select the zip version and unzip it so you can see what perks are inside. After unzipping the file, you are produced with a JuniperMibs directory and a StandardMibs directory. Now let's take a look inside the JuniperMibs folder.

The JuniperMibs folder has 141 text files! Way too many to display here, so let's pick one that you can work with:

```
$ cat mib-jnx-ex-smi.txt
--
-- Juniper Enterprise Specific MIB: Structure of Management Information
--
-- Copyright (c) 2002-2009, Juniper Networks, Inc.
-- All rights reserved.
--
-- The contents of this document are subject to change without notice.
--
JUNIPER-EX-SMI DEFINITIONS ::= BEGIN
IMPORTS
                jnxExMibRoot            FROM JUNIPER-SMI
--
-- This MIB file added the nodes to create the Juniper Security
-- tree structure under the object node: jnxExObjects.
-- In general, the prefix jnxEx is used to name the object identifiers
-- and to designate them.
--
-- The jnxEx node is designed to provide a branch for the Switching
-- related MIB defintions specific to the EX products.
--
--
-- Object identifier added as the basis for identifying other EX nodes.
--
jnxExSwitching                  OBJECT IDENTIFIER ::= { jnxExMibRoot 1 }
--
-- next level object identifiers under jnxExSwitching
--
jnxExAnalyzer                   OBJECT IDENTIFIER ::= { jnxExSwitching 1 }
jnxExSecureAccessPort           OBJECT IDENTIFIER ::= { jnxExSwitching 2 }
jnxExPaeExtension               OBJECT IDENTIFIER ::= { jnxExSwitching 3 }
jnxExVirtualChassis             OBJECT IDENTIFIER ::= { jnxExSwitching 4 }
jnxExVlan                       OBJECT IDENTIFIER ::= { jnxExSwitching 5 }
jnxRPS                          OBJECT IDENTIFIER ::= { jnxExSwitching 6 }
jnxMacNotificationRoot          OBJECT IDENTIFIER ::= { jnxExSwitching 7 }
END
```

That was a very small MIB because some MIBs can take up several pages to display. You'll notice right away that this MIB is copyright protected by Juniper Networks as each vendor has their own set of MIBS. Notice the `Imports` section referencing JUNIPER-SMI and jnxExMibRoot and if you think that those are other MIBs, then you're catching on quick.

Juniper has its own vendor ID in the tree and that is 2636. The actual full OID representing Juniper is `1.3.6.1.4.1.2636`. What does that OID represent?

```
1 = ISO assigned ID's
1.3 = ISO Identified Organizations
1.3.6 = US Department of Defense
1.3.6.1 = OID Assignments from 1.3.6.1 – Internet
1.3.6.1.4 = Internet Private
1.3.6.1.4.1 = IANA-Registered Private Enterprises
1.3.6.1.4.1.2636 = JuniperMIB
```

So, essentially you could get all of the information a Juniper device has to offer by accessing the 1.3.6.1.4.1.2636 OID using SNMP, but it takes a lot of time, and we're talking about thousands of OIDs (remember the 141 MIBs?) So, let's be more selective with the query.

The first Object-Identifier in the JUNIPER-EX-SMI MIB is:

```
jnxExSwitching              OBJECT IDENTIFIER ::= { jnxExMibRoot 1 }
```

Essentially that is telling you to get to jnxExSwitching OID using jnxExRootMib, and add 1 to it. The Juniper jnxExMibRoot OID is 1.3.6.1.4.1.2636.3.40:

```
{iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) 2636 jnxMibs(3)
jnxExMibRoot(40)}
```

Therefore, jnxExSwitching OID becomes 1.3.6.1.4.1.2636.3.40.1, so now you can run a specific walk on your lab EX2300 using that specific OID, but first you need to make sure that the switch can communicate using SNMP. Configure SNMPv2c, first, since it is simple:

```
# set snmp community juniper123
# commit and-quit
> show snmp mib walk 1.3.6.1.4.1.2636.3.40.1
jnxAuthProfileName.0
jnxVirtualChassisMemberSerialnumber.0 = HV0216501848
jnxVirtualChassisMemberRole.0 = 1
jnxVirtualChassisMemberMacAddBase.0 = 28 a2 4b 88  21 5d
jnxVirtualChassisMemberSWVersion.0 = 15.1X53-D51
jnxVirtualChassisMemberPriority.0 = 128
jnxVirtualChassisMemberUptime.0 = 1119964
jnxVirtualChassisMemberModel.0 = ex2300-c-12p
jnxVirtualChassisMemberLocation.0
jnxVirtualChassisMemberAlias.0
jnxVirtualChassisMemberFabricMode.0 = VC
jnxVirtualChassisMemberMixedMode.0 = Not-Mixed
```

That was from the Junos operational prompt, which is something that a lot of other vendors cannot do, by the way. Now, let's do one from a remote device, like a laptop:

```
$ snmpwalk -v 2c -c juniper123 192.168.2.6 1.3.6.1.4.1.2636.3.40.1
SNMPv2-SMI::enterprises.2636.3.40.1.3.1.1.1.0 = ""
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.2.0 = STRING: "HV0216501848"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.3.0 = INTEGER: 1
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.4.0 = Hex-STRING: 28 A2 4B 88 21 5D
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.5.0 = STRING: "15.1X53-D51"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.6.0 = INTEGER: 128
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.7.0 = INTEGER: 1120119
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.8.0 = STRING: "ex2300-c-12p"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.9.0 = ""
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.10.0 = ""
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.11.0 = STRING: "VC"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.12.0 = STRING: "Not-Mixed"
```

You should notice a difference in the output here. On the actual switch the output provides you with the named MIB attribute, such as `jnxVirtualChassisMemberSerial-number.0 = HV0216501848`, whereas on my laptop, it gives me `SNMPv2-SMI::enterprises` out to the Juniper OID 2636 and the rest of the OIDs.

NOTE This is an important point for network engineers who are supporting Network Management Systems such as Junos Space, Cacti, HP Openview, Solarwinds, etc. You need to load the MIBs into the NMS so that it can process the actual details within that MIB and doesn't just present you with an OID. That way you can alert on events from SNMP Traps that are coming into your system and you get granular with what you are requesting from the system.

That single MIB provided us with a lot of information: Chassis Serial Number, Version of Junos running on the system, the model of the chassis MAC address. All of it very useful information.

SNMPv2c and SNMPv3 both use UDP port 161 for SNMP and UDP port 162 for SNMP-TRAPS. Traps are events that are triggered and sent as an SNMP OID from the device to an NMS platform. The difference between SNMPv2c and SNMPv3 is that SNMPv3 is very secure because it uses a Secure Hash Algorithm (SHA1) for authentication and Advanced Encryption Standard (AES) for privacy. This is much more secure than a plain text community, and only limiting to clients, because now you can't sniff the network and see plain text community strings on the wire.

CAUTION A recent article from Tripwire *"Insider Threats as the Main Security Threat in 2017"* brings up a good point. If you have SNMPv2c running and not locked down to a private VLAN, or if you are polling SNMP via a VLAN that is shared by other users, then you might have a serious security problem that someone on the inside of your network can exploit with very little effort.

Managing Via SNMPv3

The good news is that you have already done the bulk of getting to know MIBs and OIDs and how to do a typical *snmpwalk* from Junos CLI and from a remote device. The bad news is, while SNMPv3 is very secure, it is also very labor intensive when configuring. So, for this demonstration, let's configure SNMPv3 with the minimum requirements to get it: a) to be secure; and, b) to work.

The first thing you need to do is to establish a user based security model or USM. Start by identifying the user as `JUNIPER` and making the authentication password `JUNIPER123` (Note that while you are configuring the password in plain text it will be hashed with SHA1 once it is committed):

```
# set snmp v3 usm local-engine user JUNIPER authentication-sha authentication-password JUNIPER123
# set snmp v3 usm local-engine user JUNIPER privacy-aes128 privacy-password JUNIPER123
```

Next create a view-based access control model (VACM). Use a group name of `JUNIPER` and our security model will be USM. The security level will be authentication with privacy and you will only allow a read-view:

```
# set snmp v3 vacm security-to-group security-model usm security-name JUNIPER group JUNIPER
# set snmp v3 vacm access group JUNIPER default-context-prefix security-model usm security-level
privacy read-view JUNIPER
```

Next set the `engine-id` (see the Best Practice sidebar below on engine-id):

```
# set snmp engine-id use-mac-address
# commit
```

Now, if everything is correct, you should be able to walk the device using snmpv3 with authentication and privacy:

```
$ snmpwalk -v 3 -u JUNIPER -a sha -A JUNIPER123 -x aes -X JUNIPER123 -c JUNIPER -l authPriv 192.168.2.6
1.3.6.1.4.1.2636.3.40.1
SNMPv2-SMI::enterprises.2636.3.40.1.3.1.1.1.0 = ""
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.2.0 = STRING: "HV0216500848"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.3.0 = INTEGER: 1
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.4.0 = Hex-STRING: 28 A2 4B 88 21 5D
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.5.0 = STRING: "15.1X53-D51"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.6.0 = INTEGER: 128
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.7.0 = INTEGER: 1123957
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.8.0 = STRING: "ex2300-c-12p"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.9.0 = ""
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.10.0 = ""
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.11.0 = STRING: "VC"
SNMPv2-SMI::enterprises.2636.3.40.1.4.1.1.1.12.0 = STRING: "Not-Mixed"
```

And to see the entire `snmp` hierarchy from the device itself:

```
# show snmp | display set
set snmp v3 usm local-engine user JUNIPER authentication-sha authentication-key
"$9$5Q69tpB1Ec9CORhcleUjiqmTn6At0BjHApBIcSYg4oUjPfz6/tF3revMXxDiHkfT/9pB1hik5F36AtLx7-24aZUkqfoai.
PT3nSrlM7-2gJDHqsYaUDif5Fn/ApBIRSeK81IEyreW8X7-dYgik.Qz6wYaUHkPfhSyl8XwYgaJD-dHqfT3nylevxN"
set snmp v3 usm local-engine user JUNIPER privacy-aes128 privacy-key "$9$tDgXuIESyKv8XEheWLX-
```

```
dTzF69p1IcSlKz3cyKMXxikq.TzCA0IRSOBNdbwg4QF3nApREyKvLFntOBIcS24oJHqmfTn6A.mF/CpB1xN-woJHkPQ36jimTQFA
tO1RcyKMWxdVYvM87NdsYgoJZikFn/u0IDimT3nCALx7-YgDikmPQJZ36ApB17-db4a"
set snmp v3 vacm security-to-group security-model usm security-name JUNIPER group JUNIPER
set snmp v3 vacm access group JUNIPER default-context-prefix security-model usm security-level privacy
read-view READ_JUNIPER
set snmp engine-id use-mac-address
set snmp view READ_JUNIPER oid 1.3.6.1.4.1.2636 include
```

Notice that the passwords have now been hashed. If you try copying this to another device, it will not work with the hashed keys because they are based off of the engine-id that uses the MAC address of the management interface. If you wanted to copy this same SNMP config and paste it to numerous devices you would need to do it with the password.

BEST PRACTICE The `use-mac-address` keyword for the engine-id works great for standalone systems, but not so great for VC (or any device with multiple routing-engines) that could have mastership changes. If you are deploying this on a VC then it would be best to create a local `engine-id` that can be shared across the switches that make up the VC. This would change the SNMP configuration above to: `set snmp engine-id local 0123abcd`. The local `engine-id` can be 24 characters long. For more on creating the local `engine-id` and information on how the `engine-id` is used to encrypt the password please see: http://www.juniper.net/documentation/en_US/junos12.3/topics/reference/configuration-statement/engine-id-edit-snmp.html.

TIP If `snmpwalk` doesn't work the first time after a commit, try applying the USM authentication and privacy keys again and then committing the configuration. See the section above about the engine-id.

TIP Snmpwalk is found in net-snmp-utils on CentOS and Ubuntu. Use yum to install net-snmp-utils on CentOS and apt-get install net-snmp-utils on Ubuntu. On Mac, brew install net-snmp. There are also other SNMP utilities such as snmpget and snmpgetbulk. You should take the time to research those on your own.

MORE? SNMP is a great tool to capture performance metrics, hardware information, interface descriptions, and a plethora of other facts from a device. It can also be used to push configurations, ping, and traceroute. To learn more about the SNMP configuration, as we just scratched the surface here, please go to the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/snmp-configuring-junos-nm.html for SNMPv2c, and go here for SBMPv3: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/snmpv3-minimum-config-junos-nm.html.

Managing Via Mini-USB Type B

You cannot connect to the CON port when managing a device via the Mini-USB as the console input will only be active on one port at a time. The `mini-usb` is set to passive and the normal RS-232 Console is active by default. To use the `mini-usb` you have to explicitly set the port from the CLI:

```
EX4300-6_7-VC# set system ports auxiliary port-type mini-usb

EX4300-6_7-VC# show | compare
[edit system]
+   ports {
+       auxiliary port-type mini-usb;
+   }

EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
```

The next step is to reboot the switch so the mini USB becomes the active console. As you can see, this is not something that you want to make a decision on after the switch is already in a VC on your operational network! This is an up-front decision when performing the initial configuration.

TIP *Do not* use `delete` on the system ports auxiliary `port-type` to change it back to the RJ-45 port. Simply run the command again to replace it with the port type you desire:

```
EX4300-6_7-VC# set system ports auxiliary port-type rj45
EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
```

Once the switch is rebooted you should be able to console using the Mini USB. If your laptop or PC does not have the correct driver for the Mini USB, you could run into issues so make sure you know how to configure the laptop for the proper port.

MORE? For more information regarding the Mini USB port please refer to the TechLibrary: http://www.juniper.net/documentation/en_US/release-independent/junos/topics/task/installation/port-ex2300-mini-usb-management-console-connecting.html.

Managing Storage Via USB

The USB interface on all Juniper EX Series switches is enabled by default. This interface can be used to attach a bootable USB, to boot the switch from a specific Junos version, or you can use the USB as storage to move files to and from the device as needed.

Some organizations do not allow the USB port to be enabled for obvious security reasons. Let's disable the USB on the EX4300 VC in the lab:

```
EX4300-6_7-VC# set chassis usb storage disable
EX4300-6_7-VC# show | compare
[edit chassis]
+   usb {
+       storage {
+           disable;
+       }
+   }

EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
```

There are several items you need to be aware of after disabling the USB port:

- USB ports are enabled by default.
- Even when the USB is disabled the LED will still light if a USB is plugged in.
- If, or when, a switch is booted from a USB you will not have access to disable the USB.
- If the USB port is disabled, you cannot use the command > **`request system reboot media usb`**

To re-enable the port, you have to delete the `disable` statement:

```
EX4300-6_7-VC# delete chassis usb storage disable
EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
EX4300-6_7-VC# run show chassis usb storage
USB Enabled
```

Earlier the capability to mount a USB for storage was mentioned. All flash drives must support USB 2.0 or later and be formatted with a FAT or MS-DOS file system.

When a USB flash drive is inserted into the USB port on the EX Switch you should see messages in the log indicating that the device was attached, the type of device, and its mount point:

```
=====================================================
umass1: SanDisk Cruzer Micro, rev 2.00/2.00, addr 3
da1 at umass-sim1 bus 1 target 0 lun 0
<SanDisk Cruzer 7.01> Removable Direct Access SCSI-0 device
da1: 40.000MB/s transfers
1939MB (3973119 512 byte sectors: 255H 63S/T 247C)
=====================================================
```

This log message gives a valuable piece of information to mount the USB to the filesystem. The bold `da1` identifies the disk and you always assume slice 1. With that information, you can conclude that the USB is `/dev/da1s1`. Now you can drop to the shell as root and mount the USB:

```
EX4300-6_7-VC:RE:1% mount_msdosfs /dev/da1s1 /mnt
```

This will mount the USB into the filesystem at /mnt. You can cd to the /mnt directory (`% cd /mnt`) and then use `ls -al` to see all files on the USB, or you can copy files to or from the `/mnt` directory, and they will be stored on the USB. For more information on mounting the USB to your device please see: https://kb.juniper.net/InfoCenter/index?page=content&id=KB12880&actp=METADATA.

You also need to know how to un-mount the USB properly. This is also accomplished from the root shell. Once it is completed you can pull the USB from the port:

```
EX4300-6_7-VC:RE:1% umount /dev/das1
umount: /dev/das1:
```

TIP If you get a `cannot unmount` message it may be due to you being parked in your USB directory, /mnt in our case, and you need to `cd /` before running the `unmount` command.

We also discussed booting from a USB with a Junos image. There are two methods for getting the image on the USB. The easiest is to use a switch that is already on the right code and do a snapshot.

First, place the USB into the port and then from the CLI issue the command:

```
EX4300-6_7-VC> request system snapshot media external partition
```

The system may go unstable if module traces or syslog messages are enabled during snapshot. It is recommended to disable all debug logging:

```
Do you wish to continue? [yes,no] (no) yes
```

This will copy the image and snapshot files to the USB and it will be ready to use

on a switch to upgrade the code or recover the original device by issuing the command:

```
EX4300-6_7-VC> request system reboot media external
```

The other way is to format your USB as FAT/FAT32 and then copy the Junos software package from your PC/laptop to the USB device. Here are the steps:

1. Format the USB FAT/FAT32.
2. Copy the install media from your PC/Laptop to the USB.
3. Make sure the switch you want to upgrade is powered off.
4. Once the file is on the USB you can plug it in to the USB port on the switch.
5. Power on the switch.
6. Once the Boot Loader message is displayed: press [Enter] to boot immediately, or Space Bar for command prompt.
7. Press the space bar (within one sec of seeing the message) to enter the `loader>` prompt.
8. Type `loader> install source file:///jinstall-ex-4300-14.1X53-D42.3-domestic-signed.tgz.`

Note the colon and three slashes `:///` just after the file keyword. This is the correct syntax to pull the file from the USB.

If successful, the EX Series switch will load the image and then you will be presented with a login prompt. At that point, you can remove the USB from the EX USB port.

Summary

This chapter discussed several different ways to manage an EX Series switch: RS-232 Console, Dedicated Ethernet Management, J-WEB, SNMPv2c and SNMPv3, Mini USB, and LCD, as well as how to use the USB to upgrade or recover an image on an EX Series switch. In addition, there were a lot of additional notes and a short history lesson on SNMP. Hopefully you now have a better understanding of how to access and manage your EX Series device.

Chapter 3

Junos Basics

Junos Initial Configuration

This chapter didn't exist in the first draft but the feedback was that plenty of places are migrating to the new ELS EX Switches, coming from other platforms. So if you are familiar with the initial configuration of an EX Series switch you can skip this section but if you are new to the EX and to the Junos OS, stick around for a quick tutorial.

We've already discussed ways to manage the device and thrown a few commands at you. You might see a repeat of some of those commands in this section, but that's okay, repetition is a good learning tool. And instead of just throwing commands at you, we thought it would be a good idea to run through a typical scenario so you can actually follow the steps and see how they are used in a real implementation.

For this chapter, let's use the lab diagram and assume that you are configuring the standalone switch EX2300-16 for placement into the network. You could attack this like a kid with a new present, but let's do some pre-planning:

1. The root password will be jnpr123.
2. The host-name will be EX2300-16.
3. You will create a local user: lab with password jnpr123.
4. You want to enable SSH and disable telnet, FTP, and web-services.
5. The dedicated OOB Ethernet port MGMT will be used in the network and the address will be 172.16.0.16/27 with a gateway of 172.16.0.1.
6. This device uses VLAN 100 on two separate interfaces ge-0/0/1 and ge-0/0/2.

7. The L3 gateway for VLAN100 is 10.100.0.1/24.
8. The L3 IP address for irb.100 on EX2300-16 for VLAN100 is 10.100.0.254/24.
9. Interfaces ge-0/0/1 and ge-0/0/2 will be configured as TRUNKS and carry VLAN 100.
10. You will configure RSTP on the two L2 interfaces.
11. RSTP bridge priority for this device will be 60k (We never want this device to be root).
12. Interface xe-0/1/0 will be configured for point-to-point L3 with ip address 10.1.0.9/31.
13. Interface ge-0/0/0 will be configured for point-to-point L3 with ip address 10.1.0.13/31.
14. A default route will be set pointing to the 10.100.0.1 gateway.
15. You want to place interface descriptions on all interfaces in use.
16. You want to disable all interfaces that are not in use.
17. You want to make sure to set the date on the system and connect to NTP.
18. You want to add a simple banner.

That is quite the list of requirements! But it will go quickly and it's a real-world example of adding a new switch into an existing network. If you are familiar with Junos, but not the newer ELS commands, you will be made aware of them as we run into them. It should be noted that the lab in this book is in a Juniper proof-of-concept lab many miles away, but the authors have positive control of the device through a terminal server connected to the console.

SHOUT OUT! A special thanks to Sameer Sharma for building this POC and our SE Manager Michael Loefflad for helping to make this transpire. All of the hands-on examples in this book would not be possible without your assistance.

Using our information from Chapter 2, you know that the RS-232 serial port is labeled CON and the terminal server is using settings (9600-8-none-1). Terminal servers usually have several RS-232 connections and you have to know which physical port your device is connected to in order to determine the correct logical port to connect through. In our case, it is logical port 2015, so let's get started:

```
$ telnet 172.16.0.15 2015
Trying 172.16.0.15...
Connected to 172.16.0.15.
Escape character is '^]'.
FreeBSD/arm (Amnesiac) (ttyu0)
```

```
login: root
```

NOTE If you are new to Junos then we want to introduce you to the Amnesiac prompt. If you see `Amnesiac` when you are logging in a Junos device, then that system has the factory load on it. Amnesiac is basically telling you it has no configuration. If you see `Amnesiac` the only way to access the system is using the `root` user with `no password`:

```
--- Junos 15.1X53-D50.2 Kernel 32-bit  JNPR-11.0-20160614.329646_build
root@:RE:0%
```

NOTE Whenever you log in Junos as the root user, you will be placed into the FreeBSD shell indicated by the `%` prompt. To get to the operational CLI simply type `cli` and press Return:

```
Auto Image Upgrade: DHCP Client Bound interfaces:

Auto Image Upgrade: DHCP Client Unbound interfaces: irb.0 vme.0

Auto Image Upgrade: To stop, on CLI apply
"delete chassis auto-image-upgrade" and commit
```

NOTE And with perfect timing, the `Auto Image Upgrade` console message comes up. This message can be quite annoying if you do not do what it tells you to do. Unfortunately, for anyone new to Junos it may not be intuitive. To get to the point where you can issue the `delete chassis auto-image-upgrade` command you first have to get to the operational CLI and then you assume configuration mode. Let's continue so we can do that.

Junos Operational Mode

The Junos Operational CLI prompt is indicated with `>`. The operational mode is where you use show commands for troubleshooting or gathering information or looking at logs. Using context-sensitive help "`?`" after the `show` command will provide you a long list of possible completions you can act on within the Junos hierarchy. Most of the show commands will complete with just one possible completion, but if you want to get more granular and instead of showing all interface information, for example, you can use context sensitive help "`?`" after the first possible completion for example, `> show interfaces ?` will give you a list of the interfaces to select from whereas `> show route ?` will give you another list of protocols and possible completions. You should become familiar with using context-sensitive help.

```
root@:RE:0% cli
{master:0}
root>
```

MORE? Covering each and every operational command is not the goal of this book. If you are looking for a specific command you can utilize the help apropos topic of information you are looking for, an example being, > `help apropos mtu` and it will give you a list of commands that you may be interested in. The help command can be run in operational or configuration mode. For more information on CLI commands and the difference between operational and configuration hierarchy please go to the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/topic-map/logical-systems-operational-mode.html.

MORE? An excellent source of 'new to Junos' information is the *Day One* library. Look for *Day One* books in the *Fundamentals Series*, such as *Day One: Exploring the Junos CLI, Second Editon*.

Junos Configuration or Edit mode

The Junos edit mode is clearly identified as `[edit]` as well as the octothorpe `#` prompt. This is where all changes are made and committed to the active configuration. Now that you truly have the power – can you feel it surging through your fingertips? Now let's get rid of that pesky Auto Image Upgrade message:

```
{master:0}
root> edit
Entering configuration mode

{master:0}[edit]
root#
```

Active and Candidate Configuration

When you enter configuration mode (or edit mode) you automatically check out a "candidate" configuration, meaning the "active" configuration remains untouched until it is committed. You and all of your coworkers can be in the same device working on your own candidate configurations and be oblivious as to what the others are doing. Another feature to combat this is the `edit exclusive` command that locks everyone else out of edit mode until you have committed your configuration or logged back out to the operational mode. Until you actually use the `commit` command to make your candidate configuration active, you will not be impacting the running operational active configuration of the device.

TIP When working on your candidate configuration you can check what is slotted to be added or removed from the active configuration, once you commit, by using the `# show | compare` command. This will show you the deltas between the active configuration and your candidate configuration. You should use this often if you are making more than a couple of changes just to review your work:

```
{master:0}[edit]
root# delete chassis auto-image-upgrade
{master:0}[edit]
root# commit
[edit]
  'system'
    Missing mandatory statement: 'root-authentication'
error: commit failed: (missing mandatory statements)
{master:0}[edit]
root#
Auto Image Upgrade: DHCP Client Bound interfaces:
```

NOTE The Auto Image Upgrade message was removed but there's a commit failed (`missing mandatory statements`) message. The mandatory statement is the `root-authentication password` so let's configure that and try to commit again:

```
{master:0}[edit]
root# set system root-authentication plain-text-password
New password: jnpr123
Retype new password: jnpr123
{master:0}[edit]
root# commit
configuration check succeeds
commit complete
{master:0}[edit]
root#
```

Bask in the silence from the console messages but not for long, there's a lot of work to do. So far we have logged in to the switch, made it from the shell `%` to the operational CLI prompt `>` and then into edit mode `#` where we issued the `delete chassis auto-image-upgrade` command and then set our root password with the `set system root-authentication plain-text-password` command. What did you learn from just those two commands? Hopefully that `delete` (deletes stuff) and `set` (sets stuff) and that nothing changes until you commit. Now let's press on with our configuration and set the host-name, a local user named *lab*, enable SSH, and configure the MGMT interface:

```
{master:0}[edit]
root# set system host-name EX2300-16
root# run set cli screen-width 100
Screen width set to 100
root# set system login user lab class super-user authentication plain-text-password
New password:
Retype new password:
{master:0}[edit]
root# set system services ssh protocol-version v2
root# set interfaces me0.0 family inet address 172.16.0.16/27
root# show | compare
 [edit system login]
+    user lab {
+        class super-user;
+        authentication {
+            encrypted-password "$5$9YdJY4Uf$PvnFzbKAPcxYVbV8GsLmLqL2DcWfgYKRRQx4xga7
DI7"; ## SECRET-DATA
+        }
```

```
+    }
[edit system services]
+    ssh {
+        protocol-version v2;
+    }
[edit interfaces me0 unit 0 family inet]
+       address 172.16.0.16/27;
{master:0}[edit]
root# commit
configuration check succeeds
commit complete
```

A couple of things to note here. You see `run` was used to preface an operational command: `set cli screen-width 100`. You can "run" all operational commands by prefacing the command with `run`. You can also stop paging by setting the screen length to 0: `set cli screen-length 0`. At this point you should be able to ping the gateway on the MGMT interface and then test SSH by logging into the MGMT interface at 172.16.0.16:

```
root@EX2300-16# run ping 172.16.0.1
PING 172.16.0.1 (172.16.0.1): 56 data bytes
64 bytes from 172.16.0.1: icmp_seq=0 ttl=64 time=3.212 ms
64 bytes from 172.16.0.1: icmp_seq=1 ttl=64 time=1.021 ms
^C
--- 172.16.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.021/2.117/3.212/1.095 ms

$ ssh 172.16.0.16
The authenticity of host '172.16.0.16 (172.16.0.16)' can't be established.
RSA key fingerprint is ae:c8:75:95:fb:e6:45:6a:e7:0b:df:46:99:79:47:44.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.0.16' (RSA) to the list of known hosts.
Password:
--- Junos 15.1X53-D50.2 Kernel 32-bit  JNPR-11.0-20160614.329646_build
{master:0}
EX2300-16>
```

Success!! We can ping the default gateway and can log in using SSH protocol version 2 to our dedicated Ethernet OOB management interface. Now you need to tackle steps 6 through 11 and provision our VLANs, TRUNKS, and L3 Interfaces to connect the switch to the rest of the lab.

ELS ALERT The following commands will differ substantially from the non-ELS commands. Make sure you know your hardware and software and compare it to Table 1.1 in *Chapter 1: ELS Overview*.

```
root@EX2300-16# show vlans
default {
    vlan-id 1;
    l3-interface irb.0;
}
root@EX2300-16# set vlans V100 vlan-id 100 l3-interface irb.100
```

```
root@EX2300-16# delete interfaces ge-0/0/0.0
root@EX2300-16# set interfaces ge-0/0/1.0 family ethernet-switching interface-
mode trunk vlan members V100
root@EX2300-16# set interfaces ge-0/0/2.0 family ethernet-switching interface-
mode trunk vlan members V100
root@EX2300-16# set interfaces irb.100 family inet address 10.100.0.254/24
root@EX2300-16# delete protocols rstp
root@EX2300-16# set protocols rstp interface ge-0/0/1
root@EX2300-16# set protocols rstp interface ge-0/0/2
root@EX2300-16# set protocols rstp bridge-priority 60k
root@EX2300-16# show | compare
[edit interfaces]
-   ge-0/0/0 {
-       unit 0 {
-           family ethernet-switching {
-               storm-control default;
-           }
-       }
-   }
[edit interfaces ge-0/0/1 unit 0 family ethernet-switching]
+      interface-mode trunk;
+      vlan {
+          members V100;
+      }
[edit interfaces ge-0/0/2 unit 0 family ethernet-switching]
+      interface-mode trunk;
+      vlan {
+          members V100;
+      }
[edit interfaces irb]
+    unit 100 {
+        family inet {
+            address 10.100.0.254/24;
+        }
+    }
[edit protocols rstp]
+   bridge-priority 60k;
[edit protocols rstp]
-    interface ge-0/0/0;
-    interface ge-0/0/1;
-    interface ge-0/0/2;
-    interface ge-0/0/3;
… Truncated for brevity
-    interface ge-0/0/23;
-    interface ge-0/1/0;
-    interface xe-0/1/0;
-    interface ge-0/1/1;
-    interface xe-0/1/1;
-    interface ge-0/1/2;
-    interface xe-0/1/2;
-    interface ge-0/1/3;
-    interface xe-0/1/3;
[edit vlans]
+   V100 {
+       vlan-id 100;
+       l3-interface irb.100;
+   }
```

Committing a Configuration

Throughout this chapter you have used the `commit` command to move your candidate configuration into the active configuration. You will be introduced to some other useful commit features throughout this book such as: `commit check`, `commit confirmed`, and `commit comment`. These commands are used where it makes the most sense and they will be pointed out as you progress:

```
root@EX2300-16# commit
configuration check succeeds
commit complete

root@EX2300-16# run ping 10.100.0.1
PING 10.100.0.1 (10.100.0.1): 56 data bytes
64 bytes from 10.100.0.1: icmp_seq=0 ttl=64 time=28.323 ms
64 bytes from 10.100.0.1: icmp_seq=1 ttl=64 time=30.508 ms
^C
--- 10.100.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.323/29.416/30.508/1.092 ms
```

We can ping the VRRP virtual-ip on QFX5100-2, which means that spanning tree has converged on RSTP, and our VLAN 100 is being carried by the TRUNK that we configured on interfaces ge-0/0/1 and ge-0/0/2. Next up are steps 12 and 13, configuring the Layer 3 point-to-point interfaces on your device:

```
root@EX2300-16# set interfaces xe-0/1/0 unit 0 family inet address 10.1.0.9/31
root@EX2300-16# set interfaces ge-0/0/0.0 family inet address 10.1.0.13/31
root@EX2300-16# commit
configuration check succeeds
commit complete
root@EX2300-16# run ping 10.1.0.8
PING 10.1.0.8 (10.1.0.8): 56 data bytes
64 bytes from 10.1.0.8: icmp_seq=0 ttl=64 time=16.651 ms
```

We can ping the end of our p2p interface but we aren't going to do anything with it right now. Notice that `unit 0` was used on one command and then `.0` on the other. Junos allows you to identify the unit as "." instead of spelling it out. You should notice when you do a `show interfaces terse` from the operational CLI that interfaces are displayed with the .0 format. Also, when you want to see the logical interface using a `show` command you will specify it with `.0` or in the case of our irb: `irb.100`. Let's move on and finish up steps 14–18:

```
root@EX2300-16# set routing-options static route 10.100.0.0/24 next-hop 10.100.0.1
root@EX2300-16# wildcard range set interfaces ge-0/0/[0-
24] description "THIS INTERFACE IS FOR LAB USE"
|
root@EX2300-16# commit
configuration check succeeds
commit complete

root@EX2300-16# run show interfaces descriptions
Interface       Admin Link Description
ge-0/0/0        up    up   THIS INTERFACE IS FOR LAB USE
ge-0/0/1        up    up   THIS INTERFACE IS FOR LAB USE
```

```
ge-0/0/2        up    up   THIS INTERFACE IS FOR LAB USE
ge-0/0/3        up    up   THIS INTERFACE IS FOR LAB USE
ge-0/0/4        up    down THIS INTERFACE IS FOR LAB USE

root@EX2300-16# run show interfaces terse | match down | except eth-switch
ge-0/0/4                up    down
ge-0/0/5                up    down
ge-0/0/6                up    down
ge-0/0/7                up    down
…
root@EX2300-16# wildcard range set interfaces ge-0/0/[4-23] disable
root@EX2300-16# show | compare
[edit interfaces ge-0/0/4]
+   disable;
[edit interfaces ge-0/0/5]
+   disable;
…
```

Setting a static route is a common configuration if you want to set a default route use 0/0 next-hop. The `wildcard` command is very handy especially when you start working with 10-member Virtual Chassis with 480 ports, or in the case of our 10-member QFX5100-96S VC, 960 ports! Be careful with the wildcard as you could lock yourself out if you are changing settings on the interface you are currently logged into. Therefore, you want to use `commit confirmed` as done in the example.

Add License

One thing that has not been covered, and this is as good a time as any to address it, is adding an EFL to the system. EFLs are provided by Juniper to its partners and customers as either a hard paper license or in digital format. If you know you have a license and just misplaced it, you can search for it on the Juniper License Manager homepage at https://lms.juniper.net/lcrs/license.do.

The license file is just a text file that looks like this:

```
Serial No :              NV0216340092
Features :               EX-24-EFL : Enhanced feature license for 24 port EX2200, EX3300,
EX2300, and EX3400 series switch including IBM OEM SKUs
Issue Date :             04-AUG-2017
License Key :
Junos942358 aeaqia qmjzld amrrgy ztimbq hezama uqiyds
            qcaztl qbljc6 mpk3zq luzz23 bizbte 47kvo6
            bzfcw6 eufdek t262hr ettskh nbqaic j4
```

You will immediately know if you need a license on your system when you do a commit, therefore:

```
EX3400-10_11-VC# commit
[edit protocols]
  'ospf'
    warning: requires 'ospf2' license
[edit routing-instances DO_VRF instance-type]
  'instance-type virtual-router'
```

```
   warning: requires 'virtual-router' license
fpc0:
configuration check succeeds
fpc1:
commit complete
fpc0:
commit complete
```

NOTE Juniper doesn't just lock you out of a feature. We know you may be waiting for a purchase order to go through or you are mocking something up in advance so there is a certain amount of leniency on licensing and Juniper believes our customers will do the right thing. It's an end user ethics thing at this point. So do the right thing.

To apply our license, you can paste it into the terminal or secure copy the file to the device and add it from there. Let's just use the copy paste method:

```
EX3400-10_11-VC> show system license
License usage:
                                Licenses     Licenses    Licenses    Expiry
  Feature name                       used    installed      needed
  ospf2                                 1            0           1    invalid
  virtual-router                        1            0           1    invalid

EX3400-10_11-VC> request system license add terminal
[Type ^D at a new line to end input,
 enter blank line between each license key]
Serial No :              NV0216340092
Features :              EX-24-EFL : Enhanced feature license for 24 port EX2200, EX3300, EX2300, and
EX3400 series switch including IBM OEM SKUs
Issue Date :             04-AUG-2017
License Key :
Junos942358 aeaqia qmjzld amrrgy ztimbq hezama uqiyds
            qcaztl qbljc6 mpk3zq luzz23 bizbte 47kvo6
            bzfcw6 eufdek t262hr ettskh nbqaic j4
Junos942358: successfully added
add license complete (no errors)
```

Once you copy and paste the contents of the license into the terminal you have to use CTRL-D to end the input and then, if all is well, you will see it successfully added and no error statements.

```
EX3400-10_11-VC> show system license
License usage:
                                Licenses     Licenses    Licenses    Expiry
  Feature name                       used    installed      needed
  ospf3                                 0            1           0    permanent
  ripng                                 0            1           0    permanent
  ospf2                                 1            1           0    permanent
  bfd-liveness-detection                0            1           0    permanent
  unicast-rpf                           0            1           0    permanent
  virtual-router                        1            1           0    permanent
  igmp                                  0            1           0    permanent
  pim-mode                              0            1           0    permanent
  pim-ssm                               0            1           0    permanent
  connectivity-fault-management         0            1           0    permanent
```

```
  vrrp                                0          1          0    permanent
  dot1q-tunneling                     0          1          0    permanent
  svlan                               0          1          0    permanent
  services-rpm                        0          1          0    permanent
  juniper-msdp                        0          1          0    permanent
  multicast-listener-discovery        0          1          0    permanent
  gr-ifd                              0          1          0    permanent
Licenses installed:
  License identifier: Junos942358
  License version: 4
  Valid for device: NV0216340092
  Features:
    extended-feature-list - Licensed extended feature list
     permanent
```

Now you are licensed and legal and there are no more annoying messages when you commit.

Rollback

The rollback command is one of the features that differentiates Juniper Networks and the Junos OS from other vendors. There are numerous scenarios where you might want to use the rollback command—a couple of scenarios utilize it here—but please, don't let that be the end of your rollback learning experience.

Real World Scenario 1:

Let's assume someone in the NOC was directed to make a change to a routing protocol or a route in the middle of the night when there are very few users on your network. For some reason, your phone starts ringing at 0500 in the morning, because one of the early riser finance managers can't get to her reports and it's the end of the quarter. You drag yourself out of bed and pad over to your trusty laptop and log in through the SSL VPN to see in the logs that someone made a change on a switch around 1am in the morning. You know this is a huge mistake and that in a couple of hours all hell will break lose if this is not fixed.

Luckily you can see the commits and the times by using `the > show system commit` command:

```
ACCESS-1> show system commit
0   2017-05-05 01:00:58 UTC by tier1 via cli
    Removed interface ge-0/0/0.0 from OSPF area 0.0.0.0//NOC
1   2017-05-01 19:19:28 UTC by tier2 via cli
2   2017-05-01 17:49:30 UTC by tier2 via cli
```

Perfect – the NOC used the tier1 account and followed standard operating procedure (SOP) to include a comment with the commit statement when making changes. So you can see the time the last commit was made, as well as what was performed.

Now you can use the `> show configuration | compare rollback 1` command to see what

changes were made. (Note that Junos will hold 50 commits numbered 0-49 with 0 being the current active configuration):

```
ACCESS-1> show configuration | compare rollback 1
[edit protocols ospf area 0.0.0.0]
-     interface ge-0/0/0.0 {
-         interface-type p2p;
-     }
```

Just as you suspected, the interface they removed was the wrong one. You now know that was the only change implemented and that it is indeed the problem and you can use the rollback command to fix it:

```
ACCESS-1> edit
Entering configuration mode

ACCESS-1# show | compare rollback 1
[edit protocols ospf area 0.0.0.0]
-     interface ge-0/0/0.0 {
-         interface-type p2p;
-     }

ACCESS-1# rollback 1
load complete

ACCESS-1# show | compare
[edit protocols ospf area 0.0.0.0]
+     interface ge-0/0/0.0 {
+         interface-type p2p;
+     }
      interface lo0.0 { ... }

ACCESS-1# commit comment "Tier2 rolling back the ospf change from last night"
configuration check succeeds
commit complete
```

To fix the problem you assumed edit mode, did a quick `show | compare rollback 1` to see what changes would be made, performed the `rollback 1` , and did a `show | compare` to see that the interface will indeed be added back to OSPF, and then used `commit comment` to make a note in the system commit log that you rolled back the change.

In this scenario, you not only see the power of rollback command but you were also introduced to the `show system commit` operational command to see a list of up to 50 changes on the system and when they were changed. You were also introduced to the `commit comment` feature that allows for comments to be made when committing.

Real World Scenario 2:

This is probably the most common scenario and is used often. Imagine you were making updates by adding interfaces to a new marketing section in your company and when you were right in the middle of those changes you received a phone call and they changed the ports that were going to be used.

Not only did you get distracted from what you were originally doing, now you have to make a bunch of changes to the changes you were already making. You forgot where you were and everything has to be undone. You can actually use `delete` commands to undo everything one at a time or you can use `rollback`.

You perform a quick `show | compare` to see all of the work in your candidate configuration that needs to be removed. Since you have not issued a commit during your session and it is still in the candidate configuration you can simply use the rollback command. Issue `# rollback` and do a `show | compare` and you should see that there are no deltas between the active and candidate configuration. You can issue a `commit comment` here or simply exit configuration mode and move on to your task.

Using `rollback` is a feature that you will learn to love as a network engineer. It's a huge time saver.

MORE? You have only been introduced to a couple of scenarios and a subset of the rollback command. To find out more about the rollback command go to: https://www.juniper.net/documentation/en_US/junos/topics/reference/command-summary/rollback.html.

Getting Help

The Junos OS is based on FreeBSD, which is a Linux operating system. If you have been around Linux at all then you should be familiar with the man pages. The man pages allow users to get help on specific commands and topics. Fortunately, Juniper has integrated that same feature into the operating system and you can get help on everything right from the command line. So instead of following a link to the rollback technical documentation you can invoke help directly from the CLI. Let's see if we can get help on the help command:

```
> help apropos help
…
help
    Provide help information
help apropos
    Find help information about a topic
help apropos <topic>
    Help topic, regular expression, or command
help topic
    Help for high level topics
help topic system op-scripts description
…
```

As you can see you can even get help on the `help` command. Using `help apropos` and then specifying the topic you want help with, will give you a list of commands for that topic. Let's look at something more interesting, such as OSPF:

```
> help apropos ospf
show route protocol ospf
    Open Shortest Path First
```

```
show route protocol ospf2
    Open Shortest Path First Version 2
show route protocol ospf3
    Open Shortest Path First Version 3
show ospf
    Show Open Shortest Path First information
show ospf overview
    Show overview of OSPF information
…
```

The list of commands for OSPF is quite lengthy. As stated earlier, a format of help, very much like the man pages in Linux, is included as well. Instead of `help apropos` you can use `help topic`. The output from these can get quite lengthy, so let's pick something small but useful for the book. A question that is often asked by field engineers is how to automatically back up the configuration to a remote system. For that topic, you can issue:

```
> help topic system configuration
```

Configuring the Switch to Transfer Its Currently Active Configuration to an Archive

If you want to back up your device's current configuration to an archive site, you can configure the switch to transfer its currently active configuration by FTP or secure copy (SCP) periodically or after each commit.

To configure the router or switch to transfer its currently active configuration to an archive site, include statements at the `[edit system archival configuration]` hierarchy level:

```
[edit system archival configuration]
archive-sites {
    ftp://username<:password>@host-address<:port>/url-path;
    scp://username<:password>@host-address<:port>/url-path;
}
    transfer-interval interval;
    transfer-on-commit;
+-------------------------------------------------------------------------+
|     |When specifying a URL in a Junos OS statement using an IPv6 host  |
|Note:|address, you must enclose the entire URL in quotation marks (“”)  |
|     |and enclose the IPv6 host address in brackets ([ ]). For example, |
|     |”ftp://username<:password>@[ipv6-host-address]<:port>/url-path”   |
```

Related-Topics

So not only did we get help on the topic it gave us the instructions on how to implement it, where the command is placed in the hierarchy, and even some tips. You never had to leave the CLI to open a browser for this and the information is very accurate. You should get to know the help apropos and help topic commands,

especially if you're new to Junos. If you are not new to Junos but are new to ELS commands help is at your fingertips:

```
> help topic interfaces interface-mode-trunk
```

Configuring a Logical Interface for Trunk Mode

As an alternative to configuring a logical interface for each VLAN, enterprise network administrators can configure a single logical interface to accept untagged packets or packets tagged with any VLAN ID specified in a list of VLAN IDs. Using a VLAN ID list conserves switch resources and simplifies configuration. A logical interface configured to accept packets tagged with any VLAN ID specified in a list is called a *trunk interface* or *trunk port*. Trunk interface configuration is supported on MX Series routers only. Trunk interfaces support integrated routing and bridging (IRB).

To configure a logical interface to accept any packet tagged with a VLAN ID that matches the list of VLAN IDs, include the `interface-mode` statement and specify the `trunk` option:

```
interface-mode trunk;
```

You can include this statement at the following hierarchy levels:

```
    * [edit interfaces interface-name unit logical-unit-number family bridge]
    * [edit logical-systems logical-system-name interfaces interface-name unit logical-unit number
family bridge]
```

One more help command that deserves attention is `help reference`. The `help reference` command provides summary information about a specific command or area of the config:

```
> help reference dot1x authentication-profile-name
                          authentication-profile-name
    Syntax
   authentication-profile-name access-profile-name;
    Hierarchy Level
   [edit protocols dot1x authenticator]
    Release Information
   Statement introduced in Junos Release 9.3.
    Description
   Specify the RADIUS authentication profile to use for user authentication
   when establishing an IEEE 802.1x Port-Based Network Access Control (dot1x)
   connection.
    Usage Guidelines
   See "Configuring IEEE 802.1x Port-Based Network Access Control".
    Required Privilege Level
   interface--To view this statement in the configuration.
   interface control--To add this statement to the configuration.
    Related Topics
   dot1x, authenticator
```

System Snapshot

Juniper introduced the Resilient Dual root partition in Junos version 10.4R3 and 11.4R1. This turned the original 3-slice partition into a 4-slice partition:

```
EX4300-6_7-VC> show system storage partitions
fpc0:
--------------------------------------------------------------------------
Boot Media: internal (da0)
Active Partition: da0s1a
Backup Partition: da0s2a
Currently booted from: active (da0s1a)
Partitions information:
  Partition  Size   Mountpoint
  s1a        316M   /
  s2a        324M   altroot
  s3d        887M   /var/tmp
  s3e        170M   /var
  s4d        116M   /config
fpc1:
--------------------------------------------------------------------------
Boot Media: internal (da0)
Active Partition: da0s1a
Backup Partition: da0s2a
Currently booted from: active (da0s1a)
Partitions information:
  Partition  Size   Mountpoint
  s1a        316M   /
  s2a        324M   altroot
  s3d        887M   /var/tmp
  s3e        170M   /var
  s4d        116M   /config
```

Juniper moved the /var partition into its own slice to reduce the possibility of corruption issues. The idea behind this is that /var has much more read and write activity, and by moving it to its own slice it will isolate it from the configuration and the operating system. If the switch fails to boot, the system automatically boots from the alternate root partition. (If the switch fails to boot from the active root partition and instead boots from the alternate root partition, an alarm is triggered.) You will also get a message upon login that the system has booted from the alternate partition.

From the factory both the / root and altroot partitions are loaded with the same Junos so when you upgrade it is necessary to perform a system snapshot to copy the image and configuration to the altroot partition. Otherwise, you could boot up on an old image and an old configuration previously stored in the altroot partition.

You can see the snapshot and the last time a snapshot was performed by issuing the following on any dual root partition device:

```
EX4300-6_7-VC> show system snapshot media internal
fpc0:
--------------------------------------------------------------------------
Information for snapshot on       internal (/dev/da0s1a) (primary)
```

```
Creation date: Apr 18 21:34:50 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
Information for snapshot on       internal (/dev/da0s2a) (backup)
Creation date: Apr 18 21:35:33 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
fpc1:
--------------------------------------------------------------------------
Information for snapshot on       internal (/dev/da0s1a) (primary)
Creation date: Apr 13 16:19:25 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
Information for snapshot on       internal (/dev/da0s2a) (backup)
Creation date: Apr 13 16:20:09 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
```

You can see from this output that both switches making up this Virtual Chassis (VC) have dual root partitions and the information shows the last date of the snapshot that was performed. We want these to be the identical so let's perform a `snapshot` on the EX4300 VC:

```
EX4300-6_7-VC> request system snapshot slice alternate all-members
System may go unstable if module traces or syslog messages are enabled during snapshot.
It is recommended to disable all debug logging.
Do you wish to continue? [yes,no] (no) yes
fpc0:
--------------------------------------------------------------------------
Formatting alternate root (/dev/da0s2a)
...
Copying '/dev/da0s1a' to '/dev/da0s2a' .. (this may take a few minutes)
The following filesystems were archived: /
fpc1:
--------------------------------------------------------------------------
Formatting alternate root (/dev/da0s2a)...
Copying '/dev/da0s1a' to '/dev/da0s2a' .. (this may take a few minutes)
The following filesystems were archived: /
```

Now we can take a look at the media to see the status of the `snapshot`:

```
EX4300-6_7-VC> show system snapshot media internal
fpc0:
--------------------------------------------------------------------------
Information for snapshot on       internal (/dev/da0s1a) (primary)
```

```
Creation date: Apr 18 21:34:50 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
Information for snapshot on       internal (/dev/da0s2a) (backup)
Creation date: Jul 7 01:25:22 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
fpc1:
--------------------------------------------------------------------------
Information for snapshot on       internal (/dev/da0s1a) (primary)
Creation date: Apr 13 16:19:25 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
Information for snapshot on       internal (/dev/da0s2a) (backup)
Creation date: Jul 7 01:26:47 2017
Junos version on snapshot:
  jdocs-ex: 14.1X53-D40.8
  junos  : ex-14.1X53-D40.8
  junos-ex-4300: 14.1X53-D40.8
  jweb-ex: 14.1X53-D40.8
```

By enabling the `auto-snapshot` feature you can be sure that the snapshot will automatically take place on system commits:

```
EX4300-6_7-VC# set system auto-snapshot
EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
```

That is all there is to it. It may seem trivial but it is part of the care and maintenance of the network to make sure that our High Availability (HA) features are going to provide HA and recovery if needed.

TIP You can also `snapshot` to external media such as a USB and that USB can be used to upgrade other devices. For more information please refer to: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/ex-series-system-snapshot-boot-directory.html.

It should be noted that the EX9200 cannot auto-snapshot and the EX4550 auto-snapshot is not enabled by default. It should also be noted that the EX2300 does not use a dual root partition but it does perform auto-snapshot. However, the snapshot is different because it creates a snap file like so:

```
EX2300> request system snapshot
fpc0:
--------------------------------------------------------------------------
NOTICE: Snapshot snap.20170706.204138 created successfully
```

The snapshots are stored in the packages directory:

```
EX2300> file list /packages/sets/
/packages/sets/:
active/
optional/
previous/
snap.20161212.143637/
snap.20170706.203845/
snap.20170706.204138/
```

For more in-depth information on dual root partitions please refer to: https://www.juniper.net/documentation/en_US/junos/topics/concept/switch-resilient-dual-root-partitions.html#jd0e132.

Rescue Configuration

The rescue configuration has to be set in the Junos OS or you will receive a system alarm stating that it hasn't been set. It is a non-impacting alarm but it can drive those engineers with a touch of OCD crazy if all the lights aren't green and the alarms aren't cleared:

```
EX4300-6_7-VC> show system alarms
1 alarms currently active
Alarm time               Class  Description
2017-06-01 17:56:48 UTC  Minor  Rescue configuration is not set
```

This is a simple fix and it's an operational command:

```
EX4300-6_7-VC> request system configuration rescue save
```

And then you can view the rescue configs that has been saved by issuing the `show` command:

```
EX4300-6_7-VC> show system configuration rescue
## Last changed: 2017-07-07 01:35:05 UTC
version 14.1X53-D40.8;
groups {
    POC_Lab {
        system {
            host-name EX4300-6_7-VC;
            backup-router 172.16.0.1;
            authentication-order [ radius password ];
            root-authentication {
                encrypted-password "$1$mlMH7KjC$gOLJ9YI4KWlwDMgAfc5iT1"; ## SECRET-DATA
            }
            name-server {
                8.8.8.8;
… Truncated for brevity
```

And finally, you can verify that the alarm is cleared:

```
EX4300-6_7-VC> show system alarms
No alarms currently active
```

The rescue configuration is valuable for golden configurations or a known good baseline as you can rollback to the rescue:

```
EX4300-6_7-VC# rollback rescue
load complete
EX4300-6_7-VC# show | compare
EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
```

Since you just saved the entire configuration as the rescue configuration there will not be any deltas in between the active and candidate configurations.

This is another one of those housekeeping items that most people set and forget. If you want to make the most out of this feature it's best to baseline the network occasionally and create a rescue configuration on all of your Juniper devices. For more information on rescue configuration please refer to: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/junos-software-rescue-configuration-creating-restoring.html.

Summary

This chapter introduced you to the basic commands that you will see during a typical installation of a new device or a device that has been zeroized and set to its factory-default configuration.

By no means is this an all-inclusive list of commands! If you are new to Junos or just picked this book up to get EX Series information, then kudos to you! Keep learning as much as you can about operational and configuration commands. That will help you become more confident and more attuned to Junos across all Juniper Network products.

To get started with Junos see the Day One books in the "Fundamentals" series, starting with *Day One: Exploring the Junos CLI, 2nd Edition* at: http://www.juniper.net/dayone.

Chapter 4

Virtual Chassis

What is a Virtual Chassis (VC)

EX Series switches have the ability to be connected together, via a QSFP+ or SFP+ Optic, to create a single, logical EX Virtual Chassis (VC). You create a VC by connecting two or more switches via the 10G Direct-Attach-Copper (DAC) SFP+ ports, or the 40G DAC using the QSFP+ port. The QSFP+ port is a 40Gig port (80G Full duplex) that can take either a Direct-Attach Copper (DAC) or a 40G Optic. Do not mistake this for the VCP port, on the back of the EX4200 series switch, as that cable is not compatible with the new VC Technology. This book only focuses on ELS switches and one of the ways you can identify an ELS capable switch is the Quad Small Form-factor Pluggable Plus (QSFP+) transceiver on the back of the device, or in the case of the EX4600 and EX9200, on the front.

MORE? The Day One Book *Configuring EX Series Ethernet Switches, Third Edition* by Yong Kim does a great job of explaining Virtual Chassis on Non-ELS EX Series switches and I recommend you read that if you are supporting Non-ELS switches such as EX2200, EX3200, EX3300, EX4200 or EX4500. If you are really serious and want to get certified in Juniper Enterprise, then you should pick up *Junos Enterprise Switching, A Practical Guide to Junos Switches and Certification,* by Harry Reynolds and Doug Marschke, published by O'Reilly Media.

MORE? Each of the EX Switches are designed for specific areas of the network and it makes sense that each model will have different capabilities when it comes to pluggable optics or transceivers. The following URL will give you a good jumping point to find the compatible optics and the relevant information for each of those optics that you can plug into your specific model: http://www.juniper.net/techpubs/en_US/release-independent/junos/topics/reference/specifications/optical-interface-

ex-series-support.html. Also, please take a look at the QSFP+ table for 40G DAC support on EX Switches: https://www.juniper.net/techpubs/en_US/release-independent/junos/topics/reference/general/cable-ex-series-qsfp-plus-direct-attach.html.

The benefit of a Virtual Chassis (VC) is that you get two routing engines (RE0 and RE1) and the flexibility to expand from one to ten switches in a VC (four switches when using EX2300 VC). Once you have a VC you also now have the added benefit of being able to configure some of the High Availability (HA) mechanisms, such as Non-Stop Routing (NSR), Non-Stop Bridging (NSB), Graceful Routing-Engine Switchover (GRES), and Non-Stop Software Upgrades (NSSU) or In Service Software Upgrades (ISSU). You have also improved your switch to touch-point ratio by potentially reducing ten switches into one managed VC.

Another benefit of a Virtual Chassis over a real chassis is that you are not confined to a single rack, row, building or campus with a VC. You can expand a VC across multiple racks (think Top of Rack) or multiple rows in a datacenter. You can also have a VC that spans multiple buildings and if that isn't enough you can even have VCs that will span multiple campuses or cities or states. The point is with a VC you are not confined to that 19-inch rack where the chassis is installed. VCs are flexible and you can have a single point of management for multiple physical locations, while cutting cable costs and operations.

NOTE Some network vendors use "Stacked" solutions. These are not the same as Junipers Networks Virtual Chassis. With Virtual Chassis, you can extend your switch with true 40G up to 70km. The other vendors are limited to 3 meters. So, when you are looking at value-added networks and future proofing networks that information needs to be taken into consideration.

It is important to know the depth of the Virtual Chassis solution for each model. Table 4.1 will guide you on the number and models that can be stacked. Notice that we do have some mixed VC solutions and that Virtual Chassis Fabric (VCF) is included but *should not* be confused with a standard VC. VCF is an *entirely different architecture* than the standard Virtual Chassis solution.

Table 4.1 *Virtual Chassis Hardware and Software Matrix*

EX-VC Model	Max Devices	Min Junos	Junos
EX2200	4	12.2R1	NON-ELS
EX2300	4	15.1X53-D50	ELS
EX3300	10	12.2R1	NON-ELS
EX3400	10	15.1X53-D50	ELS
EX4200	10	9.0R1	NON-ELS

EX4300	10	13.2X50-D10	ELS
EX4300 Mixed VCF	20	13.2X50-D20	ELS
EX4500	10	11.4R1	NON-ELS
EX4550	10	12.2R1	NON-ELS
EX4600	10	13.2X51-D25	ELS
EX4200/4500 (mixed VC)	9	11.4R1	NON-ELS
EX4200/4500/4550 (mixed VC)	10	12.2R1	NON-ELS
EX4300/4600 (mixed VC)	10	13.2X51-D25	ELS
EX4500/4550 (mixed VC)	10	12.2R1	NON-ELS
EX9200	2	13.2R2	ELS

NOTE The EX2300 requires an EX2300-VC License when they are placed in a VC. This is a delta from all the other EX switches that do not require a license to form a VC.

Virtual Chassis (VC) Cabling

To recap, build Virtual Chassis (VC) by connecting a 40G DAC to the QSFP+ port and then create a ring or braided ring with the DAC cables. You can also create an extended Virtual Chassis, with 10G or 40G optics, where you have one or more members in a remote location. In addition, you can create a single Top of Rack (ToR) switch across ten racks in your data center.

Figure 4.1 is a stock picture of a 40G passive DAC cable. An Active DAC cable is also available for purchase. Active indicates that there is power applied to assist with attenuation over distances.

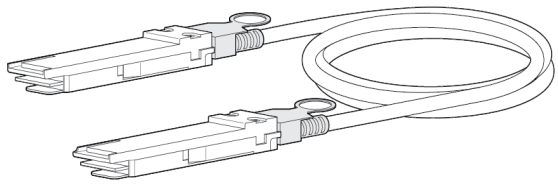


Figure 4.1 QFX-QSFP-DAC-1M

The EX2300 only stacks to four deep and it uses a 10G DAC cable versus a 40G DAC. Depending on your needs, the 10G DAC comes in different lengths. You can find the specifications for the DAC cable here: https://www.juniper.net/techpubs/en_US/release-independent/junos/topics/reference/general/cable-ex-series-sfp-plus-direct-attach.html. Figure 4.2 shows a stock picture of a 10G DAC.

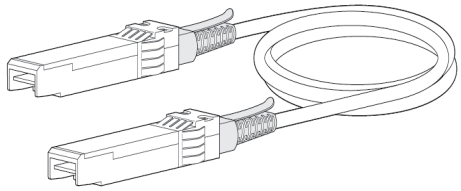


Figure 4.2 *EX-SFP-10GE-DAC-1M*

If you can't tell from the pictures, the 40G is wider than the 10G DAC. Both of these DACs are hot swappable, hot insert cables so if you need to replace a line card (LC) in a VC you can do that but make sure you update the virtual chassis configuration in the master RE with the new serial number and role, or remove that line card from the configuration. Figure 4.3 is an example of a 4X - EX4300 VC with a ring topology using 40G DAC.

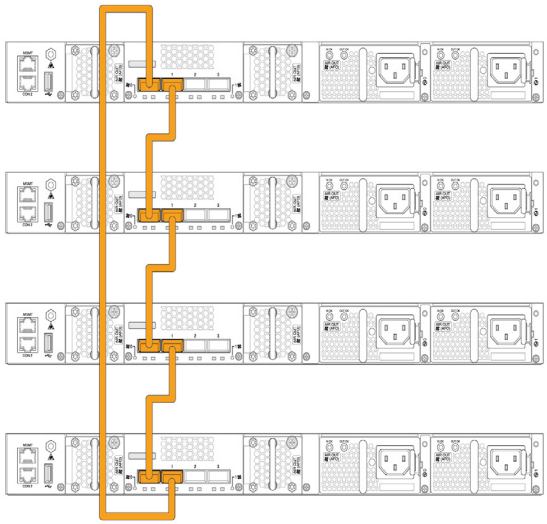


Figure 4.3 *EX Ring Topology on EX4300*

As you can see from Figure 4.3, from top to bottom, each switch is connected to the next, and then the last switch is connected to the first switch, to create a ring. It is common, when you stack over four switches, to use 50cm DAC for the all switches and then a 1-meter DAC to connect the last switch to the first. Make sure that you don't come up short on your cables! If you have an extended VC you have the option of 1-meter through 15-meter DACs and with active cables you can extend to 30-meters. You also have the option to extend your VC using 10G or 40G over fiber, that will be discussed shortly.

TIP Before you start plugging in the QSFP+ DAC cables to create your 10-member Virtual Chassis you have some work to do. All of the members in the VC should be on the same version of Junos before you start plugging in the DACs. If the master routing engine is on 15.1 and the line card is on 14.1 it will not be seen as part of the Virtual Chassis output and will not become part of the VC. You will have the best experience if you plan ahead and load the correct Junos on all members of the VC. An easy way to do this is set up a ZTP server with the correct image, zeroize the line card if not already on factory configuration, and then allow it to automatically pull the image and upgrade itself from the ZTP server.

TIP If you create a two-switch Virtual Chassis be sure to `disable split detection` or you could end up with two routing engines that both think they are the master and both advertising default gateways, protocols, etc. Let's just say it is not a good scenario for a network. The good thing is we just identified how to fix it and here is the command to do just that:

```
# set virtual-chassis no-split-detection
# commit comment "enabling no-split-detection on my 2 member vc"
```

WARNING! You *do not* want to implement `no-split-detection` on VCs with more than two members. *Only* disable `split-detection` on two-member VCs. Hopefully that is clear.

There are numerous ways to cable the devices and you want to be aware of separation of REs in the VC. With a typical ring topology, you want to have a line card (LC) between the REs. In a braided ring, you would have the REs next to each other as shown in Figure 4.4.

TIP When you look at the output of a `virtual-chassis` it will refer to the switch as member 0 through 9, or in the case of the RING VC implementation, member 0 through member 4. Just be aware that Juniper numbering starts at 0 for pretty much everything – interfaces, members, VCP ports, FPC numbers etc.

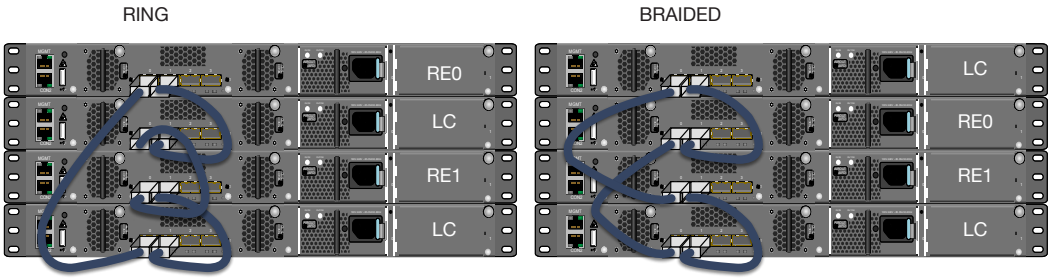


Figure 4.4 Ring and Braided QSFP+ Topology on EX4300

Virtual Chassis (VC) Configuration

So far we have discussed the different types of Virtual Chassis ports (VCPs) and the 10G and 40G direct attach copper (DAC) cables. We've discussed disabling `split-detection` on 2-member Virtual Chassis (VC) and leaving it enabled for VCs larger than two members. We have also discussed the fact that you want all of your switches on the same version of Junos before plugging in the DAC cables.

Now, if you are a lucky, and that's just the way you roll, you can plug in all of DAC cables, to form the backplane of the VC, and then power up all the switches and you might actually get lucky and have all of the switches in the exact location and in the exact role that you – well you didn't plan, you just got lucky. Normally, the first switch you boot will become the master routing engine and it will be member0.

TIP Later we'll discuss `auto-sw-upgrade` that facilitates adding members to a VC and they will be able to pull the correct Junos version to become members of the VC.

If you did use the 'power up and hope for the best' method, however, the first switch you boot will become the master routing engine and it will be member0 and then the others will become members (1-9) as they reach operational mode. The problem is that the time it takes for a switch to reach operational mode and grab the next member-id is not pre-determined and you can end up with a line card on the wrong member ID in the stack because another member booted faster.

Most of us, however, would never get that lucky, and because network engineers are control freaks, we need a way to pre-provision the individual switches to take on the role of routing engine or line card. Thankfully, we have such a mechanism and all we need is the serial number from the device to configure its role in the VC.

Let's assume the serial numbers on the 5-member VC in a ring topology in the picture above are AAAA through EEEE. Armed with that information we know that we want RE0 or AAAA to be member 0 and CCCC to be member 2 and RE1. Here is an example of that configuration as a pre-provisioned configuration:

```
# set virtual-chassis member 0 role routing-engine serial-number AAAA
# set virtual-chassis member 1 role line-card serial-number BBBB
# set virtual-chassis member 2 role routing-engine serial-number CCCC
# set virtual-chassis member 3 role line-card serial-number DDDD role line-card
# set virtual-chassis member 4 role line-card serial-number EEEE role line-card
# set virtual-chassis preprovisioned
# show | compare
# commit check
# commit comment "Creating the preprovisioned configuration for my VC"
```

TIP The master switch is usually member-id 0. Member IDs are maintained through reboots and you can renumber the member-ids from the master routing engine.

To renumber a member-id you utilize the operational `request virtual-chassis renumber` command:

```
EX4300-6_7-VC> request virtual-chassis ?
Possible completions:
  mode                 Set a member's mode (Warning: member's mode must be consistent)
  reactivate           Make active from inactive-split mode
  recycle              Recycle member ID
  renumber             Change member ID
  vc-port              Set or delete member›s virtual chassis ports
```

As you can see from the output you have several options available with this command. The `renumber` command is not available in pre-provisioned virtual chassis. So, if you had a VC and wanted to change the member-id you would first need to remove the `preprovisioned` statement. For more information on the `request virtual-chassis` commands please refer to: https://www.juniper.net/documentation/en_US/junos/topics/reference/command-summary/request-virtual-chassis-renumber.html.

TIP *Junos Enterprise Switching, A Practical Guide to Junos Switches and Certification*, by Harry Reynolds and Doug Marschke does a very good job of covering Virtual Chassis along with everything else that we have covered here. The only thing that book is missing are the ELS commands.

For troubleshooting purposes, you can also use traceoptions that you can turn on and off through `apply-groups`. Here is a sample configuration using `traceoptions` to monitor the status of the Virtual Chassis (just remember to turn off traceoptions when not in use so it does not impact performance):

```
# set groups VCTRACE virtual-chassis traceoptions file vctrace.log
# set groups VCTRACE virtual-chassis traceoptions file size 1m
# set groups VCTRACE virtual-chassis traceoptions file files 3
# set groups VCTRACE virtual-chassis traceoptions file world-readable
# set groups VCTRACE virtual-chassis traceoptions flag all
# set apply-groups VCTRACE
# show | compare
# commit check
# commit comment "Creating the traceoptions file for my VC"
```

TIP Notice that the traceoptions creates a file named `vctrace.log` and it will be placed under `/var/logs` on the operating system. You can view the logs from the operational CLI by simply running this command:

```
> show log vctrace.log
```

If you remove the `set apply-groups VCTRACE` statement you disable that group and it will no longer collect logs. To remove the VCTRACE group from `apply-groups` issue:

```
> edit
# delete apply-groups VCTRACE
# commit comment "VCTRACE no longer active"
# exit
> show log vctrace.log
```

Let's take a look at the output from a real Virtual Chassis (VC). Refer to the lab topology used for this book. Note that all of the VCs are only 2-members. What is one of the commands we should see in the configuration? You guessed it `no-split-detection`. Now let's see what we have on EX4600-4:

```
EX4600-4_5-VC> show virtual-chassis status
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                                Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    TC3714350038 ex4600-40f     129   Master*      N  VC   1  vcp-255/0/24
1 (FPC 1)  Prsnt    TC3714350071 ex4600-40f     129   Backup       N  VC   0  vcp-255/0/24
```

There are four possible states for the status of a VC member.

1. `Prsnt` (Present): Indicates that member is available and has established physical and logical connectivity to the VC and it can be managed as part of the VC.
2. `NotPrsnt` (Not Present): Indicates this member is not physically or logically connected to the VC and that it cannot be managed as part of the VC.
3. `Inactive`: Indicates that the member is physically connected but not logically. The member can still be managed as part of the VC but its logical connection needs work. This could indicate a software mismatch.
4. `UnPrvsnd` (Not Provisioned): indicates that a member's information is not in the pre-provisioned configuration, does not present itself physically or logically, and cannot be managed as part of the VC.

Now let's look at the actual configuration and see how it was pre-provisioned:

```
EX4600-4_5-VC> show configuration | display set | match virtual-chassis
set groups POC_Lab virtual-chassis preprovisioned
set groups POC_Lab virtual-chassis no-split-detection
set groups POC_Lab virtual-chassis member 0 role routing-engine
set groups POC_Lab virtual-chassis member 0 serial-number TC3714350038
set groups POC_Lab virtual-chassis member 1 role routing-engine
set groups POC_Lab virtual-chassis member 1 serial-number TC3714350071
```

Let's look at the version of Junos running on each of the members in our VC:

```
EX4600-4_5-VC> show version all-members brief
fpc0:
--------------------------------------------------------------------------
Hostname: EX4600-4_5-VC
Model: ex4600-40f
Junos: 14.1X53-D40.8
Junos Base OS boot [14.1X53-D40.8]
Junos Base OS Software Suite [14.1X53-D40.8]
Junos Online Documentation [14.1X53-D40.8]
Junos Crypto Software Suite [14.1X53-D40.8]
Junos Kernel Software Suite [14.1X53-D40.8]
Junos Packet Forwarding Engine Support (qfx-ex-x86-32) [14.1X53-D40.8]
Junos Routing Software Suite [14.1X53-D40.8]
Junos SDN Software Suite [14.1X53-D40.8]
Junos Web Management Platform Package [14.1X53-D40.8]
Junos Enterprise Software Suite [14.1X53-D40.8]
Junos py-base-i386 [14.1X53-D40.8]
Junos Host Software [14.1X53-D40.8]
fpc1:
--------------------------------------------------------------------------
Hostname: EX4600-4_5-VC
Model: ex4600-40f
Junos: 14.1X53-D40.8
Junos Base OS boot [14.1X53-D40.8]
Junos Base OS Software Suite [14.1X53-D40.8]
Junos Online Documentation [14.1X53-D40.8]
Junos Crypto Software Suite [14.1X53-D40.8]
Junos Kernel Software Suite [14.1X53-D40.8]
Junos Packet Forwarding Engine Support (qfx-ex-x86-32) [14.1X53-D40.8]
Junos Routing Software Suite [14.1X53-D40.8]
Junos SDN Software Suite [14.1X53-D40.8]
Junos Web Management Platform Package [14.1X53-D40.8]
Junos Enterprise Software Suite [14.1X53-D40.8
Junos py-base-i386 [14.1X53-D40.8]
Junos Host Software [14.1X53-D40.8]
```

You can also check the actual Virtual Chassis ports (VCPs) form the command line:

```
EX4600-4_5-VC> show virtual-chassis vc-port
fpc0:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
PIC / Port
0/1         Configured        -1    Up           10000
0/24        Configured        -1    Up           40000        1   vcp-255/0/24
fpc1:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
PIC / Port
0/24        Configured        -1    Up           40000        0   vcp-255/0/24
```

Virtual Chassis Port Numbering

One of the things that most people struggle with when first learning Virtual Chassis is how the ports are numbered. For this section, let's take our EX4300-VC and look at our actual port numbers and how those are devised. If you remember from the previous chapter, you can stack up to ten switches with the EX4300, and can even run a mixed-mode VC where you can add EX4600s.

NOTE When stacking EX4600s with EX4300s, the EX4600s have to be the master and backup routing engines.

Reviewing the output from the `show chassis hardware` command is the easiest way to determine your port numbering.

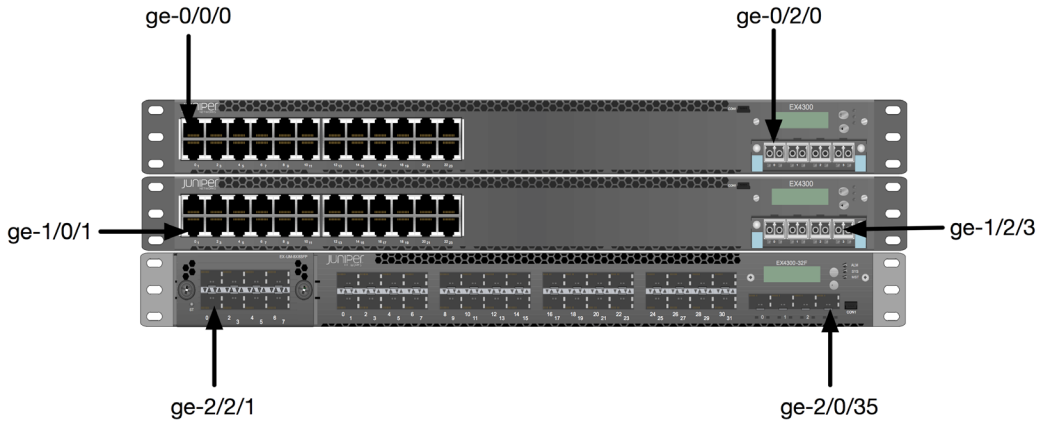


Figure 4.5 Virtual Chassis Port Numbering

```
EX4300-6_7-VC> show chassis hardware
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                PG3713290002      Virtual Chassis
Routing Engine 0 REV 06   650-044936   PG3713290062      EX4300-24T
Routing Engine 1 REV 06   650-044936   PG3713290002      EX4300-24T
FPC 0            REV 06   650-044936   PG3713290062      EX4300-24T
  CPU                     BUILTIN      BUILTIN           FPC CPU
  PIC 0          REV 06   BUILTIN      BUILTIN           24x 10/100/1000 Base-T
  PIC 1          REV 06   BUILTIN      BUILTIN           4x 40GE QSFP+
    Xcvr 0       REV 01   740-038623   MOC15306230995    QSFP+-40G-CU1M
    Xcvr 1       REV 01   740-038623   MOC15306230781    QSFP+-40G-CU1M
  PIC 2          REV 04   611-044925   MY3713290241      4x 1G/10G SFP/SFP+
    Xcvr 0       REV 01   740-021309   AWJ0621           SFP+-10G-LR
    Xcvr 1       REV 01   740-021309   AWA0BQM           SFP+-10G-LR
    Xcvr 2       REV 01   740-021308   MSN0MMM           SFP+-10G-SR
    Xcvr 3       REV 01   740-021308   MSN0M3N           SFP+-10G-SR
```

```
  Power Supply 0 REV 01   740-046873   1EDE3270441       JPSU-350-AC-AFO-A
  Fan Tray 0                                            Fan Module, Airflow Out (AFO)
  Fan Tray 1                                            Fan Module, Airflow Out (AFO)
FPC 1            REV 06   650-044936   PG3713290002      EX4300-24T
  CPU                     BUILTIN      BUILTIN           FPC CPU
  PIC 0          REV 06   BUILTIN      BUILTIN           24x 10/100/1000 Base-T
  PIC 1          REV 06   BUILTIN      BUILTIN           4x 40GE QSFP+
    Xcvr 0       REV 01   740-038623   MOC15306230995    QSFP+-40G-CU1M
    Xcvr 1       REV 01   740-038623   MOC15306230781    QSFP+-40G-CU1M
  PIC 2          REV 04   611-044925   MY3713290179      4x 1G/10G SFP/SFP+
    Xcvr 1       REV 01   740-021309   AWJ06MJ           SFP+-10G-LR
  Power Supply 0 REV 01   740-046873   1EDE3270365       JPSU-350-AC-AFO-A
  Fan Tray 0                                            Fan Module, Airflow Out (AFO)
  Fan Tray 1                                            Fan Module, Airflow O
```

Looking at the output you can see the routing engines are identified right away. Then, each FPC. The FPC is synonymous with an entire switch which has one of two roles (routing engine or line card). It's important that you know how your Virtual Chassis (VC) is cabled and which switch is where. If FPC 0 is on the middle of your stack, then you are going to have problems when trying to look at the VC and plug in cables.

BEST PRACTICE Best practice is to preprovision your VC and use a top-down approach. Start with FPC0 and go to FPC9 from the top to the bottom. Otherwise it could be confusing to anyone troubleshooting or plugging into the VC who doesn't have access to the `show chassis hardware` output. It may even be a good idea to print out the show chassis hardware and have it in the cabinet.

So far you can see that FPC 0 is an EX4300-24T, and PIC 0 on FPC 0 is our 24x tri-rate copper interfaces. Next is PIC 1 which is the built-in 4 x 40G QSFP+ ports that are automatically provisioned as Virtual Chassis Ports (VCP). Finally, there is the 4 x 1G/10G SFP/SFP+ uplink module.

From this point, you can surmise that our first available physical port in the VC will be ge-0/0/0. This is read as INT-TYPE–FPC/PIC/PORT where our FPC = 0, the PIC = 0, and the PORT = 0. The ports are labeled 0-23 (24 ports).

Provisioning and connecting the uplink ports is where most folks have trouble. This is because some of the EX Series switches have the capability of adding modules like the EX-UM-4x4SFP module. When this is inserted it actually becomes PIC 2 as seen in the hardware output from above. Therefore, if you wanted to connect to ge-1/2/0 you would plug your cable into the first port of the 4xSFP+ module on FPC 1 (the second switch in the VC).

NOTE Introducing the EX4300-32F into a VC (see Figure 4.5) with other EX4300 models can be confusing as well because the 4xSFP+ uplinks are actually part of PIC 0 and are labeled 32-35. And then, when you add the EX-UM-8x8SFP

module, it becomes PIC 2. For further information on interface numbering please see: https://www.juniper.net/documentation/en_US/junos/topics/concept/interfaces-naming-conventions.html.

It is important that you understand all network numbering in Juniper starts with 0 and not 1. If you are ever having a problem figuring out the FPC/PIC/PORT, just look at the output of show chassis hardware and follow the logic. The serial numbers can be used to locate a specific member if necessary.

Virtual Chassis Priorities and Roles

There are only two roles that you can assign to a member in a Virtual Chassis: `Backup` and `Master`.

```
EX4300-6_7-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 3cc8.b1b8.7370
Virtual Chassis Mode: Enabled
                                          Mstr          Mixed Route Neighbor List
Member ID  Status   Serial No    Model         prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    PG37290062 ex4300-24t     129   Backup       N  VC   1  vcp-255/1/0
                                                                      1  vcp-255/1/1
1 (FPC 1)  Prsnt    PG37290002 ex4300-24t     129   Master*      N  VC   0  vcp-255/1/0
                                                                      0  vcp-255/1/1
```

The simple two-member VC configured here shows that both members are present and that the Member 1 switch is currently the master routing engine. The configuration is `preprovisioned`:

```
EX4300-6_7-VC> show configuration | display set | match virtual
set groups POC_Lab virtual-chassis preprovisioned
set groups POC_Lab virtual-chassis no-split-detection
set groups POC_Lab virtual-chassis member 0 role routing-engine
set groups POC_Lab virtual-chassis member 0 serial-number PG37290062
set groups POC_Lab virtual-chassis member 1 role routing-engine
set groups POC_Lab virtual-chassis member 1 serial-number PG37290002
```

By using the predefined roles of `routing-engine` and `line-card` you can lock in the switches that are supposed to act as REs and LCs, so if there is ever an issue, they will not try to take on roles they are not assigned. In retrospect, if you wanted all switches to take over you could always use priorities.

By not using the `preprovisioned` statement in `virtual-chassis` configuration, you enable `priorities`.

The highest priority on a VC member wins. Therefore, you want the master and backup RE to have the same priority: 255. The line cards can then be set to 128:

```
QFX5100-2_3-VC# show virtual-chassis | display set
set virtual-chassis no-split-detection
set virtual-chassis member 0 mastership-priority 255
set virtual-chassis member 1 mastership-priority 255
```

```
set virtual-chassis aliases serial-number TA3716170025 alias-name RE1
set virtual-chassis aliases serial-number TA3716170306 alias-name RE0
```

Note that the keyword `preprovisioned` is not used in assigning priorities.

Virtual Chassis Alias Name

This is a very short primer on using `alias-name` for virtual chassis. Here you set the different members' names, RE0 and RE1, but you could just as easily make them DC-RACK1-ROW5 or BLDG5-RM309. This would come in handy in an extended VC where a single switch is spread across ten different locations.

```
QFX5100-2_3-VC# set virtual-chassis aliases serial-number TA3716170306 alias-name RE0
QFX5100-2_3-VC# set virtual-chassis aliases serial-number TA3716170025 alias-name RE1
QFX5100-2_3-VC# run show virtual-chassis
Virtual Chassis ID: 1783.5be0.70d8
Virtual Chassis Mode: Enabled
                                      Mstr          Mixed Route Neighbor List
Member ID  Status   Serial No    Alias-Name   Model         prio  Role     Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    TA3716170306 RE0          qfx5100-48s-6q 128   Master*     N  VC   1  vcp-255/0/48
                                                                                1  vcp-255/0/49
1 (FPC 1)  Prsnt    TA3716170025 RE1          qfx5100-48s-6q 128   Backup      N  VC   0  vcp-255/0/48
                                                                                0  vcp-255/0/49
```

And that's it pretty simple but could really help in locating specific members of an extended VC.

Virtual Chassis Convert VCP to 40G Uplink

Another value add of Juniper EX Series switches is that most of the models are 40G backbone ready. The number of QSFP+ ports vary with each model, so make sure that you check out the hardware specifications for each. For example, the EX4300-24T has four QSFP+ ports while the EX4300-32F only has two built-in QSFP+ ports but has the capability to expand to four using the EX-UM-2QSFP module.

If you are thinking about purchasing a 10G capable switch for your backbone but need to future proof for 40G in the next 2-5 years, Juniper has you covered. The QSFP+ ports which are configured as Virtual Chassis Ports (VCPs) by default can be converted to 40G ports.

NOTE You have to be careful with your planning as you will most likely use at least two QSFP+ ports for creating the VC backplane. Make sure that you have enough switches in the stack to support the 40G optics you need for the network. Remember you can always add additional switches to the VC, up to 10, in most cases.

Let's do a walkthrough of changing the VCP port to a normal 40G port on an EX4300.

First, you need to know what ports are connected on the QSFP+ ports, so let's take a look:

```
EX4300-6_7-VC> show virtual-chassis vc-port
fpc0:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
PIC / Port
1/0         Configured          5    Up           40000        1   vcp-255/1/0
1/1         Configured          5    Up           40000        1   vcp-255/1/1
1/3         Configured               Absent
1/2         Configured               Absent
fpc1:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
PIC / Port
1/0         Configured          5    Up           40000        0   vcp-255/1/0
1/1         Configured          5    Up           40000        0   vcp-255/1/1
1/3         Configured               Absent
1/2         Configured               Absent
```

As you can see, we have PIC 1 PORTS 0 and one in use on each of the switches that make up the Virtual Chassis. Let's focus on member 0 PIC 1 and PORT 2 to convert to a normal interface:

```
EX4300-6_7-VC> request virtual-chassis vc-port delete pic-slot 1 port 2 member 0
fpc0:
--------------------------------------------------------------------------
vc-port successfully deleted

{master:1}
EX4300-6_7-VC> show virtual-chassis vc-port
fpc0:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
PIC / Port
1/0         Configured          5    Up           40000        1   vcp-255/1/0
1/1         Configured          5    Up           40000        1   vcp-255/1/1
1/3         Configured               Absent
fpc1:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
PIC / Port
1/0         Configured          5    Up           40000        0   vcp-255/1/0
1/1         Configured          5    Up           40000        0   vcp-255/1/1
1/3         Configured               Absent
1/2         Configured               Absent
```

Now you can see that the PIC 1 PORT 2 interface is no longer present on the switch. The next step is to configure the 40G interface on the switch:

```
EX4300-6_7-VC# set interfaces et-0/1/2 unit 0 family inet address 192.168.99.1/31

{master:1}[edit]
```

```
EX4300-6_7-VC# commit
fpc1:
configuration check succeeds
fpc0:
commit complete
fpc1:
commit complete
```

NOTE Until an optic is actually physically in the port, the interface will not show up. Once the QSFP+ optic is inserted it will show in the output of `> show chassis hardware` and `> show interfaces terse`:

```
EX4300-6_7-VC> show interfaces terse | match et-
et-0/1/2              up    up
et-0/1/2.0            up    up   inet     192.168.99.1/31
```

Now you can see the 40G interface is up, so let's check the optic in the hardware output:

```
EX4300-6_7-VC> show chassis hardware
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                PG3713290002      Virtual Chassis
Routing Engine 0 REV 06   650-044936   PG3713290062      EX4300-24T
Routing Engine 1 REV 06   650-044936   PG3713290002      EX4300-24T
FPC 0            REV 06   650-044936   PG3713290062      EX4300-24T
  CPU                     BUILTIN      BUILTIN           FPC CPU
  PIC 0          REV 06   BUILTIN      BUILTIN           24x 10/100/1000 Base-T
  PIC 1          REV 06   BUILTIN      BUILTIN           4x 40GE QSFP+
    Xcvr 0       REV 01   740-038623   MOC15306230995    QSFP+-40G-CU1M
    Xcvr 1       REV 01   740-038623   MOC15306230781    QSFP+-40G-CU1M
    Xcvr 2       REV 01   740-032986   QA520152          QSFP+-40G-SR4
```

And then you configure the IP on the EX4600 side on FPC 0 PIC 0 PORT 26:

```
EX4600-4_5-VC# run show chassis hardware
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                TC3714350038      Virtual Chassis
Routing Engine 0          BUILTIN      BUILTIN           EX4600-40F
Routing Engine 1          BUILTIN      BUILTIN           EX4600-40F
FPC 0            REV 22   650-049940   TC3714350038      EX4600-40F
  CPU                     BUILTIN      BUILTIN           FPC CPU
  PIC 0                   BUILTIN      BUILTIN           24x10G-4x40G
    Xcvr 0       REV 01   740-021308   MWD1BEU           SFP+-10G-SR
    Xcvr 2       REV 02   740-013111   C072289           SFP-T
    Xcvr 4       REV 01   740-021309   AS93DD5           SFP+-10G-LR
    Xcvr 24      REV 01   740-038623   MOC15526231437    QSFP+-40G-CU1M
    Xcvr 25      REV 01   740-038623   MOC15526231999    QSFP+-40G-CU1M
    Xcvr 26      REV 01   740-046565   QE434942          QSFP+-40G-SR4

EX4600-4_5-VC# set interfaces et-0/0/26.0 family inet address 192.168.99.0/31
```

Once that is committed you can do a quick ping to see if there's connectivity:

```
EX4600-4_5-VC# run ping 192.168.99.1 source 192.168.99.0
PING 192.168.99.1 (192.168.99.1): 56 data bytes
```

```
64 bytes from 192.168.99.1: icmp_seq=0 ttl=64 time=2.865 ms
64 bytes from 192.168.99.1: icmp_seq=1 ttl=64 time=11.114 ms
64 bytes from 192.168.99.1: icmp_seq=2 ttl=64 time=11.119 ms
^C
--- 192.168.99.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.865/8.366/11.119/3.890 ms
```

TIP You should have noticed the IP address used to connect the 40G interfaces together is a /31 or true point-to-point address. If you ever need to reserve address space or just want a bunch of point-to-point addresses, then using a /31 CIDR scheme is the way to go.

As you can see in the highlighted output a QSFP+-40G-SR4 was the lab.

NOTE The EX3400/4300/4600/9200 add a lot of value to the network because they can scale from 1G/10G/40G just by changing an optic. No forklift upgrades required.

So far in this chapter you learned how to take a preconfigured Virtual Chassis Port (VCP) on a QSFP+ interface and change it to a normal network interface that is capable of 40G throughput. Not only can you use the 40G optic for normal traffic, but you can also use that same optic to extend your Virtual Chassis to another room, data center, building, campus, city, or even state! That is a huge discriminator from other less capable vendors who can only extend to three meters!

Virtual Chassis Mixed-Mode and Extended

In addition to the 40G optic inserted above, let's also connect another QSFP+ DAC cable from EX4300 member 1 to the EX4600 member 1 so that you can create, you guessed it, a mixed-mode VC using EX4600s and EX4300s.

You might remember from a previous note that the EX4600s, when placed in a mixed-mode virtual chassis, have to be the routing engines. So what you need to do is grab all the serial numbers from the switches, make sure you have a ring topology for the VC, and then you can `preprovision` the mixed-mode VC and see if you can turn the two VCs into one managed VC. This is the fun stuff Systems Engineers get to do!

```
EX4600-4_5-VC# run show chassis hardware | match FPC | except BUILTIN
FPC 0            REV 22   650-049940   TC3714350038      EX4600-40F
FPC 1            REV 22   650-049940   TC3714350071      EX4600-40F

EX4300-6_7-VC> show chassis hardware | match FPC | except BUILT
FPC 0            REV 06   650-044936   PG3713290062      EX4300-24T
FPC 1            REV 06   650-044936   PG3713290002      EX4300-24T
```

Great. There are the serial numbers of the proposed mixed-mode VC. The next thing needed is to create the preprovisioned configuration and commit it. First,

let's take a look at the actual cabling of the VC. Figure 4.6 depicts the cabling structure that is in place to create the virtual chassis. The EX4300-24T is showing the back side of the switch to enforce the fact that QSFP+ is on the back of EX4300s and on the front of EX4600s.

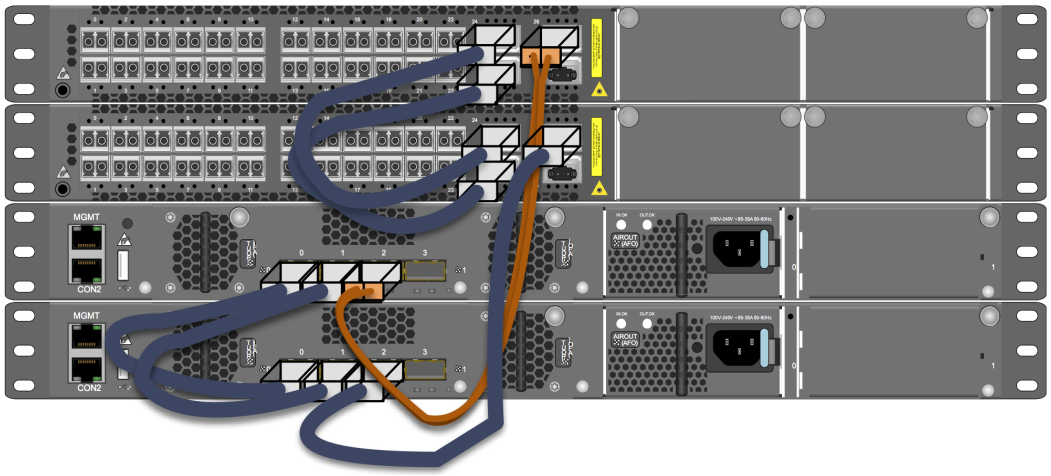


Figure 4.6 *Mixed Mode and Extended Virtual Chassis Example*

There is now a 160G Link Aggregation using QSFP+ between like switches. This is automatic when the Virtual Chassis sees multiple connections to neighboring switches over VCP ports. There is a single 40G SR optic connected between member 0 and member 2 as well as another single QSFP+ DAC cable between member 1 and member 3. If you trace out the ring, you go from member 0 to member 1 to member 3 to member 2 to member 0, completing a full redundant ring topology.

NOTE If you are OCD this might bother you and you would want to swap the DAC and the 40G SR optic on members 2 and 3, unfortunately the setup is remote and my lab guy is busy doing other stuff, but you should be aware of this detail.

The next step is to tell member 0 of the EX4600-VC that PIC 0 PORT 26 will now be a VCP port instead of a network interface. The same goes for member 0 of the EX4300-VC:

```
EX4600-4_5-VC> request virtual-chassis vc-port set pic-slot 0 port 26
Port conversion initiated,  use show virtual-chassis vc-port to verify

EX4300-6_7-VC> show virtual-chassis vc-port
fpc0:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
```

```
PIC / Port
1/0         Configured          5    Up           40000        1   vcp-255/1/0
1/1         Configured          5    Up           40000        1   vcp-255/1/1
1/3         Configured               Absent
fpc1:
--------------------------------------------------------------------------
Interface   Type              Trunk  Status       Speed        Neighbor
or                             ID                (mbps)       ID  Interface
PIC / Port
1/0         Configured          5    Up           40000        0   vcp-255/1/0
1/1         Configured          5    Up           40000        0   vcp-255/1/1
1/2         Configured         -1    Down         40000
1/3         Configured               Absent

EX4300-6_7-VC> request virtual-chassis vc-port set pic-slot member 0 1 port 2
fpc0:
--------------------------------------------------------------------------
Port conversion initiated,  use show virtual-chassis vc-port to verify
```

As soon as you convert the QSFP+ ports back on the EX4300s you can see that the EX4600 has already found the EX4300s and thinks they are part of the VC:

```
EX4600-4_5-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                                 Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    TC3714350038 ex4600-40f     129   Master*      N  VC   2  vcp-255/0/26
1 (FPC 1)  NotPrsnt TC3714350071 ex4600-40f
2 (FPC 2)  Inactive PG3713290062 ex4300-24t       0   Linecard     N  VC   3  vcp-255/1/0
                                                                          3  vcp-255/1/1
                                                                          0  vcp-255/1/2
3 (FPC 3)  Inactive PG3713290002 ex4300-24t       0   Linecard     N  VC   2  vcp-255/1/0
                                                                          2  vcp-255/1/1
```

Now, apply the following virtual-chassis configuration:

```
EX4600-4_5-VC# show virtual-chassis | display set
set virtual-chassis auto-sw-update
set virtual-chassis preprovisioned
set virtual-chassis member 0 role routing-engine
set virtual-chassis member 0 serial-number TC3714350038
set virtual-chassis member 1 role routing-engine
set virtual-chassis member 1 serial-number TC3714350071
set virtual-chassis member 2 role line-card
set virtual-chassis member 2 serial-number PG3713290062
set virtual-chassis member 3 role line-card
set virtual-chassis member 3 serial-number PG3713290002
```

Finally, the secret sauce gets applied to the `virtual-chassis` to make it mixed mode. Once this is run, the VC needs to reboot, so go ahead and issue the `reboot` command along with the `request`. Let's make it happen, captain!

```
EX4600-4_5-VC> request virtual-chassis mode mixed all-members reboot
fpc1:
--------------------------------------------------------------------------
```

```
Mode set to 'Virtual Chassis with mixed devices'.  Rebooting system...
fpc2:
--------------------------------------------------------------------------
Mode set to 'Virtual Chassis with mixed devices'.  Rebooting system...
fpc3:
--------------------------------------------------------------------------
Mode set to 'Virtual Chassis with mixed devices'.  Rebooting system...
fpc0:
--------------------------------------------------------------------------
Mode set to 'Virtual Chassis with mixed devices'.  Rebooting system...

*** System shutdown message from root@EX4600-4_5-VC ***
System going down in 1 minute
```

Once the VC is rebooted, log back in and you should see that there is now a mixed-mode Virtual Chassis with EX4600s and EX4300s.

```
EX4600-4_5-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Mixed
                                          Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model     prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    TC3714350038 ex4600-40f  129   Master*     Y  VC   1  vcp-255/0/24
                                                                    2  vcp-255/0/26
1 (FPC 1)  Prsnt    TC3714350071 ex4600-40f  129   Backup      Y  VC   0  vcp-255/0/24
2 (FPC 2)  Prsnt    PG3713290062 ex4300-24t    0   Linecard    Y  VC   0  vcp-255/1/2
                                                                    3  vcp-255/1/0
                                                                    3  vcp-255/1/1
3 (FPC 3)  Prsnt    PG3713290002 ex4300-24t    0   Linecard    Y  VC   2  vcp-255/1/1
                                                                    2  vcp-255/1/0
```

NOTE The `Y` under `Mode` indicates that Mixed Mode is enabled. An `N` indicates that it is not enabled.

To break the mixed-mode VC into two separate VCs again, you need to remove members 2 and 3 from the `virtual-chassis` configuration and add the `no-split-detection` statement back into the virtual chassis configuration. You also need to convert this back to standard `NON-MIXED` mode or `virtual-chassis` with similar devices – to do so simply run the following commands and then perform a reboot:

```
EX4600-4_5-VC> request virtual-chassis mode disable mixed all-members reboot
fpc1:
--------------------------------------------------------------------------
Mode set to 'Virtual Chassis with similar devices'.  Rebooting system...
```

Once everything has been configured and the mixed mode is disabled and the system is rebooted you can see that you are indeed back to a two-switch EX4600 VC and a two-switch EX4300 VC:

```
EX4600-4_5-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                          Mstr           Mixed Route Neighbor List
```

```
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    TC3714350038 ex4600-40f     128   Backup       N  VC   1  vcp-255/0/24
1 (FPC 1)  Prsnt    TC3714350071 ex4600-40f     129   Master*      N  VC   0  vcp-255/0/24

EX4300-6_7-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                              Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    PG3713290062 ex4300-24t     129   Master*      N  VC   1  vcp-255/1/0
                                                                            1  vcp-255/1/1
1 (FPC 1)  Prsnt    PG3713290002 ex4300-24t     129   Backup       N  VC   0  vcp-255/1/0
                                                                            0  vcp-255/1/1
```

Virtual Chassis Change Mastership

Sometimes you may want to manually switch the mastership from one routing engine to another. To do this there are three commands available from the Junos operational prompt:

```
EX4300-6_7-VC> request chassis routing-engine master ?
Possible completions:
  acquire              Attempt to become master Routing Engine
  release              Request that other Routing Engine become master
  switch               Toggle mastership between Routing Engines

EX4300-6_7-VC> request chassis routing-engine master release
warning: Traffic will be interrupted while the PFE is re-initialized
Request the other routing engine become master ? [yes,no] (no) yes
```

This results in member 1 now being the master routing engine.

NOTE It should be noted that without NSB, NSR GRES, and NSSU, the forwarding plane will be interrupted on master switchover.

```
EX4300-6_7-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                              Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    PG3713290062 ex4300-24t     129   Backup       N  VC   1  vcp-255/1/0
                                                                            1  vcp-255/1/1
1 (FPC 1)  Prsnt    PG3713290002 ex4300-24t     129   Master*      N  VC   0  vcp-255/1/0
                                                                            0  vcp-255/1/1
```

You can also use the `switch` keyword:

```
EX4300-6_7-VC> request chassis routing-engine master switch no-confirm
EX4300-6_7-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
```

```
                                                Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    PG3713290062 ex4300-24t     129   Master*      N  VC   1  vcp-255/1/0
                                                                          1  vcp-255/1/1
1 (FPC 1)  Prsnt    PG3713290002 ex4300-24t     129   Backup       N  VC   0  vcp-255/1/0
                                                                          0  vcp-255/1/1
```

And finally, there is the `acquire` keyword, but you have to be on the device that you want to acquire mastership, so first you need to `request session member 1`:

```
EX4300-6_7-VC> request session member 1
--- Junos 14.1X53-D40.8 built 2016-11-09 04:54:19 UTC
warning: This chassis is operating in a non-master role as part of a virtual-chassis (VC) system.
warning: Use of interactive commands should be limited to debugging and VC Port operations.
warning: Full CLI access is provided by the Virtual Chassis Master (VC-M) chassis.
warning: The VC-M can be identified through the show virtual-
chassis status command executed at this console.
warning: Please logout and log into the VC-M to use CLI.

{backup:1}
lab@EX4300-6_7-VC> request chassis routing-engine master acquire
warning: Traffic will be interrupted while the PFE is re-initialized
Attempt to become the master routing engine ? [yes,no] (no) yes
Command aborted. Not ready for mastership switch, try after 70 secs.
```

NOTE There is a minimum four minute (240 seconds) wait period between successive routing engine mastership switchover. This is to prevent any flapping situations which could be detrimental to a network.

```
EX4300-6_7-VC> request chassis routing-engine master acquire
warning: Traffic will be interrupted while the PFE is re-initialized
Attempt to become the master routing engine ? [yes,no] (no) yes

EX4300-6_7-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                                Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    PG3713290062 ex4300-24t     129   Backup       N  VC   1  vcp-255/1/0
                                                                          1  vcp-255/1/1
1 (FPC 1)  Prsnt    PG3713290002 ex4300-24t     129   Master*      N  VC   0  vcp-255/1/0
                                                                          0  vcp-255/1/1
```

NOTE There may be times where something happens and you have to replace a switch in a VC. Or you may have a situation where you added an additional switch to a VC, and then removed a different one, and now your member IDs are not in order. You can recycle member ID's using the `recycle` command with `request virtual-chassis recycle`. For more information on recycling member IDs, please see: https://www.juniper.net/documentation/en_US/junos/topics/reference/command-summary/request-virtual-chassis-recycle.html.

You can check for mastership changes in the mastership log file:

```
EX4300-6_7-VC> show log mastership | match "Jul 28" | match Role
Jul 28 00:26:54 Role: LINECARD
Jul 28 00:41:56 Role: LINECARD
Jul 28 01:12:01 Role: BACKUP
Jul 28 01:15:13 Role: LINECARD
Jul 28 01:36:34 Role: BACKUP
Jul 28 01:44:33 Role: BACKUP
Jul 28 01:59:00 Role: BACKUP
```

Summary

In summary, the lab is 40G backbone ready by converting QSFP+ VCP ports to normal network ports. You have been shown that you can extend a single VC over an area of ~ 4894 sq. miles using ten switches in a ring topology using 40G Extended Range optics. You can also create LAG simply by adding another VCP to your neighboring member. In addition, you took two VCs made up of different models of switches and combined them into a single VC that has one point of management. Oh, and that one VC which has 248 miles of fiber – it can be managed as a single switch and the same VLAN can appear in every location that the VCs line cards and REs are in.

The Juniper Networks Virtual Chassis (VC) solution is *not* the same as stacking. You aren't limited to three meters and you can connect up to ten switches (with the exception of the EX2200 and EX2300s) and if you wanted to go further you have Virtual Chassis Fabric (VCF). For more information on Junipers Virtual Chassis (VC) solution visit the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/virtual-chassis-ex4300-configuring.html.

Chapter 5

Configuring Virtual Chassis High Availability Features

What is Non-Stop Service Upgrade (NSSU)

Having configured a virtual chassis you now have even more capabilities and high availability features available to you.

Non-Stop Service Upgrade (NSSU) provides the ability to upgrade all members of the virtual chassis including all Line Cards (LCs) and Routing-Engines (REs) with a single command. During the upgrade, there should only be minimal network impact during routing engine switchover.

TIP Because NSSU upgrades the switches in a VC, one at a time, you should experience minimal impact. If you take the time to design your uplinks so they are not all connected to one switch in the VC, and utilize LAGs that cross multiple members, that will provide even further protection against forwarding plane interruption during upgrades.

There are some pre-requisites to enable NSSU, so let's list out the requirements up front:

- Ring Topology to prevent any isolation
- Master and Backup RE should be adjacent to each other
- Pre-Provisioned Virtual Chassis
- Graceful Routing Engine Switchover (GRES)
- Non-Stop Routing (NSR) and Non-Stop Bridging (NSB)

NSSU works by verifying that there is a backup RE that is at the same software level as the master. That GRES and NSR are both enabled and operational and that the Virtual Chassis is preprovisioned.

NOTE The reason the VC has to be preprovisioned is that the switches have to maintain their identity throughout the reboots.

When issuing the NSSU software upgrade, the first step is for the Master RE to load the new software on the Backup RE, reboot the backup, and once the software is loaded, perform a resync.

After the Backup is resynced the master loads the new software on each of the line cards and reboots them, one at a time, waiting until each line card is complete and active before moving on to the next line card.

Once all of the line cards are upgraded, and on line, the master performs a Graceful Routing Engine Switchover (GRES) and then performs the software upgrade on itself and automatically reboots. Once the original RE comes back on line and is seen as the Backup RE, you have the option of changing it back to the master if you desire.

So, you can see that by performing an upgrade on each individual switch you only impact the access users on that switch for that reboot period. If you have uplinks on those individual line cards without diverse paths connected or Link Aggregation Groups (LAG), your impact may be more substantial. By terminating uplinks on separate members of the virtual chassis you'll insure that you aren't taking all of your uplinks down at one time.

There are a couple of things you cannot do with NSSU:

1. You cannot downgrade to an earlier version using NSSU.

2. You cannot `rollback` after issuing an NSSU.

Let's configure the prerequisites and then see it in action.

```
EX4300-6_7-VC# set chassis nssu
```

Configuring Graceful Routing Engine Switchover (GRES)

One of the first prerequisites for NSSU is GRES. GRES allows the Packet Forwarding Engines (PFEs) to resynchronize their state instead of reinitializing their state. The resynchronization minimalizes any interruptions to traffic.

You'll see that when configuring GRES the kernel is synchronized across both master and backup REs. This allows the routing table and forwarding table to maintain consistency, and when a switchover does occur, those tables are already populated and in sync. This eliminates the convergence time to relearn the network.

```
EX4300-6_7-VC# set chassis redundancy graceful-switchover
```

At this point the `task replication` is still showing `disabled` so let's continue on and enable NSR and NSB.

Configuring Non-Stop Routing and Non-Stop Bridging

Non-Stop Active Routing (NSR) allows the routing table to synchronize across both Routing Engines. This allows packets to be forwarded based on that synchronized state without having to learn all routes and wait for convergence of dynamic routing protocols.

```
EX4300-6_7-VC# set routing-options nonstop-routing
```

ELS ALERT Non-stop bridging (NSB) on ELS switches has moved from the `ethernet-switching-options` (old non-ELS) hierarchy to the `protocols layer2-control` hierarchy.

```
EX4300-6_7-VC# set protocols layer2-control nonstop-bridging
```

Now, after you commit you should be able to see that task replication is enabled:

```
EX4300-6_7-VC# run show task replication
       Stateful Replication: Enabled
       RE mode: Master
```

Performing NSSU

Everything is in place for a software upgrade, so let's go ahead and upgrade the EX4300 VC from 14.1X53-D30.3 to 14.1X53-D35.3 and watch. First, you need to stage the code, and for this exercise let's use a small stepped release jinstall-ex-4300-14.1X53-D35.3:

```
$  scp jinstall-ex-4300-14.1X53-D35.3-domestic-signed.tgz 172.16.0.6:/var/tmp/
$ telnet 172.16.0.15 2006
Trying 172.16.0.15...
Connected to 172.16.0.15.
Escape character is '^]'.
EX4300-6_7-VC (ttyu0)
login: lab
Password:
--- Junos 14.1X53-D30.3 built 2017-04-23 01:56:25 UTC
{master:0}
lab@EX4300-6_7-VC> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                        Mstr          Mixed Route Neighbor List
Member ID  Status   Serial No    Model         prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    PG3713290062 ex4300-24t    129   Master*     N  VC   1  vcp-255/1/0
           1  vcp-255/1/1
1 (FPC 1)  Prsnt    PG3713290002 ex4300-24t    129   Backup      N  VC   0  vcp-255/1/0
0  vcp-255/1/1
```

The code is now staged and ready to perform the NSSU. All of our prerequisites are met.

```
EX4300-6_7-VC# show | display set | match "grace|non"
set chassis redundancy graceful-switchover
```

```
set routing-options nonstop-routing
set protocols layer2-control nonstop-bridging
EX4300-6_7-VC> show task replication
       Stateful Replication: Enabled
       RE mode: Master
EX4300-6_7-VC> set cli screen-width 160
Screen width set to 160
```

Now start the Non-Stop Service Upgrade (NSSU). The jinstall package will be copied to member 1 first and then once it is rebooted member 0 will release mastership and then load the software and reboot itself:

```
lab@EX4300-6_7-VC> request system software nonstop-upgrade /var/tmp/jinstall-ex-4300-14.1X53-D35.3-
domestic-signed.tgz
Chassis ISSU Check Done
[Jul 30 13:05:24]:ISSU: Validating Image
[Jul 30 13:05:25]:ISSU: Preparing Backup RE
Installing image on other FPC's along with the backup
[Jul 30 13:05:25]: Checking pending install on fpc1
Pushing /var/tmp/jinstall-ex-4300-14.1X53-D35.3-domestic-signed.tgz to fpc1:/var/tmp/jinstall-ex-
4300-14.1X53-D35.3-domestic-signed.tgz
NOTICE: Validating configuration against jinstall-ex-4300-14.1X53-D35.3-domestic-signed.tgz.
```

NOTE Use the `no-validate` option to skip this if desired.

Verify the signature of the new package:

```
Verified jinstall-ex-4300-14.1X53-D35.3-domestic.tgz signed by PackageProductionRSA_2016
WARNING: A reboot is required to install the software
WARNING:     Use the 'request system reboot' command immediately
[Jul 30 13:06:23]: Completed install on fpc1
[Jul 30 13:06:38]: Backup upgrade done
[Jul 30 13:06:38]: Rebooting Backup RE
Rebooting fpc1
[Jul 30 13:06:39]:ISSU: Backup RE Prepare Done
[Jul 30 13:06:39]: Waiting for Backup RE reboot
```

The upgrade on the backup RE completed at 13:14 so it took 8 minutes for the code to be copied, verified, and then upgraded and rebooted. Then member 1 becomes master and member 0 is immediately rebooted. We can now take a look at the *virtual-chassis status* to see who is master and backup:

```
lab@EX4300-6_7-VC> show virtual-chassis status
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                        Mstr          Mixed Route Neighbor List
Member ID  Status   Serial No    Model         prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  NotPrsnt PG3713290062 ex4300-24t
1 (FPC 1)  Prsnt    PG3713290002 ex4300-24t    129   Master*     N  VC
```

You can see that member 0 is `NotPrsnt` because it is currently being upgraded and rebooted. It takes a little less time for member 0 to reboot because there is no copy function required. The entire upgrade process for the 2-member VC is approximately 15 minutes but we never lost positive control of the VC because

each member was rebooted and fully operational before the next member was upgraded. We also had full GRES, NSR, and NSB, so the routing and forwarding table was synchronized before the NSSU was started.

Once member 0 is back from the software upgrade and rebooted, you can see that it is now the backup RE for the VC:

```
EX4300-6_7-VC> show virtual-chassis status
Preprovisioned Virtual Chassis
Virtual Chassis ID: 72ad.ea24.787d
Virtual Chassis Mode: Enabled
                                        Mstr          Mixed Route Neighbor List
Member ID  Status   Serial No    Model         prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    PG3713290062 ex4300-24t    129   Backup      N  VC   1  vcp-255/1/0
                                                                     1  vcp-255/1/1
1 (FPC 1)  Prsnt    PG3713290002 ex4300-24t    129   Master*     N  VC   0  vcp-255/1/0
                                                                     0  vcp-255/1/1
```

If desired, mastership can be switched to member 0. Let's do that so you can see the process.

NOTE There is some debate on maintaining member 0 as the master at all times. Some engineers see this as an immediate sign that something has happened if they log into a device and it is on member 1 as master. Others see this as normal operation and would just look at the mastership logs or get an SNMP TRAP. What do you think?

```
EX4300-6_7-VC> request chassis routing-engine master switch
Toggle mastership between routing engines ? [yes,no] (no) yes
{master:1}
lab@EX4300-6_7-VC>
EX4300-6_7-VC (ttyu0)
login: lab
Password:
--- Junos 14.1X53-D35.3 built 2016-03-01 02:32:19 UTC
{master:0}
lab@EX4300-6_7-VC>
```

TIP Whenever you are planning on performing an NSSU please make sure that you verify the Junos OS version and features to make sure that NSSU is supported and also check any release notes or check with your service manager for known issues on your specific hardware and Junos OS combination. Juniper Networks provides a Feature Explorer web app and the NSSU permlink is here: https://pathfinder.juniper.net/feature-explorer/feature-info.html?fKey=1175&fn=Nonstop+software+upgrade+(NSSU).

NSSU requires some prerequisites before upgrading. GRES, NSR, NSB are all required to set up task replication and synchronize routing and forwarding tables across the backup and master REs. The benefit of NSSU is that it does a waterfall type upgrade where it performs the upgrade on the backup RE and then each

individual line card, one at a time, and then finally relinquishes mastership and upgrades the final RE. This type of upgrade minimizes downtime.

Configuring Auto-SW Upgrade

The last thing to do is not a requirement for NSSU but it allows any new members to automatically download the software and upgrade when connected as a member to the virtual chassis.

This allows any new switches being added to upgrade or downgrade to the proper Junos OS version and be seen as an active member of the VC.

BEST PRACTICE If you are configuring tons of Virtual Chassis and aren't using something like ZTP to upgrade the code then you might want to seriously consider adding this to the preprovisioned statement to allow auto upgrade. It's a really big time saver!!

```
EX4300-6_7-VC# set virtual-chassis auto-sw-update ex-4300 package-name /var/tmp/jinstall-ex-4300-
14.1X53-D35.3-signed.tgz
```

Once the statement is committed and you add another member to the virtual chassis, that new member will automatically upgrade using the package name that you identified in the `auto-sw-update` command.

Summary

This chapter discussed how Non-Stop Service Upgrade has the capability to minimalize impact during software upgrades and bring the VC into the correct version of code. NSSU uses a waterfall like process where it upgrades one member at a time until they are all upgraded and then it relinquishes mastership and upgrades itself as the last step.

There are several prerequisites discussed such as GRES, NSR, and NSB and you want to add `auto-sw-upgrade` and `commit-synchronize` as well. Once everything is in place you can see that task replication is enabled and active across both of the REs and that convergence time for routing protocols is minimalized and our forwarding plane is synchronized as well.

NSSU cannot guarantee 0 impact but it can significantly minimize any impact using resynchronization versus reinitialization. Other vendors cannot say the same!

MORE? For more information on Virtual Chassis High Availability features, visit the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/concept/ex-series-high-availability.html.

Chapter 6

Configuring EX Series Interfaces

Configuring Layer 2 Interface Modes

The network ports on EX switches are configured with the protocol family `ethernet-switching` by factory default. This means the ports are automatically enabled for Layer 2 bridging and switching. The interfaces on EX switches are also set as access mode by default, although it is not explicit in the configuration. The procedure to configure the ports as access or trunk ports is given below.

ELS ALERT! The commands to create access and trunk ports is different on ELS enabled switches.

```
EX4300-6_7-VC# set interfaces ge-0/0/0.0 family ethernet-switching interface-mode trunk vlan members
all
```

TIP Although the EX interfaces are set to access ports by way of the factory default configuration, it's best practice to explicitly set the interface mode to access so it is clearly visible when reviewing the configuration.

```
EX4300-6_7-VC# set interfaces ge-0/0/0.0 family ethernet-switching interface-mode access vlan members
V100
```

It was mentioned in Chapter 1, but deserves a second mention, the `wildcard range` command is a simple way to configure many interfaces in one command. For example:

```
EX4300-6_7-VC# wildcard range set interfaces ge-[0-1]/0/[0-23] unit 0 family ethernet-switching
interface-mode access vlan members V100
-
EX4300-6_7-VC# show | compare
[edit interfaces ge-0/0/0 unit 0 family ethernet-switching]
```

```
-      interface-mode trunk;
+      interface-mode access;
[edit interfaces ge-0/0/0 unit 0 family ethernet-switching vlan]
-       members all;
+       members [ all V100 ];
[edit interfaces ge-0/0/1 unit 0 family ethernet-switching]
+      interface-mode access;
+      vlan {
+          members V100;
+      }
```

As you can see, the new ELS command `interface-mode` has replaced the older `port-mode` command in ELS.

Configuring VLAN Membership On an Interface

In ELS, you configure a port to be a member of one or more VLANs under the interface, a difference compared to non-ELS switches. On non-ELS switches, you had the option of configuring VLAN membership under the interface or under the `vlan` stanza.

Adding a specific VLAN to an access interface was shown previously but let's go ahead and do it again just for reinforcement:

```
EX4300-6_7-VC# set interfaces ge-0/0/1.0 family ethernet-switching interface-mode access vlan members
V100
```

NOTE You can only add a single VLAN member to an access port. The only exception to this is when you are running a shared voice or data port with LLDP-MED, which will be discussed later.

You can add multiple VLANs to a trunk port. There are also different ways to add multiple VLANs to a trunk port. You can list each VLAN name or ID individually. You could simply use the `all` keyword indicating that the port would be a member of every VLAN configured on the switch. Or you could set a range of VLANs. An example of each method is detailed below. Let's add V100, V200, and V300 to the trunk ports for this example. First, you can add them one at a time:

```
EX4300-6_7-VC# set interfaces xe-0/2/2.0 family ethernet-switching interface-mode trunk vlan members
V100
EX4300-6_7-VC# set interfaces xe-0/2/2.0 family ethernet-switching interface-mode trunk vlan members
V200
EX4300-6_7-VC# set interfaces xe-0/2/2.0 family ethernet-switching interface-mode trunk vlan members
V300
EX4300-6_7-VC# show interfaces xe-0/2/2
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ V100 V200 V300 ];
        }
    }
}
```

Or, you can add them as a set of values:

```
EX4300-6_7-VC# set interfaces xe-0/2/2.0 family ethernet-switching interface-mode trunk vlan members
[V100 V200 V300]
EX4300-6_7-VC# show interfaces xe-0/2/2
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members [ V100 V200 V300 ];
        }
    }
}
```

Finally, you can also use the `all` keyword and place all of the available VLANs on a specific trunked interface:

```
EX4300-6_7-VC# set interfaces xe-0/2/2.0 family ethernet-switching interface-mode trunk vlan members
all
EX4300-6_7-VC# show interfaces xe-0/2/2
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members all;
        }
    }
}
```

Configuring INET (layer 3 IPv4) interfaces

By default, all the ports on an ELS EX switch are in Layer 2 switching mode running the `ethernet-switching` protocol family. Any or all of those ports can be configured as Layer 3 ports by configuring the protocol family 'inet or inet6' and setting the address on the interface:

Remember that `family ethernet switching` is configured on every interface by default. Interfaces can't be configured with multiple protocol families. Therefore, you need to delete the other family either before or after changing it. Deleting the unit seems to be the easiest method:

```
EX4300-6_7-VC# show interfaces ge-0/0/4
unit 0 {
    family ethernet-switching {
        interface-mode access;
        vlan {
            members V100;
        }
    }
}
EX4300-6_7-VC# delete interfaces ge-0/0/4.0
EX4300-6_7-VC# set interfaces ge-0/0/4.0 family inet address 10.10.1.1/30
```

If you did not delete the `family ethernet-switching` prior to commit you would get a fail warning and your commit would not work:

```
EX4300-6_7-VC # commit
[edit interfaces ge-0/0/4 unit 0 family]
  'ethernet-switching'
    Family ethernet-switching and rest of the families are mutually exclusive
error: commit failed: (statements constraint check failed)
```

MORE? See this link for a complete overview on EX Series Switch interfaces: https://www.juniper.net/documentation/en_US/junos/topics/concept/interfaces-ex-series-overview.html.

Configuring INET6 (Layer 3 IPv6) Addresses

IPv6 addresses are configured using the INET6 protocol family. Other than the different protocol family, the process is the same as configuring INET/IPv4 addresses:

```
EX4300-6_7-VC# delete interfaces ge-0/0/4.0
EX4300-6_7-VC# set interfaces ge-0/0/4.0 family inet6 address 2001:470:1f11:51f::3:2

EX4300-6_7-VC# run show interfaces ge-0/0/4.0 detail
Logical interface ge-0/0/4.0 (Index 578) (SNMP ifIndex 604) (HW Token 4092) (Generation 221)
    Flags: Device-Down SNMP-Traps 0x0 Encapsulation: ENET2
    Traffic statistics:
…
    Protocol inet6, MTU: 1500, Generation: 239, Route table: 0
      Flags: Is-Primary
      Addresses, Flags: Is-Default Is-Primary
        Destination: Unspecified, Local: 2001:470:1f11:51f::3:2
        INET6 Address Flags: Tentative
    Generation: 148
      Addresses, Flags: Is-Preferred
        Destination: fe80::/64, Local: fe80::4e96:14ff:fee4:6607
        INET6 Address Flags: Tentative
    Generation: 150
```

When you configure an IPv6 address using the inet6 family you also have a link local address with FE80. This address will not show up in the routing table. It's there for things like neighbor discovery. For more information on IPv6 please check out *Day One: Exploring IPv6*: https://www.juniper.net/us/en/training/jn-books/day-one/networking-technologies-series/exploring-ipv6/.

Configuring IRB Interfaces

Integrated Routing and Bridging (IRB) is a new arrival via ELS. IRB interfaces provide support for Layer 2 bridging or Layer 3 routing on the same interface. The IRB interface is a virtual interface and is commonly used to route packets between VLANs. The IRB interface replaces the Routed VLAN Interface (RVI) in non-ELS switches. The old syntax was *vlan.100* or *vlan.10*, while the new ELS syntax is *irb.100* or *irb.10*.

NOTE IRB interfaces serve as gateways to VLANs and they will *not* support the `ethernet-switching` family. The supported families on IRB are `inet`, `inet6`, `iso`, and `mpls`.

```
EX4300-6_7-VC# set interfaces irb.200 family inet address 10.10.4.2/31
```

Unless the IRB interface is associated to a VLAN and there is a supported interface, it will show `down`. To associate the IRB to a VLAN use the `l3-interface` keyword (note that it's *L3* not *13*).

```
EX4300-6_7-VC# set interfaces irb.200 family inet address 10.10.4.2/31
EX4300-6_7-VC# set vlans V200 l3-interface irb.200
EX4300-6_7-VC# commit
EX4300-6_7-VC# show vlans V200 | display set
set vlans V200 vlan-id 200
set vlans V200 l3-interface irb.200

EX4300-6_7-VC# run show interfaces irb.200 terse
Interface               Admin Link Proto    Local                 Remote
irb.200                 up    down inet     10.10.4.2/31

EX4300-6_7-VC# set interfaces xe-0/2/1.0 family ethernet-switching interface-mode trunk vlan members
V200
EX4300-6_7-VC# commit
EX4300-6_7-VC# run show interfaces irb.200 terse
Interface               Admin Link Proto    Local                 Remote
irb.200                 up    up   inet     10.10.4.2/31
```

As you can see, initially we configured the IRB interface and associated it to V200 through the `l3-interface` command, but the interface showed `down` until the VLAN was associated with an operational interface.

Configuring Aggregated Ethernet Interfaces

EX switches support IEEE 802.3ad link aggregation allowing users to bundle multiple Ethernet interfaces together to form a link aggregation group (LAG). Traffic is load-balanced across the multiple physical Ethernet interfaces in the LAG. An Aggregated Ethernet (ae) interface is used to configure Layer 2 functions like VLAN membership and LACP.

MORE? For complete information on IEEE 802.3ad Link Aggregation Support visit: https://www.juniper.net/documentation/en_US/junos/topics/concept/interfaces-lag-overview.html.

TIP Before starting to configure Link Aggregation Groups (LAG), you should make sure that you set the number of aggregate ports on the chassis. This step is often missed and it can waste a lot of time just for want of a simple configuration statement.

The first step in configuring aggregate interfaces is to tell the chassis how many ae's you are going to configure, or are planning. This command is under the `chassis` hierarchy:

```
EX4300-6_7-VC# set chassis aggregated-devices ethernet device-count 2
```

You don't have to add Link Aggregation Control Protocol (LACP) but it's a great idea to do so because LACP provides automatic link addition and removal as well as monitoring both ends of the aggregate interface to ensure connectivity. For more information on configuring LACP please see: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/lacp-cli.html.

You also do not have to configure link protection but we wanted to introduce this command in case you haven't seen it before. Link protection is an LACP feature that allows you to have a two-link LAG but only one link is active. The other link is in standby status in case the primary link fails. For more information on link protection please see: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/lacp-link-protection-cli.html.

So far, you have only enabled the chassis for LAG but you haven't configured the Aggregate Ethernet (ae) interfaces. You actually create the LAG by identifying the `802.1ae` keyword on the individual interfaces and then assigning it to the ae interface that you want to build. Since we only enabled a device count of two, there is ae0 and ae1 to work with:

```
EX4300-6_7-VC# set interfaces ge-0/0/5 ether-options 802.3ad ae0
EX4300-6_7-VC# set interfaces ge-0/0/6 ether-options 802.3ad ae0
EX4300-6_7-VC# commit
EX4300-6_7-VC# show interfaces ge-0/0/5
ether-options {
    802.3ad ae0;
}
EX4300-6_7-VC# show interfaces ge-0/0/6
ether-options {
    802.3ad ae0;
}
EX4300-6_7-VC# set interfaces ae0.0 family inet address 10.20.30.1/30
EX4300-6_7-VC# run show interfaces terse | match ae
ge-0/0/5.0              up    down aenet    --> ae0.0
ge-0/0/6.0              up    down aenet    --> ae0.0
ae0                     up    down
ae0.0                   up    down inet     10.20.30.1/30
ae1                     up    down
```

As you can see, `ether-options` was used under the physical interface to apply 802.1ae along with the ae interface desired. An inet family address to the unit on ae0 was added. You can have numerous units on a single aggregate interface and they can be tagged or IP. For more information on LAG interfaces visit the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/link-aggregation-cli.html.

Configuring VRRP

Virtual Routing Redundancy Protocol (VRRP) allows multiple switches or routers on a LAN to provide a default gateway to multiple hosts. VRRP share a virtual IP address (VIP) corresponding to the default route on the hosts. One of the VRRP switches functions as the master switch while others serve as backups. If the master fails, one of the backup switches becomes the new master and assumes default routing responsibilities for hosts on the LAN. This capability provides resiliency to network failures.

Let's configure a EX2300_8_9 VC to be a Layer 2 access device and the QFX5100-VC and EX4600-VCs will be the VRRP gateways on the aggregation layer.

The first thing you need is to identify your VLAN default gateway on the access layer. For this exercise, let's use VLAN V200 for two access links and 10.200.0.0/24 for our network. The QFX5100 VC will have a physical address of .2 and the EX4600 VC will have a .3 physical, while the Virtual IP (VIP) will be .1. Let's configure that part now:

```
EX2300-8_9-VC# show vlans
V200 {
    vlan-id 200;
    l3-interface irb.200;
}

EX2300-8_9-VC# show interfaces irb.200
family inet {
    address 10.200.0.254/24;
}

QFX5100-2_3-VC# show vlans V200
vlan-id 200;
l3-interface irb.200;

QFX5100-2_3-VC# show interfaces irb.200
family inet {
    address 10.200.0.2/24;
}
QFX5100-2_3-VC# show interfaces ge-0/0/4
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members V200;
        }
    }
}

QFX5100-2_3-VC# run ping 10.200.0.254 count 5 rapid
PING 10.200.0.254 (10.200.0.254): 56 data bytes
!!!!!
--- 10.200.0.254 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 10.156/15.771/32.989/8.705 ms
```

```
EX4600-4_5-VC# show interfaces ge-1/0/4
unit 0 {
    family ethernet-switching {
        interface-mode trunk;
        vlan {
            members V200;
        }
    }
}

EX4600-4_5-VC# show vlans V200
vlan-id 200;
l3-interface irb.200;

EX4600-4_5-VC# show interfaces irb.200
family inet {
    address 10.200.0.3/24;
}

EX4600-4_5-VC# run ping 10.200.0.254 count 5 rapid
PING 10.200.0.254 (10.200.0.254): 56 data bytes
!!!!!
--- 10.200.0.254 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 10.881/11.654/14.427/1.387 ms
```

So far, all of the interfaces on VLAN V200 and provided IRB family inet addresses in the 10.200.0.0/24 network have been configured. However, there still isn't a single gateway address. The way you manage this is to create VRRP interfaces for each of the Aggregation layer switches on irb.200:

```
EX4600-4_5-VC# set interfaces irb.200 family inet address 10.200.0.3/24 vrrp-group 200 virtual-address
10.200.0.1 authentication-type md5

EX4600-4_5-VC# set interfaces irb.200 family inet address 10.200.0.3/24 vrrp-group 200 virtual-address
10.200.0.1 authentication-key Juniper123

EX4600-4_5-VC# set interfaces irb.200 family inet address 10.200.0.3/24 vrrp-group 200 virtual-address
10.200.0.1 priority 200

QFX5100-2_3-VC# set interfaces irb.200 family inet address 10.200.0.2/24 vrrp-group 200 virtual-
address 10.200.0.1 authentication-type md5 authentication-key Juniper123

QFX5100-2_3-VC# set interfaces irb.200 family inet address 10.200.0.2/24 vrrp-group 200 priority 100
preempt
```

A common issue for engineers when dealing with VRRP for the first time is they do all of the configuration and then they jump on the access layer switch and ping the VIP of the VRRP group. The problem is it doesn't reply. This is per RFC-5798 (pg. 24 of 650). To allow the master VRRP device to accept and respond to ICMP destined for the VIP address, you have to add the `accept-data` command in Junos:

```
QFX5100-2_3-VC# set interfaces irb.200 family inet address 10.200.0.2/24 vrrp-group 200 accept-data

EX4600-4_5-VC# set interfaces irb.200 family inet address 10.200.0.3/24 vrrp-group 200 accept-data
```

Once committed, it's back to our access device to see if we can ping the VIP:

```
EX2300-8_9-VC> ping 10.200.0.1
PING 10.200.0.1 (10.200.0.1): 56 data bytes
64 bytes from 10.200.0.1: icmp_seq=0 ttl=64 time=11.999 ms
64 bytes from 10.200.0.1: icmp_seq=1 ttl=64 time=9.708 ms
^C
--- 10.200.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 9.708/10.854/11.999/1.145 ms
```

This is great. There is now a VIP or default gateway you can point to but which device is it, and how do you test failover for this? You can influence the device that you want to be master of the VIP for VRRP group 200. Let's assume you want the QFX5100 VC to always be the master if it is online and available:

```
QFX5100-2_3-VC# run show vrrp
Interface     State       Group   VR state VR Mode   Timer    Type   Address
irb.200       up            200   backup   Active     D  3.349 lcl    10.200.0.2
                                                              vip    10.200.0.1
                                                              mas    10.200.0.3
QFX5100-2_3-VC# set interfaces irb unit 200 family inet address 10.200.0.2/24 vrrp-group 200 priority
254
QFX5100-2_3-VC# run show vrrp
Interface     State       Group   VR state VR Mode   Timer    Type   Address
irb.200       up            200   master   Active     A  0.722 lcl    10.200.0.2
                                                              vip    10.200.0.1
```

As you can see, initially the QFX5100 was the backup for the VRRP 200 group, due to the priority set earlier being lower than the priority on the EX4600. Once the simple priority change was made you can see that the QFX5100 is the master for the VRRP group 200. Let's look at the group on EX4600 now:

```
EX4600-4_5-VC# run show vrrp
Interface     State       Group   VR state VR Mode   Timer    Type   Address
irb.200       up            200   backup   Active     D  2.884 lcl    10.200.0.3
                                                              vip    10.200.0.1
                                                              mas    10.200.0.2
```

Using the `detail` keyword on `show VRRP` provides a lot of feedback. As you can see from the following output, QFX5100 is the active master and that we are using authentication between the two devices:

```
QFX5100-2_3-VC# run show vrrp detail
Physical interface: irb, Unit: 200, Address: 10.200.0.2/24
  Index: 550, SNMP ifIndex: 538, VRRP-Traps: enabled, VRRP-Version: 2
  Interface state: up, Group: 200, State: master, VRRP Mode: Active
  Priority: 254, Advertisement interval: 1, Authentication type: md5
  Advertisement threshold: 3, Computed send rate: 0
  Preempt: yes, Accept-data mode: yes, VIP count: 1, VIP: 10.200.0.1
  Advertisement Timer: 0.343s, Master router: 10.200.0.2
  Virtual router uptime: 00:18:03, Master router uptime: 00:04:42
  Virtual Mac: 00:00:5e:00:01:c8
  Tracking: disabled
```

Now, let's fail our interface ge-0/0/4 on the QFX5100 VC and see if we can still ping our gateway. Let's start a ping from the access to see how many packets are lost during the failover:

```
EX2300-8_9-VC> ping 10.200.0.1
PING 10.200.0.1 (10.200.0.1): 56 data bytes
64 bytes from 10.200.0.1: icmp_seq=0 ttl=64 time=10.326 ms
64 bytes from 10.200.0.1: icmp_seq=1 ttl=64 time=10.210 ms
64 bytes from 10.200.0.1: icmp_seq=2 ttl=64 time=14.352 ms
64 bytes from 10.200.0.1: icmp_seq=3 ttl=64 time=13.971 ms
64 bytes from 10.200.0.1: icmp_seq=4 ttl=64 time=16.522 ms
64 bytes from 10.200.0.1: icmp_seq=5 ttl=64 time=10.377 ms
64 bytes from 10.200.0.1: icmp_seq=6 ttl=64 time=12.052 ms
64 bytes from 10.200.0.1: icmp_seq=20 ttl=64 time=41.638 ms
64 bytes from 10.200.0.1: icmp_seq=21 ttl=64 time=11.830 ms
```

This experimen lost about 14 pings during the VRRP mastership switchover. The good thing is that our ping session just didn't die. It was able to recover just like most user traffic would in this scenario. Without changing the default `fast-interval` settings the delay can be between 5-10 seconds.

Let's see if the failover time can improve by configuring `fast-interval`:

```
EX4600-4_5-VC# set interfaces irb.200 family inet address 10.200.0.3/24 vrrp-group 200 fast-interval
100

QFX5100-2_3-VC# set interfaces irb.200 family inet address 10.200.0.2/24 vrrp-group 200 fast-interval
100
```

Now bring up the master ge-0/0/4 interface and allow the QFX5100 to switch to master:

```
QFX5100-2_3-VC# run show vrrp
Interface     State      Group   VR state VR Mode   Timer    Type   Address
irb.200       up           200   master   Active      A  0.004 lcl    10.200.0.2
                                                              vip    10.200.0.1
QFX5100-2_3-VC# run show vrrp detail
Physical interface: irb, Unit: 200, Address: 10.200.0.2/24
  Index: 550, SNMP ifIndex: 538, VRRP-Traps: enabled, VRRP-Version: 2
  Interface state: up, Group: 200, State: master, VRRP Mode: Active
  Priority: 254, Advertisement interval: .100, Authentication type: md5
  Advertisement threshold: 3, Computed send rate: 10
  Preempt: yes, Accept-data mode: yes, VIP count: 1, VIP: 10.200.0.1
  Advertisement Timer: 0.010s, Master router: 10.200.0.2
  Virtual router uptime: 00:02:40, Master router uptime: 00:02:36
  Virtual Mac: 00:00:5e:00:01:c8
  Tracking: disabled
```

And now re-run the same failover test done previously to see if there is any improvement:

```
EX2300-8_9-VC> ping 10.200.0.1
PING 10.200.0.1 (10.200.0.1): 56 data bytes
64 bytes from 10.200.0.1: icmp_seq=0 ttl=64 time=10.686 ms
64 bytes from 10.200.0.1: icmp_seq=1 ttl=64 time=5.831 ms
64 bytes from 10.200.0.1: icmp_seq=2 ttl=64 time=66.205 ms
64 bytes from 10.200.0.1: icmp_seq=3 ttl=64 time=10.295 ms
```

```
64 bytes from 10.200.0.1: icmp_seq=4 ttl=64 time=15.327 ms
64 bytes from 10.200.0.1: icmp_seq=5 ttl=64 time=31.729 ms
64 bytes from 10.200.0.1: icmp_seq=6 ttl=64 time=15.994 ms
64 bytes from 10.200.0.1: icmp_seq=7 ttl=64 time=13.904 ms
64 bytes from 10.200.0.1: icmp_seq=14 ttl=64 time=43.363 ms
```

Not a whole lot of improvement but it got down to seven seconds. There are several timers that you can tweak to improve the VRRP performance, but we can't cover them all so please refer to the TechLibrary for more detailed information on VRRP: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/vrrp-basic-support-configuring.html.

Configuring Energy Efficient Ethernet Interfaces

Energy Efficient Ethernet (EEE) is an IEEE standard (802.3az) that reduces power consumption on switch ports during periods of low utilization. When a link is idle the EEE capable switch can put the port in low power mode to save energy:

```
EX2300-8_9-VC# set interfaces ge-0/0/4 ether-options ieee-802-3az-eee

EX2300-8_9-VC# run show interfaces ge-0/0/4
Physical interface: ge-0/0/4, Enabled, Physical link is Down
  Interface index: 652, SNMP ifIndex: 517
  Link-level type: Ethernet, MTU: 1514, LAN-PHY mode, Link-mode: Half-duplex, Speed: Auto, Duplex:
Auto, BPDU Error: None, Loop Detect PDU Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
Source filtering: Disabled, Flow control: Enabled, Auto-negotiation: Enabled, Remote fault: Online,
Media type: Copper, IEEE 802.3az Energy Efficient Ethernet: Enabled, NO LPI
… Truncated for brevity
```

EEE is disabled by default. When links are idle, EEE places those interfaces into low power mode. This could translate to huge cost-savings in large POE+ environments and data centers, and is a value-added feature for Juniper EX Series devices.

Configuring Power Over Ethernet (POE) Interfaces

PoE on Juniper EX Series switches is straightforward. For simplicity purposes, it is a plug-and-play type of function for devices capable of using the 802.3af/at standard for power, as well as supplying network connectivity over Cat5e or above. Typically, PoE is used to power VOIP phones, cameras, and APs (Access Points).

There are two modes of POE. One being *active* and other referred to as *passive*. Active PoE is any device that conforms to either the 802.3af or 802.3at standard. Essentially this is any device that is capable of doing a PoE handshake with the switch and negotiating its power requirement. If this does not happen, then the device will not power up.

The majority of Juniper switches have full support across all ports for POE, and with two power supplies, most Juniper P models will support PoE+ on all ports. An example is the EX4200-24T which has eight ports that are PoE-capable.

If any device model has a P in it, then all ports are PoE capable. Just remember, the PoE supported standard varies from switch to switch so you will need to identify the capabilities for that specific model. Here is a good starting point: https://www.juniper.net/documentation/en_US/junos/topics/concept/poe-overview.html.

Juniper switches support one of the following depending on model:

- *PoE*: PoE is limited to 15.4 watts
- *PoE+*: PoE Enhanced is an extension to PoE for a total of 18.6 watts per port
- *Enhanced PoE*: PoE+ is 30 watts per port

By default, the Junos OS has all interfaces configured out of the box for PoE:

```
EX2300-C# show poe
interface all;
```

Setting Up PoE on a Juniper Switch for a Device.

It's best practice to apply descriptions to ports and interfaces so you know which one is which. In this next example, there are several devices that are PoE-capable and are, in fact, using PoE to function. Let's use these as a reference for setting up PoE because you need an interface that a PoE device is connected to.

BEST PRACTICE While the default setting for PoE is `interface all`, it's best practice to get more granular and specify `poe` for only those ports that need power versus all ports.

You can remove all PoE by just deleting the `poe` stanza and then applying `poe` to specific interfaces as required:

```
EX2300-C# show poe
interface all;

EX2300-C# delete poe

EX2300-C# set poe interface ge-0/0/9

EX2300-C# set poe interface ge-0/0/2

EX2300-C# set poe interface ge-0/0/4

EX2300-C# set poe interface ge-0/0/6

EX2300-C# set poe interface ge-0/0/10

EX2300-C# set poe interface ge-0/0/11

EX2300-C# commit
configuration check succeeds
```

You can now verify the PoE interfaces that are enabled, maximum wattage per interface priority, and their actual power consumption, like this:

```
EX2300-C> show poe interface
Interface    Admin       Oper    Max        Priority       Power          Class
             status      status  power                     consumption
 ge-0/0/0    Disabled    Disabled 0.0W      Low            0.0W           not-applicable
 ge-0/0/1    Disabled    Disabled 0.0W      Low            0.0W           not-applicable
 ge-0/0/2    Enabled      ON     15.4W      Low            2.6W            3
 ge-0/0/3    Disabled    Disabled 0.0W      Low            0.0W           not-applicable
 ge-0/0/4    Enabled      ON     15.4W      Low            2.4W            3
 ge-0/0/5    Disabled    Disabled 0.0W      Low            0.0W           not-applicable
 ge-0/0/6    Enabled      ON     15.4W      Low            2.8W            3
 ge-0/0/7    Disabled    Disabled 0.0W      Low            0.0W           not-applicable
 ge-0/0/8    Disabled    Disabled 0.0W      Low            0.0W           not-applicable
 ge-0/0/9    Enabled      ON     15.4W      Low            4.0W            0
ge-0/0/10    Enabled      ON     15.4W      Low            3.6W            0
ge-0/0/11    Enabled      ON     15.4W      Low            4.2W            0
```

You can also check to see on the total amount of power being used across all PoE ports by issuing the `show poe controller` command:

```
EX2300-C> show poe controller
Controller  Maximum   Power         Guard    Management   Status        Lldp
index       power     consumption   band                               Priority
   0        125W      20.60W          0W     Class        AT_MODE       Disabled
```

Here you can see the maximum power available and the actual total power consumption that is being used by our PoE interfaces on the switch.

Summary

That is as far as we're going to explore PoE configuration and its show commands. All of Juniper Networks Power over Ethernet (PoE) switches have `poe` enabled across all interfaces by default. For more information on configuring and monitoring `poe` please visit: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/poe-cli.html.

Chapter 7

Configuring EX Series Datacenter Technology

Configuring Multi-Chassis Link Aggregation

Multi-Chassis LAG has been around for quite some time and it's used quite a bit in data center environments. The EX4300, EX4600, EX9200, and QFX Series support Multi-Chassis Link Aggregation Group or MC-LAG. You can see the other devices that support MC-LAG via the Juniper Feature Explorer here: https://pathfinder.juniper.net/feature-explorer/feature-info.html?fKey=4070&fn=Multichassis+link+aggregation+(MC-LAG).

Unfortunately, this book's lab is not setup to support MC-LAG or JFE. For more information on configuring MC-LAG please refer to: https://www.juniper.net/documentation/en_US/junos/topics/concept/mc-lag-feature-summary-best-practices.html.

Junos Fusion Enterprise

Traditional enterprise switch architectures can be very difficult and time consuming to install and maintain. The networks have multiple layer switches interconnected together in various architectures. As networks have grown to support new services, so too has the complexity. The physical cabling alone is enough to drive administrators crazy, not to mention the potentially hundreds or even thousands of management points. This makes configuration errors hard to avoid. Troubleshooting becomes next to impossible.

But it doesn't end there! Throw in multiple layers of security and it's an administrator's nightmare. A seemingly simple change can break something resulting in service outages and holes in security.

So, what's the solution? How can a legacy, three-tiered switch network be simpler to deploy and manage? How can the network to be flexible enough to absorb hardware adds and new services without massive disruption?

The answer is to adopt a simplified architecture that's flexible, scalable, and easier to manage. This is an evolution that's been taking place in the data center for a while now. Data centers of all sizes have been transitioning to various fabric architectures that simplify management with architectural simplicity and automation, while simultaneously accommodating massive scale. Junos Fusion Enterprise (JFE) leverages data center fabric technology and brings the same concepts to the enterprise. At a high level, JFE collapses the legacy three tiers of switch layers into a single, logical device. In a JFE fabric, the access switches become line cards in an aggregation switch. All configurations and operations tasks are performed on the aggregation switch, called the *aggregation device*. The access switches in a JFE fabric become *satellite devices* and are no longer managed individually. The Satellite Devices only provide user or server network connectivity. Figure 7.1 illustrates JFE.

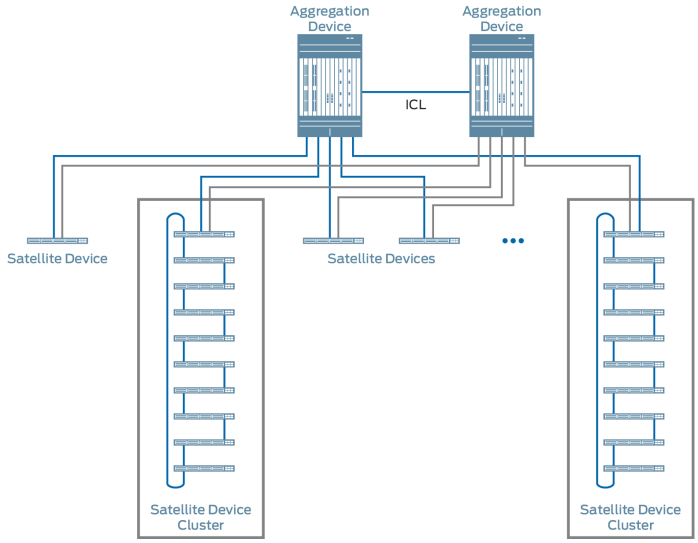


Figure 7.1 *Junos Fusion Enterprise (JFE) Sample Topology.*

In a network like this it's obvious how much simpler it would be to consolidate the management down to two devices. And with features like configuration synchronization, Fusion commands can be entered on one AD and automatically synched to the other AD. Now, new VLANs, filters, CoS configurations, and policies can be added once instead of dozens of times. Also, troubleshooting becomes much simpler. Tracking down the MAC address of a client device does not require

administrators logging in to multiple switches. It is all accomplished centrally. To that end, the Satellite Devices will forward syslog and SNMP trap information to the Aggregation Devices as well.

Let's step through the hardware components of JFE:

- Aggregation Device
  - This is the brains of a JFE fabric. All management of the JFE fabric happens at the AD.
  - Currently the EX9200 switches can act as the AD in a JFE fabric.
  - Runs the Junos OS.
  - JFE supports one or two Ads.
  - The ports that connect to the Satellite Devices are called *Cascade Ports*
- Satellite Device
  - Provides the network interfaces in the JFE fabric.
  - Runs Satellite Network OS (SNOS).
  - Currently the EX2300, EX3400, and EX4300 can act as Satellite Devices in a JFE fabric.
  - JFE supports 128 SDs.
  - Ports that connect to the Aggregation Devices are called *Uplink Ports*.
  - Ports that connect to client devices are called *Extended Ports*

Figure 7.2 illustrates the port naming in JFE:

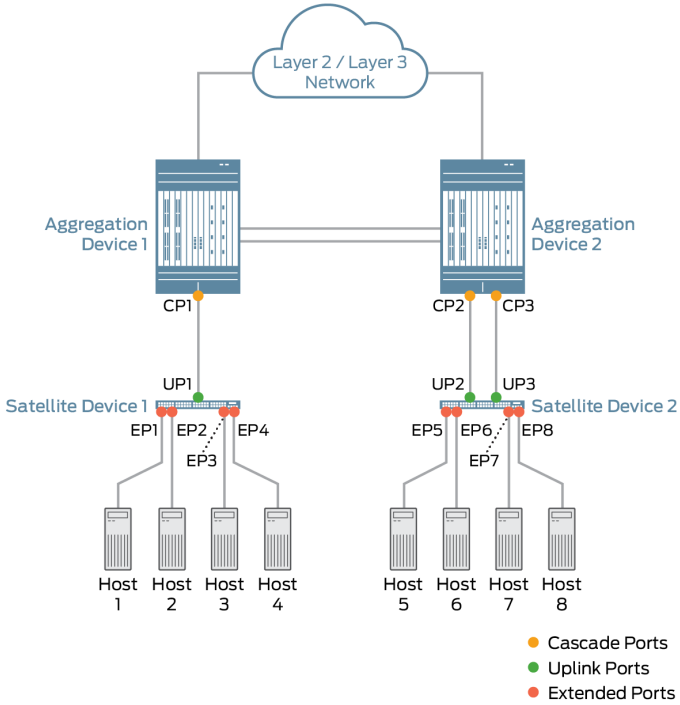


Figure 7.2 *Junos Fusion Enterprise (JFE) Port Naming Example*

On the software side, JFE utilizes standard protocols to manage traffic flows, as shown in this short summary of the protocols used in a JFE fabric:

1. LLDP: Used for device discovery and auto provisioning
2. 802.1BR+: Satellite Device management for all traffic related aspects. Ex: configurations, statistics, port states.
3. json-rpc: Satellite Device management for non-traffic related aspects. Ex: chassis, environmentals, upgrades.
4. Netconf: Configuration Synchronization between Aggregation Devices in a dual-AD configuration.
5. ICCP: Inter Chassis Control Protocol. LAG synchronization between Aggregation Devices (MC-LAG).
6. IS-IS: Satellite Device cluster topology discovery and control.

MORE? For information on how traffic flows through a JFE fabric see: https://www.juniper.net/documentation/en_US/junos/topics/concept/fusion-data-packets-flow.html.

One of the great advantages of JFE is its flexibility in how you can deploy satellite devices. The SDs can be configured in a cluster as shown in the Figure 7.3. An individual SD can be multi-homed or single-homed. And, you can also connect devices to the ADs that aren't part of the JFE fabric.

In summary, when a JFE fabric is formed, an Enterprise switch network comprised of 128 switches and ~6,000 ports are managed as a single device. So, your campus/enterprise switch network shrinks from 130 management points to two (when you have a dual-AD configuration). Below are the basic configuration steps for deploying a JFE fabric that includes Satellite Devices that are connected in a cluster. More features such as Configuration Synchronization, Configuration Consistency Checks, and Satellite Device Autoconversion are detailed in the link below.

MORE? For more information on Junos Fusion see: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/junos-fusion/junos-fusion-enterprise.html.

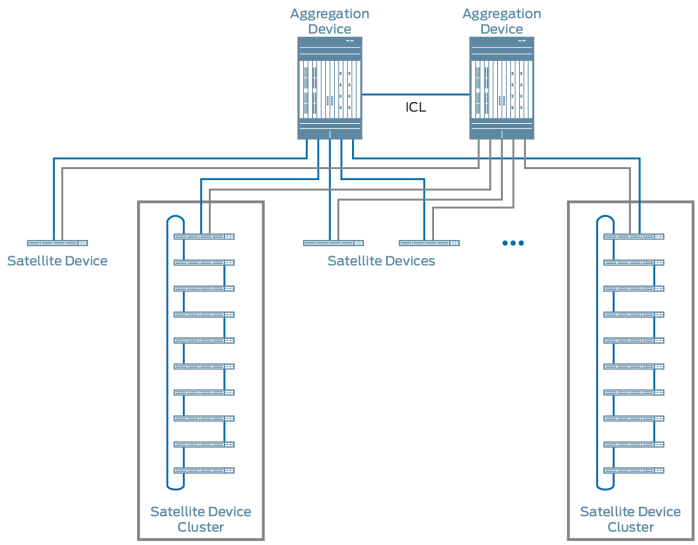


Figure 7.3 *Junos Fusion Enterprise (JFE) Topology.*

Configuration

EX9204-AD1

ICCP Link - Explicit configuration. In JFE you can explicitly configure ICCP or have it automatically created using `auto-iccp`. First is the explicit configuration option followed by an `auto-iccp` option:

```
set interfaces xe-2/0/4 description "ICCP Layer 3"
set interfaces xe-2/0/4 vlan-tagging
set interfaces xe-2/0/4 unit 0 vlan-id 400
set interfaces xe-2/0/4 unit 0 family inet address 192.168.10.1/30
ICL Link:
set interfaces xe-2/0/5 description "ICL Layer 2 link"
set interfaces xe-2/0/5 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/5 unit 0 family ethernet-switching vlan members all
```

Loopback and routing:

```
set interfaces lo0 unit 0 family inet address 192.168.10.11/32
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface xe-2/0/4.0
```

Explicit ICCP configuration:

```
set protocols iccp local-ip-addr 192.168.10.11
set protocols iccp peer 192.168.10.12 session-establishment-hold-time 50
set protocols iccp peer 192.168.10.12 redundancy-group-id-list 2
set protocols iccp peer 192.168.10.12 backup-liveness-detection backup-peer-ip 10.19.11.44
set protocols iccp peer 192.168.10.12 liveness-detection minimum-interval 2000
set protocols iccp peer 192.168.10.12 liveness-detection multiplier 4
```

EX9208-AD2

ICCP link:

```
set interfaces xe-3/0/3 description "ICCP Layer3 link"
set interfaces xe-3/0/3 vlan-tagging
set interfaces xe-3/0/3 unit 0 vlan-id 4000
set interfaces xe-3/0/3 unit 0 family inet address 192.168.10.2/30
```

ICL link:

```
set interfaces xe-3/0/4 description "ICL Layer 2 Link"
set interfaces xe-3/0/4 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-3/0/4 unit 0 family ethernet-switching vlan members all
```

Loopback and routing:

```
set interfaces lo0.0 family inet address 192.168.10.12/32
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ospf area 0.0.0.0 interface xe-3/0/3.0
```

Explicit ICCP configuration:

```
set protocols iccp local-ip-addr 192.168.10.12
set protocols iccp peer 192.168.10.11 session-establishment-hold-time 50
```

```
set protocols iccp peer 192.168.10.11 redundancy-group-id-list 2
set protocols iccp peer 192.168.10.11 backup-liveness-detection backup-peer-ip 10.19.11.36
set protocols iccp peer 192.168.10.11 liveness-detection minimum-interval 2000
set protocols iccp peer 192.168.10.11 liveness-detection multiplier 4
```

Verify ICCP is up between the two EX9200s:

```
EX9204-01-RE0# run show iccp
Redundancy Group Information for peer 192.168.10.12
  TCP Connection       : Established
  Liveliness Detection : Up
Backup liveness peer status: Up
  Redundancy Group ID          Status
    2                           Up
Client Application: mclag_cfgchkd
  Redundancy Group IDs Joined: 0 2
Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None



EX9208-01-RE0# run show iccp
Redundancy Group Information for peer 192.168.10.11
  TCP Connection       : Established
  Liveliness Detection : Up
Backup liveness peer status: Up
  Redundancy Group ID          Status
    2                           Up
Client Application: mclag_cfgchkd
  Redundancy Group IDs Joined: 0 2
Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None
```

Configure Fusion on the ADs

AD1:

```
set chassis satellite-management redundancy-groups chassis-id 1
set chassis satellite-management redundancy-groups rg2 redundancy-group-id 2
set chassis satellite-management redundancy-groups rg2 peer-chassis-id 2 inter-chassis-link xe-2/0/5
```

AD2:

```
set chassis satellite-management redundancy-groups chassis-id 2
set chassis satellite-management redundancy-groups rg2 redundancy-group-id 2
set chassis satellite-management redundancy-groups rg2 peer-chassis-id 1 inter-chassis-link xe-3/0/4
```

Verify Fusion is enabled on the ADs:

```
EX9204-01-RE0# run show chassis satellite redundancy-group
            Cluster              Peer          Peer               Local         Device
Name        State                Chassis ID    Chassis SN         Chassis ID    Count
rg2         Online               2             44:f4:77:04:a7:c0  1             0/0/0

EX9208-01-RE0# run show chassis satellite redundancy-group
            Cluster              Peer          Peer               Local         Device
Name        State                Chassis ID    Chassis SN         Chassis ID    Count
rg2         Online               1             44:f4:77:05:ef:c0  2             0/0/0
```

Configure Junos Fusion using the Auto ICCP mode. This allows administrators to deploy Fusion in just a few commands.

First let's rollback to start with a fresh configuration with no ICCP or Fusion:

```
EX9204-01-RE0# run show iccp
Client Application: mclag_cfgchkd
  Redundancy Group IDs Joined: 0 2
Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None

EX9208-01-RE0# run show iccp
Client Application: mclag_cfgchkd
  Redundancy Group IDs Joined: 0 2
Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None
Configure the ICL on EX9204-AD1:
set interfaces xe-2/0/5 description "ICL Layer 2 Link"
set interfaces xe-2/0/5 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-2/0/5 unit 0 family ethernet-switching vlan members all
```

Configure the ICL on EX9208-AD2:

```
set interfaces xe-3/0/4 description "ICL Layer 2 Link"
set interfaces xe-3/0/4 unit 0 family ethernet-switching interface-mode trunk
set interfaces xe-3/0/4 unit 0 family ethernet-switching vlan members all
```

Enable Fusion on the ADs

AD1:

```
set chassis satellite-management redundancy-groups chassis-id 1
set chassis satellite-management redundancy-groups rg2 redundancy-group-id 2
set chassis satellite-management redundancy-groups rg2 peer-chassis-id 2 inter-chassis-link xe-2/0/5
```

AD2:

```
set chassis satellite-management redundancy-groups chassis-id 2
set chassis satellite-management redundancy-groups rg2 redundancy-group-id 2
set chassis satellite-management redundancy-groups rg2 peer-chassis-id 1 inter-chassis-link xe-3/0/4
```

Verify that ICCP is up between the ADs.

AD1:

```
EX9204-01-RE0> show iccp
Redundancy Group Information for peer 10.0.0.2
  TCP Connection       : Established
  Liveliness Detection : Up
  Redundancy Group ID          Status
    2                           Up
Client Application: mclag_cfgchkd
  Redundancy Group IDs Joined: 0 2
Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None
```

AD2:

```
EX9208-01-RE0> show iccp
Redundancy Group Information for peer 10.0.0.1
  TCP Connection       : Established
  Liveliness Detection : Up
  Redundancy Group ID          Status
    2                           Up
Client Application: mclag_cfgchkd
  Redundancy Group IDs Joined: 0 2
Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None
```

Verify Fusion is enabled on both ADs.

AD1:

```
EX9204-01-RE0> show chassis satellite redundancy-group
            Cluster              Peer          Peer               Local         Device
Name        State                Chassis ID    Chassis SN         Chassis ID    Count
rg2         Online               2             44:f4:77:04:a7:c0  1             0/0/0
```

AD2:

```
EX9208-01-RE0> show chassis satellite redundancy-group
            Cluster              Peer          Peer               Local         Device
Name        State                Chassis ID    Chassis SN         Chassis ID    Count
rg2         Online               1             44:f4:77:05:ef:c0  2             0/0/0

*****With Auto ICCP Fusion can be enabled in three commands
*****Will need explanation of Auto ICCP
Convert a VC to a Junos Fusion Satellite Device Cluster
Obtain MAC addresses of the VC members
EX4300-48T-09> show chassis mac-addresses
    FPC 0   MAC address information:
      Public base address     80:ac:ac:6a:3f:a0
      Public count            96
    FPC 1   MAC address information:
      Public base address     80:ac:ac:6a:18:40
      Public count            96
    FPC 2   MAC address information:
      Public base address     80:ac:ac:6a:15:a0
      Public count            96
```

Remove devices from the VC by deleting VC ports from each member. (must be done from the console of each VC member).

```
> request virtual-chassis vc-port delete pic-slot n port n
```

Now, let's configure the SD cluster on the ADs.

AD1:

```
set interfaces xe-2/0/7 cascade-port
set chassis satellite-management cluster Closet2 cluster-id 12
set chassis satellite-management cluster Closet2 cascade-ports xe-2/0/7
set chassis satellite-management cluster Closet2 fpc 140 member-id 0
set chassis satellite-management cluster Closet2 fpc 140 system-id 80:ac:ac:6a:3f:a0
```

```
set chassis satellite-management cluster Closet2 fpc 141 member-id 1
set chassis satellite-management cluster Closet2 fpc 141 system-id 80:ac:ac:6a:18:40
set chassis satellite-management cluster Closet2 fpc 142 member-id 2
set chassis satellite-management cluster Closet2 fpc 142 system-id 80:ac:ac:6a:15:a0
set chassis satellite-management redundancy-groups rg2 cluster Closet2
set chassis satellite-management auto-satellite-conversion satellite 140
set chassis satellite-management auto-satellite-conversion satellite 141
set chassis satellite-management auto-satellite-conversion satellite 142
```

AD2:

```
set interfaces xe-3/0/5 cascade-port
set chassis satellite-management cluster Closet2 cluster-id 12
set chassis satellite-management cluster Closet2 cascade-ports xe-3/0/5
set chassis satellite-management cluster Closet2 fpc 140 member-id 0
set chassis satellite-management cluster Closet2 fpc 140 system-id 80:ac:ac:6a:3f:a0
set chassis satellite-management cluster Closet2 fpc 141 member-id 1
set chassis satellite-management cluster Closet2 fpc 141 system-id 80:ac:ac:6a:18:40
set chassis satellite-management cluster Closet2 fpc 142 member-id 2
set chassis satellite-management cluster Closet2 fpc 142 system-id 80:ac:ac:6a:15:a0
set chassis satellite-management redundancy-groups chassis-id 2
set chassis satellite-management redundancy-groups rg2 redundancy-group-id 2
set chassis satellite-management redundancy-groups rg2 peer-chassis-id 1 inter-chassis-link xe-3/0/4
set chassis satellite-management redundancy-groups rg2 cluster Closet2
set chassis satellite-management auto-satellite-conversion satellite 140
set chassis satellite-management auto-satellite-conversion satellite 141
set chassis satellite-management auto-satellite-conversion satellite 142
```

Add the SNOS image for the SD Cluster on both ADs:

```
> request system software add /var/tmp/satellite-2.0R1.1-signed.tgz upgrade-group Closet2
```

From the console of each access switch, let's `zeroize` the devices:

```
> request system zerioize
***This process takes 15-30 minutes to complete
```

Check the status of the SD cluster on the ADs:

```
EX9204-01-RE0> show chassis satellite cluster Closet2
                        Device          Cascade      Port       Extended Ports
Alias           Slot    State           Ports        State      Total/Up
_sd140          140     Online          xe-142/2/3   present    51/3
                                        xe-141/2/1   present
                                        xe-2/0/5*    backup
_sd141          141     Online          xe-142/2/1   present    51/6
                                        xe-140/2/2   present
                                        xe-2/0/7     online
                                        xe-2/0/5*    backup
_sd142          142     Online          xe-141/2/2   present    50/4
                                        xe-140/2/3   present
                                        xe-2/0/5*    backup

EX9208-01-RE0> show chassis satellite cluster Closet2
                        Device          Cascade      Port       Extended Ports
Alias           Slot    State           Ports        State      Total/Up
_sd140          140     Online          xe-142/2/3   present    51/3
                                        xe-141/2/1   present
```

```
                                    xe-3/0/5     online
                                    xe-3/0/4*    backup
_sd141          141    Online       xe-142/2/1   present    51/6
                                    xe-140/2/2   present
                                    xe-3/0/4*    backup
_sd142          142    Online       xe-140/2/3   present    50/4
                                    xe-141/2/2   present
                                    xe-3/0/4*    backup
```

Let's explore the Junos Fusion Enterprise monitoring commands:

```
EX9204-01-RE0> show iccp
Redundancy Group Information for peer 10.0.0.2
  TCP Connection       : Established
  Liveliness Detection : Up
  Redundancy Group ID          Status
    2                           Up
Client Application: mclag_cfgchkd
  Redundancy Group IDs Joined: 0 2
Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: None

EX9204-01-RE0> show chassis satellite
                        Device          Cascade      Port       Extended Ports
Alias           Slot    State           Ports        State      Total/Up
_sd130          130     Online          xe-131/2/2   present    51/3
                                        xe-2/0/5*    backup
_sd131          131     Online          xe-130/2/1   present    51/2
                                        xe-2/0/6     online
                                        xe-2/0/5*    backup
_sd140          140     Online          xe-142/2/3   present    51/3
                                        xe-141/2/1   present
                                        xe-2/0/5*    backup
_sd141          141     Online          xe-142/2/1   present    51/6
                                        xe-140/2/2   present
                                        xe-2/0/7     online
                                        xe-2/0/5*    backup
_sd142          142     Online          xe-141/2/2   present    50/4
                                        xe-140/2/3   present
                                        xe-2/0/5*    backup

EX9204-01-RE0> show chassis hardware satellite
Hardware inventory:
Item             Version  Part number  Serial number     Description
FPC 130          REV 18   650-044931   PE3715490528      EX4300-48T
  PIC 0          REV 18   BUILTIN      BUILTIN           48x 10/100/1000 Base-T
  PIC 1          REV 18   BUILTIN      BUILTIN           4x 40GE
  PIC 2          REV 02   611-063980   MY3715490631      4x 1G/10G SFP/SFP+
    Xcvr 1       REV 01   740-031980   MVD0KW4           SFP+-10G-SR
    Xcvr 2       REV 01   740-031980   MVD0MH1           SFP+-10G-SR
    Xcvr 3       REV 01   740-031980   MVC11V6           SFP+-10G-SR
Power Supply 0   REV 01   740-046873   1EDE5347288       JPSU-350-AC-AFO
Fan Tray 0       EX4300 Fan Tray 0, Front to Back Airflow - AFO
Fan Tray 1       EX4300 Fan Tray 1, Front to Back Airflow - AFO
FPC 131          REV 18   650-044931   PE3715490478      EX4300-48T
  PIC 0          REV 18   BUILTIN      BUILTIN           48x 10/100/1000 Base-T
  PIC 1          REV 18   BUILTIN      BUILTIN           4x 40GE
  PIC 2          REV 02   611-063980   MY3715491086      4x 1G/10G SFP/SFP+
```

```
    Xcvr 1       REV 01   740-031980   MVD0EKV           SFP+-10G-SR
    Xcvr 2       REV 01   740-031980   MVD0LG9           SFP+-10G-SR
    Xcvr 3       REV 01   740-031980   MVD0EV3           SFP+-10G-SR
Power Supply 0   REV 01   740-046873   1EDE5346518       JPSU-350-AC-AFO
Fan Tray 0       EX4300 Fan Tray 0, Front to Back Airflow - AFO
Fan Tray 1       EX4300 Fan Tray 1, Front to Back Airflow - AFO
FPC 140          REV 18   650-044931   PE3715490468      EX4300-48T
  PIC 0          REV 18   BUILTIN      BUILTIN           48x 10/100/1000 Base-T
  PIC 1          REV 18   BUILTIN      BUILTIN           4x 40GE
  PIC 2          REV 02   611-063980   MY3715490103      4x 1G/10G SFP/SFP+
    Xcvr 1       REV 01   740-031980   MVD0MUR           SFP+-10G-SR
    Xcvr 2       REV 01   740-031980   MVD0EV7           SFP+-10G-SR
    Xcvr 3       REV 01   740-031980   MVC1K7S           SFP+-10G-SR
Power Supply 0   REV 01   740-046873   1EDE5346480       JPSU-350-AC-AFO
Fan Tray 0       EX4300 Fan Tray 0, Front to Back Airflow - AFO
Fan Tray 1       EX4300 Fan Tray 1, Front to Back Airflow - AFO
FPC 141          REV 18   650-044931   PE3715490346      EX4300-48T
  PIC 0          REV 18   BUILTIN      BUILTIN           48x 10/100/1000 Base-T
  PIC 1          REV 18   BUILTIN      BUILTIN           4x 40GE
  PIC 2          REV 02   611-063980   MY3715490529      4x 1G/10G SFP/SFP+
    Xcvr 1       REV 01   740-031980   MVD0E6G           SFP+-10G-SR
    Xcvr 2       REV 01   740-031980   MVD0K4H           SFP+-10G-SR
    Xcvr 3       REV 01   740-031980   MVC1W1L           SFP+-10G-SR
Power Supply 0   REV 01   740-046873   1EDE5346411       JPSU-350-AC-AFO
Fan Tray 0       EX4300 Fan Tray 0, Front to Back Airflow - AFO
Fan Tray 1       EX4300 Fan Tray 1, Front to Back Airflow - AFO
FPC 142          REV 18   650-044931   PE3715490140      EX4300-48T
  PIC 0          REV 18   BUILTIN      BUILTIN           48x 10/100/1000 Base-T
  PIC 1          REV 18   BUILTIN      BUILTIN           4x 40GE
  PIC 2          REV 02   611-063980   MY3715470141      4x 1G/10G SFP/SFP+
    Xcvr 1       REV 01   740-031980   MVD0NZ9           SFP+-10G-SR
    Xcvr 3       REV 01   740-031980   MVD0GH0           SFP+-10G-SR
Power Supply 0   REV 01   740-046873   1EDE5347420       JPSU-350-AC-AFO
Fan Tray 0       EX4300 Fan Tray 0, Front to Back Airflow - AFO
Fan Tray 1       EX4300 Fan Tray 1, Front to Back Airflow - AFO

EX9204-01-RE0> show interfaces terse | except down
Interface               Admin Link Proto    Local                 Remote
lc-2/0/0                up    up
lc-2/0/0.32769          up    up   vpls
pfe-2/0/0               up    up
pfe-2/0/0.16383         up    up   inet
                                   inet6
pfh-2/0/0               up    up
pfh-2/0/0.16383         up    up   inet
pfh-2/0/0.16384         up    up   inet
xe-2/0/0                up    up
xe-2/0/1                up    up
xe-2/0/4                up    up
xe-2/0/4.0              up    up   inet     192.168.10.1/30
                                   multiservice
xe-2/0/4.32767          up    up   multiservice
xe-2/0/5                up    up
xe-2/0/5.0              up    up   eth-switch
xe-2/0/5.32769          up    up   inet     10.0.0.1/30
xe-2/0/5.32770          up    up   eth-switch
xe-2/0/6                up    up
xe-2/0/6.32769          up    up   inet     10.17.0.25/30
xe-2/0/6.32770          up    up   eth-switch
```

```
xe-2/0/7                up    up
xe-2/0/7.32769          up    up   inet     10.17.0.29/30
xe-2/0/7.32770          up    up   eth-switch
xe-2/1/0                up    up
xe-2/1/2                up    up
…
vtep                    up    up
sd-130/0/0              up    up
sd-130/0/0.32770        up    up   eth-switch
ge-130/0/23             up    up
xe-130/2/1              up    up
xe-130/2/3              up    up
sd-131/0/0              up    up
sd-131/0/0.32770        up    up   eth-switch
xe-131/2/1              up    up
xe-131/2/2              up    up
sd-140/0/0              up    up
sd-140/0/0.32770        up    up   eth-switch
xe-140/2/1              up    up
xe-140/2/2              up    up
xe-140/2/3              up    up
ge-141/0/0              up    up
sd-141/0/0              up    up
sd-141/0/0.32770        up    up   eth-switch
ge-141/0/1              up    up
ge-141/0/2              up    up
xe-141/2/1              up    up
xe-141/2/2              up    up
xe-141/2/3              up    up
ge-142/0/0              up    up
sd-142/0/0              up    up
sd-142/0/0.32770        up    up   eth-switch
ge-142/0/2              up    up
xe-142/2/1              up    up
xe-142/2/3              up    up
```

You're now ready to configure user network connectivity on the extended ports (therefore, ge-142/0/2) that connect to users. Now, full management, operations, and provisioning of your enterprise or campus switch network will be done on the Aggregation Device.

In summary, Junos Fusion Enterprise allows for significant scale in campus and enterprise switch networks while greatly simplifying management and operations.

Configuring Generic Route Encapsulation (GRE)

Generic Route Encapsulation (GRE) over EX Series switches provides a secure tunnel to transport packets across the network using the IP encapsulation protocol. GRE tunnels provide a clear path from point to point over an existing routable network.

There are only specific products that are capable of GRE so make sure you use the Juniper Pathfinder tool to verify: https://pathfinder.juniper.net/feature-explorer/feature-info.html?fKey=4026&fn=IPv4+over+generic+routing+encapsulation+(GRE)+tunnels%E2%80%94encapsulation+support.

In the book's lab, there are three devices that are capable of supporting GRE: the QFX5100 Series, the EX4600 Series, and the EX4300 Series can all support GRE tunnels. Let's create a clear path from QFX5100_2_VC to the EX4600_4_VC to show you how easily this is accomplished.

First build the GRE tunnel from interface xe-1/0/0 on the EX4600 to interfaces xe-0/0/3 on the QFX5100. The tunnel will use the routing between 10.1.0.5 and 10.1.0.2 to create a single hop clear path between the two devices. First, let's verify that there is more than one hop:

```
EX4600-4_5-VC# run traceroute 10.1.0.2
traceroute to 10.1.0.2 (10.1.0.2), 30 hops max, 40 byte packets
 1  10.1.0.4 (10.1.0.4)  16.507 ms  11.683 ms  10.980 ms
 2  10.1.0.2 (10.1.0.2)  22.121 ms  22.583 ms  21.970 ms
```

You can clearly see that the EX4300 is between our endpoints. Now let's configure the first endpoint on the EX4600 using 10.100.0.0/31 as the tunnel endpoint on xe-1/0/0:

```
EX4600-4_5-VC# set interfaces gr-0/0/0 unit 0 family inet address 10.100.0.0/31
EX4600-4_5-VC# set interfaces gr-0/0/0 unit 0 tunnel source 10.1.0.5
EX4600-4_5-VC# set interfaces gr-0/0/0 unit 0 tunnel destination 10.1.0.2
EX4600-4_5-VC# set interfaces gr-0/0/0 unit 0 tunnel path-mtu-discovery
EX4600-4_5-VC# commit
```

Once that is committed you can move on to the QFX5100 and do the same thing. The tunnel IP for the QFX5100 will be 10.100.0.1/31:

```
QFX5100-2_3-VC# set interfaces gr-0/0/0.0 family inet address 10.100.0.1/31
QFX5100-2_3-VC# set interfaces gr-0/0/0.0 tunnel source 10.1.0.2
QFX5100-2_3-VC# set interfaces gr-0/0/0.0 tunnel destination 10.1.0.5
QFX5100-2_3-VC# set interfaces gr-0/0/0.0 tunnel path-mtu-discovery
QFX5100-2_3-VC# commit
```

Now that both tunnel endpoints are up, let's see if we can ping the other side:

```
QFX5100-2_3-VC# run ping 10.100.0.0 source 10.100.0.1
PING 10.100.0.0 (10.100.0.0): 56 data bytes
64 bytes from 10.100.0.0: icmp_seq=0 ttl=64 time=12.762 ms
64 bytes from 10.100.0.0: icmp_seq=1 ttl=64 time=11.156 ms
64 bytes from 10.100.0.0: icmp_seq=2 ttl=64 time=99.145 ms
^C
--- 10.100.0.0 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 11.156/41.021/99.145/41.105 ms
```

That was super easy! Now we can send specific traffic down the gre tunnel. Let's use the loopback addresses between the two devices and see what happens with the routing table.

```
QFX5100-2_3-VC# run show route 1.1.1.4
inet.0: 20 destinations, 21 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
1.1.1.4/32         *[OSPF/150] 00:19:03, metric 0, tag 0
                    > to 10.1.0.3 via xe-0/0/3.0

QFX5100-2_3-VC# set protocols ospf area 0.0.0.0 interface gr-0/0/0.0 interface-type p2p

EX4600-4_5-VC# set protocols ospf area 0.0.0.0 interface gr-0/0/0.0 interface-type p2p
EX4600-4_5-VC# run show route 1.1.1.2
inet.0: 22 destinations, 23 routes (22 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
1.1.1.2/32         *[OSPF/150] 00:00:14, metric 0, tag 0
                    > via gr-0/0/0.0

EX4600-4_5-VC# run show route terse | match gr
* ? 1.1.1.2/32         O 150          0            >gr-0/0/0.0
* ? 10.1.0.0/31        O  10          2            >gr-0/0/0.0
* ? 10.1.0.2/31        O  10          2             gr-0/0/0.0
* ? 10.1.0.8/31        O  10          3            >gr-0/0/0.0
* ? 10.100.0.0/31      D   0                       >gr-0/0/0.0
```

By setting up a GRE tunnel from the EX4600 to the QFX5100, and dropping the interface into OSPF, you can create a tunnel directly between the two devices. You can be very selective of the traffic that you want traversing this route and you can treat it as if it were a physical interface. For more information on GRE tunnels visit: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/gre-tunnel-services-qfx-series.html.

Configuring Virtual Routing and Forwarding (VRF-Lite)

One of the really cool things about Juniper EX Series Switches is that they all support VRF-Lite. However, you will need to purchase the Enhanced Feature License (EFL) to run virtual routers on specific models (see the note below). Remember, the Junos OS already has all of the Layer 3 features built in. VRF stands for Virtual Routing and Forwarding and on Junos this is accomplished using the `routing-instance` hierarchy. VRF-Lite is applied to mean that no Multi-Protocol Label Switching (MPLS) is being used to create the Layer-3 Virtual Private Network (VPN).

NOTE Be sure to check which features require licensing on the EX Series switches. The EX4600 already has the EFL along with dual power supplies. To check the model of switches and the features that need license refer to: https://www.juniper.net/documentation/en_US/junos/topics/concept/ex-series-software-licenses-overview.html#jd0e420.

TIP The EX4600 and EX9200 support MPLS. The EX4600 is a beast and uses the same architecture as the QFX5100. You should check out the datasheets here: http://www.juniper.net/us/en/products-services/switching/ex-series/ex4600/.

When you create a VRF in Junos it creates an entirely separate routing table to hold the routes for any ingress or egress traffic applied to the routing instance. Another neat feature about using VRF-Lite across a switched network is that you can have a true Layer 3 core and extend the same VLANs across the core from one side of the network to another. Or how about multicast applications that you would like to keep segregated from the rest of the network? There are tons of reasons why you want to utilize a VRF on the network.

For example, let's disable the ge-1/0/4 interface on EX4600_4 as well as the ge-1/0/0 0interface on QFX5100_2. We've now isolated the two EX2300 access switches to a single VLAN uplink on both sides of the network. What you want to do is extend a VLAN from the EX2300_16 to the EX2300_8 (in our lab topology). This will allow VLAN 1000 to show up and communicate across the network without having to build "Hybrid L2/L3" interfaces in the core to trunk VLANs.

```
EX2300-16# set vlans V1000 vlan-id 1000
EX2300-16# set vlans V1000 l3-interface irb.1000
EX2300-16# set interfaces irb.1000 family inet address 10.0.0.1/25
EX2300-16# set interfaces ge-0/0/2 disable
EX2300-16# set interfaces xe-0/1/0 disable
EX2300-16# delete interfaces ge-0/0/1.0
EX2300-16# set interfaces ge-0/0/1.0 family ethernet-switching vlan members V1000
EX2300-16# commit

EX2300-8_9# set vlans V1000 vlan-id 1000
EX2300-8_9# set vlans V1000 l3-interface irb.1000
EX2300-8_9# set interfaces irb.1000 family inet address 10.0.0.129/25
EX2300-8_9# set interfaces ge-1/0/1 disable
EX2300-8_9# set interfaces xe-0/1/2 disable
EX2300-8_9# delete interfaces ge-0/0/1.0
EX2300-8_9# set interfaces ge-0/0/1.0 family ethernet-switching vlan members V1000
EX2300-8_9# commit
```

Now that we have disabled some interfaces and added some VLANs, the lab network looks like Figure 7.4 – a Layer 3 core and the same VLAN on both sides of the network.

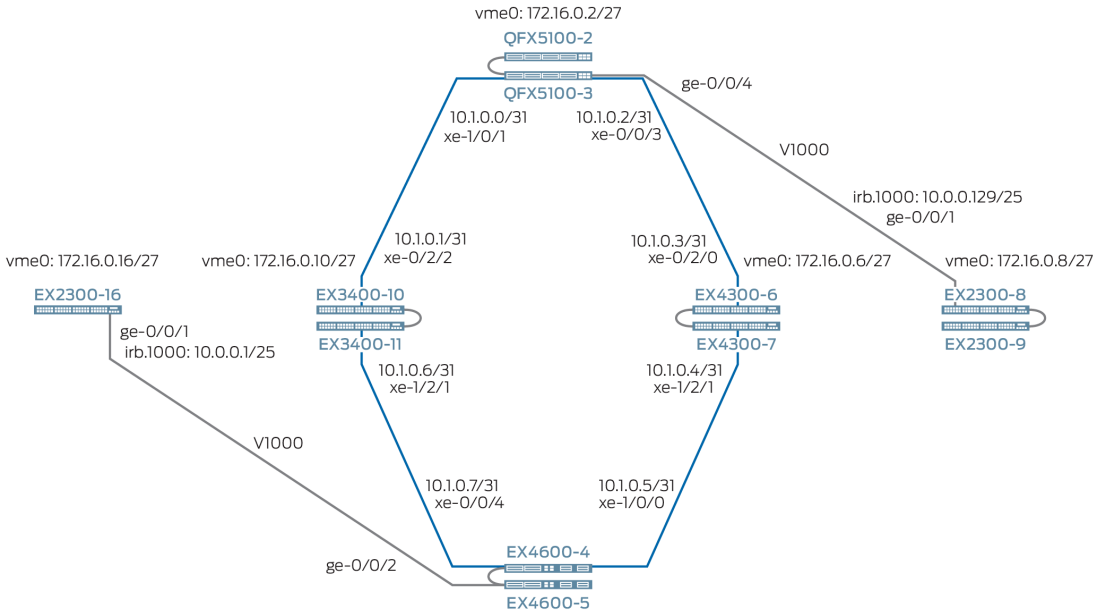


Figure 7.4 Virtual Routing and Forwarding (VRF) topology.

Next let's move to the EX4600_4 switch to where the fun begins:

```
EX4600-4_5-VC# set vlans V1000 vlan-id 1000
EX4600-4_5-VC# set vlans V1000 l3-interface irb.1000
EX4600-4_5-VC# set interfaces irb.1000 family inet address 10.0.0.4/25
EX4600-4_5-VC# set interfaces ge-0/0/2.0 family ethernet-switching vlan members V1000
EX4600-4_5-VC# commit
EX4600-4_5-VC# run ping 10.0.0.1 count 5 rapid source 10.0.0.4
PING 10.0.0.1 (10.0.0.1): 56 data bytes
!!!!!
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.831/28.665/47.457/9.586 ms
```

Since we have confirmed that our VLAN 1000 is operational, let's do the same on the QFX5100 side:

```
QFX5100-2_3-VC# set vlans V1000 vlan-id 1000
QFX5100-2_3-VC# set vlans V1000 l3-interface irb.1000
QFX5100-2_3-VC# set interfaces irb.1000 family inet address 10.0.0.130/25
QFX5100-2_3-VC# set interfaces ge-0/0/4.0 family ethernet-switching vlan members V1000
QFX5100-2_3-VC# commit
QFX5100-2_3-VC# run ping 10.0.0.8
PING 10.0.0.8 (10.0.0.8): 56 data bytes
64 bytes from 10.0.0.8: icmp_seq=0 ttl=64 time=22.674 ms
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=11.174 ms
^C
```

```
--- 10.0.0.8 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 11.174/16.924/22.674/5.750 ms
```

Excellent! We've confirmed that VLAN 1000 exists on both sides of the network and we can ping them from the core. The next thing to do is to create the VRF, or routing instance, on both the QFX5100 and the EX4600:

```
QFX5100-2_3-VC# set routing-instances DO_VRF instance-type virtual-router
QFX5100-2_3-VC# set routing-instances DO_VRF interface irb.1000
QFX5100-2_3-VC# commit
QFX5100-2_3-VC# run ping 10.0.0.8 count 5 rapid
PING 10.0.0.8 (10.0.0.8): 56 data bytes
.....
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
```

Once you place the irb.1000 interface inside the routing instance you are unable to ping from the master routing instance. To ping or traceroute from a routing instance you have to reference the VRF name. For example:

```
QFX5100-2_3-VC# run ping 10.0.0.8 count 5 rapid source 10.0.0.2 routing-instance DO_VRF
PING 10.0.0.8 (10.0.0.8): 56 data bytes
!!!!!
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 9.831/13.292/22.287/4.554 ms
```

You should note that we referenced `routing-instance DO_VRF` to tell ping to use the inet.0 table from that routing instance. Once done, it worked. This shows you that you are segregating your routing tables from the master instance as you move along. Now, let's do the same on the EX4600:

```
EX4600-4_5-VC# set routing-instances DO_VRF instance-type virtual-router
EX4600-4_5-VC# set routing-instances DO_VRF interface irb.1000
EX4600-4_5-VC# commit

EX4600-4_5-VC# run ping 10.0.0.1 rapid count 5
PING 10.0.0.1 (10.0.0.1): 56 data bytes
.....
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
EX4600-4_5-VC# run ping routing-instance DO_VRF 10.0.0.1 rapid count 5
PING 10.0.0.1 (10.0.0.1): 56 data bytes
!!!!!
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 22.915/30.813/34.024/4.056 ms
```

Again, we've added irb.1000 to our VRF on the EX4600 and can only ping the distant end by referencing the `routing-instance DO_VRF`.

Right now, there are two routing instances, one on each side of the network, but there is no connectivity between those routing instances. There are a few options to solve this dilemma:

- Convert all Layer 3 interfaces to Layer 2 trunks and pin-up point-to-point IP addresses across each physical link using VLANs with IRBs.
- Use the `vlan-tagging` keyword on the Layer 3 interfaces and create a new logical unit with just the IP addresses needed.
- Use a GRE tunnel, recently discussed, to carry the traffic from one side of the network to the other and terminate it inside the routing instance.
- Add new Layer 2 links between the two routers, bypassing the rest of the devices.

NOTE Option 1 is often referred to as *hybrid* interfaces because they are Layer 2 and Layer 3 at the same time, but the core is still susceptible to spanning tree.

Option 1 is a no-go. Not only would that be a serious amount of work, and total redesign, it would make the lesson irrelevant because you would no longer need to create a VRF, you could just create a VLAN.

Option 2 is an option. You can use `vlan-tagging` to create sub-interfaces (logical units) and use those in the `routing-instance` to carry traffic on the north bound segment.

Option 4 is an option but it is cost- and time-prohibitive, so not an attractive solution at all.

Option 3 is the clear winner: just configure a GRE tunnel from QFX5100_2 to EX4600_4 and drop that interface into the `routing-instance` to provide connectivity. Let's do that first!

For our GRE tunnel, let's go from xe-1/2/1 on the EX3400_10 to xe-1/2/1 on the EX4300_6, make sure we can ping both ends, and then drop it into the routing-instance:

```
EX4600-4_5-VC# set interfaces gr-0/0/0.0 family inet address 10.3.0.1/30
EX4600-4_5-VC# set interfaces gr-0/0/0.0 tunnel source 10.1.0.5
EX4600-4_5-VC# set interfaces gr-0/0/0.0 tunnel destination 10.1.0.2
EX4600-4_5-VC# commit
EX4600-4_5-VC# run show interfaces terse | match gr-0/0/0
gr-0/0/0                up    up
gr-0/0/0.0              up    up   inet     10.3.0.1/30

QFX5100-2_3-VC# set interfaces gr-0/0/0.0 family inet address 10.3.0.2/30
QFX5100-2_3-VC# set interfaces gr-0/0/0.0 tunnel source 10.1.0.2
QFX5100-2_3-VC# set interfaces gr-0/0/0.0 tunnel destination 10.1.0.5
QFX5100-2_3-VC# commit
QFX5100-2_3-VC# run show interfaces terse | match gr-0/0/0
gr-0/0/0                up    up
gr-0/0/0.0              up    up   inet     10.3.0.2/30
```

Now, ping from one end of the tunnel to the other:

```
EX4600-4_5-VC> ping 10.3.0.2 source 10.3.0.1 count 5 rapid
PING 10.3.0.2 (10.3.0.2): 56 data bytes
!!!!!
--- 10.3.0.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 3.953/10.724/15.621/3.784 ms
```

Awesome! Now you can push this interface into the `routing-instance` on both sides of the network:

```
QFX5100-2_3-VC# set routing-instances DO_VRF interface gr-0/0/0.0

EX4600-4_5-VC# set routing-instances DO_VRF interface gr-0/0/0.0
```

Once you commit those two statements, check the connectivity:

```
EX4600-4_5-VC# run ping 10.3.0.2 count 5 rapid
PING 10.3.0.2 (10.3.0.2): 56 data bytes
.....
--- 10.3.0.2 ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
EX4600-4_5-VC# run ping 10.3.0.2 count 5 rapid routing-instance DO_VRF
PING 10.3.0.2 (10.3.0.2): 56 data bytes
!!!!!
--- 10.3.0.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 10.825/29.545/103.013/36.736 ms
```

Your lab should be similar! But we still need to provide a routing mechanism to get the routing instances to advertise their connectivity. For that, let's configure `ospf area 0` inside the VRF:

```
EX4600-4_5-VC# set routing-instances DO_
VRF protocols ospf area 0.0.0.0 interface irb.1000 interface-type p2p
EX4600-4_5-VC# set routing-instances DO_VRF protocols ospf area 0.0.0.0 interface gr-
0/0/0.0 interface-type p2p
EX4600-4_5-VC# commit

QFX5100-2_3-VC# set routing-instances DO_
VRF protocols ospf area 0.0.0.0 interface irb.1000 interface-type p2p
QFX5100-2_3-VC# set routing-instances DO_VRF protocols ospf area 0.0.0.0 interface gr-
0/0/0.0 interface-type p2p
QFX5100-2_3-VC# commit
```

Once OSPF is configured you should see routes on both sides:

```
QFX5100-2_3-VC# run show route table DO_VRF.inet.0
DO_VRF.inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.0.0.0/25        *[OSPF/10] 01:15:33, metric 2
                    > via gr-0/0/0.0
10.0.0.128/25      *[Direct/0] 01:15:33
                    > via irb.1000
10.0.0.130/32      *[Local/0] 01:15:33
                       Local via irb.1000
10.3.0.0/30        *[Direct/0] 01:25:57
                    > via gr-0/0/0.0
```

```
                    [OSPF/10] 01:25:57, metric 1
                    > via gr-0/0/0.0
10.3.0.2/32         *[Local/0] 01:25:57
                      Local via gr-0/0/0.0
224.0.0.5/32        *[OSPF/10] 01:46:30, metric 1
                      MultiRecv

EX4600-4_5-VC# run show route table DO_VRF.inet.0
DO_VRF.inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
10.0.0.0/25         *[Direct/0] 01:16:19
                     > via irb.1000
10.0.0.4/32         *[Local/0] 01:16:19
                      Local via irb.1000
10.0.0.128/25       *[OSPF/10] 01:15:56, metric 2
                     > via gr-0/0/0.0
10.3.0.0/30         *[Direct/0] 01:26:49
                     > via gr-0/0/0.0
                     [OSPF/10] 01:26:49, metric 1
                     > via gr-0/0/0.0
10.3.0.1/32         *[Local/0] 01:26:49
                      Local via gr-0/0/0.0
224.0.0.5/32        *[OSPF/10] 01:47:20, metric 1
                      MultiRecv
```

Notice a couple of things here. You can see that we referenced the inet.0 table for the routing instance DO_VRF as `DO_VRF.inet.0`. This shows that you have separated the DO_VRF routing table from the master table or inet.0. None of the routes will show up in a standard inet.0 route table.

TIP If you are not familiar with the different routing tables in the Junos OS, try this quick tutorial: https://www.juniper.net/documentation/en_US/junos/topics/concept/routing-tables-understanding.html.

The last thing to do is provide a route to our access layer. You can set a static default route or you can point to the other half of the network. Let's choose the latter and see if we can get end-to-end reachability:

```
EX2300-8_9-VC# set routing-options static route 10.0.0.0/25 next-hop 10.0.0.130
EX2300-8_9-VC# commit

EX2300-16# set routing-options static route 10.0.0.128/25 next-hop 10.0.0.4
EX2300-16# commit
EX2300-8_9-VC# run ping 10.0.0.1 count 5 rapid
PING 10.0.0.1 (10.0.0.1): 56 data bytes
!!!!!
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 69.103/92.536/124.694/23.632 ms
EX2300-16> ping 10.0.0.129 count 10 rapid
PING 10.0.0.129 (10.0.0.129): 56 data bytes
!!!!!!!!!!
--- 10.0.0.129 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
```

```
round-trip min/avg/max/stddev = 64.333/78.790/147.934/23.645 ms
EX2300-16> ssh lab@10.0.0.129
Password:
Last login: Thu Jul  6 21:46:02 2017 from 10.1.1.10
--- Junos 15.1X53-D50.2 Kernel 32-bit  JNPR-11.0-20160614.329646_build
{master:0}
lab@EX2300-8_9-VC> exit
```

There you go. We have successfully connected the same VLAN across the network over a GRE tunnel. We were even able to SSH over it to show the TCP connectivity. However, is isn't finished. This is only one side of the network that the traffic is following. You need to configure a path on the west side as well, but this time let's use the `vlan-tagging` command and drop the interfaces into the VRF:

```
EX4600-4_5-VC# set interfaces xe-0/0/4 vlan-tagging
EX4600-4_5-VC# set interfaces xe-0/0/4.0 vlan-id 10
EX4600-4_5-VC# set interfaces xe-0/0/4.1 vlan-id 1000 family inet address 10.3.0.5/30
EX4600-4_5-VC# set routing-instances DO_VRF interface xe-0/0/4.1
EX4600-4_5-VC# commit

EX3400-10_11-VC# set interfaces xe-0/2/2 vlan-tagging
EX3400-10_11-VC# set interfaces xe-0/2/2 unit 0 vlan-id 10
EX3400-10_11-VC# set interfaces xe-0/2/2 unit 1 vlan-id 1000 family inet address 10.3.0.9/30
EX3400-10_11-VC# set interfaces xe-1/2/1 vlan-tagging
EX3400-10_11-VC# set interfaces xe-1/2/1 unit 0 vlan-id 10
EX3400-10_11-VC# set interfaces xe-1/2/1 unit 1 vlan-id 1000 family inet address 10.3.0.6/3
EX3400-10_11-VC# set routing-instances DO_VRF instance-type virtual-router
EX3400-10_11-VC# set routing-instances DO_VRF interface xe-0/2/2.1
EX3400-10_11-VC# set routing-instances DO_VRF interface xe-1/2/1.1
EX3400-10_11-VC# set routing-instances DO_VRF protocols ospf area 0.0.0.0 interface xe-
1/2/1.1 interface-type p2p
EX3400-10_11-VC# set routing-instances DO_VRF protocols ospf area 0.0.0.0 interface xe-
0/2/2.1 interface-type p2p
EX3400-10_11-VC# commit

QFX5100-2_3-VC# set interfaces xe-1/0/1 vlan-tagging
QFX5100-2_3-VC# set interfaces xe-1/0/1 unit 0 vlan-id 10
QFX5100-2_3-VC# set interfaces xe-1/0/1 unit 1 vlan-id 1000 family inet address 10.3.0.10/30
QFX5100-2_3-VC# commit
```

Once the `vlan-tagging` keyword is used all sub interfaces need a `vlan-id`. We used `vlan-id 10` for the normal point-to-point traffic and will use `vlan-id 1000` for our V1000 traffic on unit 1 of all the interfaces. Now that the path is built, you need to drop the interfaces in the routing instance and place them into the OSPF area. You should notice in the above configuration that the EX3400_10 also has the `DO_VRF routing-instance` now, as well.

After everything is committed you should have reachability in your lab:

```
QFX5100-2_3-VC# run ping routing-instance DO_VRF 10.3.0.5 count 5 rapid
PING 10.3.0.5 (10.3.0.5): 56 data bytes
!!!!!
--- 10.3.0.5 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 32.908/55.580/112.970/29.846 ms
```

```
QFX5100-2_3-VC# run traceroute routing-instance DO_VRF 10.3.0.5
traceroute to 10.3.0.5 (10.3.0.5), 30 hops max, 40 byte packets
 1  10.3.0.9 (10.3.0.9)  86.509 ms  24.447 ms  22.645 ms
 2  10.3.0.5 (10.3.0.5)  58.308 ms  55.040 ms  58.979 ms

And finally, let's ping from end-to-end using the new path and vlan-tagging.
EX2300-16> ping 10.0.0.129 count 50 rapid
PING 10.0.0.129 (10.0.0.129): 56 data bytes
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
--- 10.0.0.129 ping statistics ---
50 packets transmitted, 50 packets received, 0% packet loss
round-trip min/avg/max/stddev = 8.665/14.189/22.671/4.445 ms

EX2300-16> traceroute 10.0.0.129
traceroute to 10.0.0.129 (10.0.0.129), 30 hops max, 52 byte packets
 1  10.0.0.4 (10.0.0.4)  44.409 ms  67.677 ms  45.950 ms
 2  10.3.0.6 (10.3.0.6)  10.967 ms  14.641 ms  10.985 ms
 3  10.3.0.10 (10.3.0.10)  43.208 ms  46.107 ms  35.762 ms
 4  10.0.0.129 (10.0.0.129)  11.718 ms  6.953 ms  10.885 ms
```

We deactivated the gr-0/0/0 interface in the lab to make sure that the traffic took the new path we created. Let's see what happens when that interface is reactivated:

```
EX2300-16> traceroute 10.0.0.129
traceroute to 10.0.0.129 (10.0.0.129), 30 hops max, 52 byte packets
 1  10.0.0.4 (10.0.0.4)  35.460 ms  48.465 ms  34.289 ms
 2  10.3.0.2 (10.3.0.2)  53.690 ms  47.345 ms  55.257 ms
 3  10.0.0.129 (10.0.0.129)  80.624 ms  75.877 ms  77.272 ms
```

Because the GRE tunnel is a single hop, it is the OSPF-preferred path and traffic will follow that route. If you had two GRE tunnels on the networks they would be Equal Cost Multi Path (ECMP) routes and traffic would load balance across the network inside a VRF.

Summary

This chapter has discussed some of the more complicated switching scenarios, including MC-LAG, Junos Fusion Enterprise, GRE Tunnels, and Virtual Routing and Forwarding (VRF). All of these technologies are technical responses to business needs, and can help resolve management of the network as well as getting packets from point A to point B.

The Juniper web site has tons of information on all of these, from the TechLibrary to *Day One* to *Solution Briefs*. Check them all out at http://www.juniper.net/.

Chapter 8

Configuring EX Spanning Tree Protocols

Why Is Spanning Tree Still important?

The first Spanning Tree Protocol (STP) was developed in 1985 by Radia Perlman while working at the Digital Equipment Corporation (DEC). It became an IEEE standard in 1990 as 802.1D and was subsequently updated in 1998 and 2004 incorporating numerous updates.

The purpose behind spanning tree is to allow redundant links in a network topology while at the same time preventing loops and broadcast storms. If you have ever been on a Layer 2 network without spanning tree and have witness it just melt down, you will obviously have an appreciation for STP.

IEEE brought 802.1D in as a standard but other vendors took off on their own and created proprietary implementations of spanning tree that do not work well with open standards. This mindset locks customers into a specific vendor and causes issues with interoperability so be aware of using proprietary protocols in your network!

That's it for history, now let's actually discuss what STP is, how it prevents loops, and the fun part, how to configure it. To do that you need a simple switch topology with each switch having redundant links. You need to know the chassis MAC address for each of the switches.

Figure 8.1 depicts a small four-node network with redundant links connecting each of the nodes and an additional link between Switch-2 and Switch-4. (This is a typical training scenario used in Juniper's documentation and likely to be seen in certification testing.) Each switch has the neighboring interface labeled as well as it's chassis MAC address annotated and the unique IP that it is reachable by.

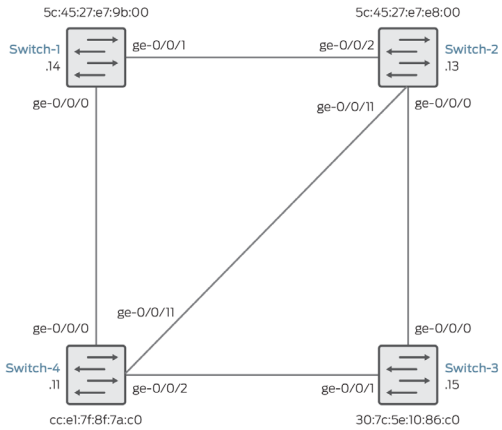


Figure 8.1 *Spanning Tree Protocol (STP) topology.*

Without spanning tree in Figure 8.1, this little network will melt down and become unusable in seconds. We've reduced the entire configuration to just the management piece allowing SSH access to the me0.0 interface and removed all protection and only have Layer 2 interfaces connected to each other. Actually, that is not true, we have ge-0/0/0 on Switch-1 and ge-0/0/11 on Switch-4 disconnected before starting this little experiment.

The reason those interfaces are disconnected is to break any loops in the topology, otherwise, we couldn't the experiment.

CAUTION This is being done in the lab. *Do not attempt this on any production network!*

The following configuration is applied to all four switches with only the IP address for me0.0 and the `host-name` changed per switch:

```
SWITCH-1> show configuration | display set
set version 15.1R6-S2.1
set groups LAB system host-name SWITCH-1
set groups LAB system backup-router 192.168.2.5
set groups LAB system authentication-order radius
set groups LAB system authentication-order password
set groups LAB system root-authentication encrypted-password "$1$4SUDuuFj$ZvqhdxrLzJWYkfDhraZt90"
set groups LAB system name-server 8.8.8.8
set groups LAB system login message "\n \n LAB SWITCH-1\n \n"
set groups LAB system login user lab uid 2000
set groups LAB system login user lab class super-user
set groups LAB system login user lab authentication encrypted-password "$1$zWm9E0R2$vC1i0E9.
tI9HfZ0Mzszqv."
set groups LAB system login user lab uid 2002
```

```
set groups LAB system login user lab class super-user
set groups LAB system services ssh
set groups LAB system services netconf ssh
set groups LAB interfaces me0 unit 0 family inet address 192.168.2.14/24
set groups LAB routing-options static route 0.0.0.0/0 next-hop 192.168.2.1
set apply-groups LAB
set interfaces ge-0/0/0 unit 0 family ethernet-switching
set interfaces ge-0/0/1 unit 0 family ethernet-switching
set interfaces ge-0/0/2 unit 0 family ethernet-switching
set interfaces ge-0/0/3 unit 0 family ethernet-switching
set interfaces ge-0/0/4 unit 0 family ethernet-switching
set interfaces ge-0/0/5 unit 0 family ethernet-switching
set interfaces ge-0/0/6 unit 0 family ethernet-switching
set interfaces ge-0/0/7 unit 0 family ethernet-switching
set interfaces ge-0/0/8 unit 0 family ethernet-switching
set interfaces ge-0/0/9 unit 0 family ethernet-switching
set interfaces ge-0/0/10 unit 0 family ethernet-switching
set interfaces ge-0/0/11 unit 0 family ethernet-switching
set interfaces ge-0/1/0 unit 0 family ethernet-switching
set interfaces ge-0/1/1 unit 0 family ethernet-switching
```

NOTE Notice the `groups` hierarchy. Groups are an easy way to apply large sections of code at a time. You could change the personality of this device very easy by having multiple groups that give it different host-names, IP addresses, etc.

Next, let's clear all the interface statistics on each switch and then plug in the connections on Switch-1 and Switch-4, and then ping from an off-network device to see response times. Okay, let's melt a Layer 2 network!

This is the ping that was started and you can see that sequence 17 is where we plugged in the cable creating the loop. Immediately, the broadcast storm that ensued enveloped all the devices and we were unable to reach this device. ICMP times are shown in seconds, and even the connectivity is sporadic at best.

```
$ ping 192.168.2.14
PING 192.168.2.14 (192.168.2.14): 56 data bytes
64 bytes from 192.168.2.14: icmp_seq=0 ttl=62 time=2.708 ms
 …
64 bytes from 192.168.2.14: icmp_seq=17 ttl=62 time=10.623 ms
Request timeout for icmp_seq 18
Request timeout for icmp_seq 19
Request timeout for icmp_seq 20
Request timeout for icmp_seq 21
Request timeout for icmp_seq 22
Request timeout for icmp_seq 23
Request timeout for icmp_seq 24
Request timeout for icmp_seq 25
Request timeout for icmp_seq 26
64 bytes from 192.168.2.14: icmp_seq=18 ttl=62 time=9793.608 ms
64 bytes from 192.168.2.14: icmp_seq=19 ttl=62 time=8793.551 ms
64 bytes from 192.168.2.14: icmp_seq=20 ttl=62 time=7791.432 ms
64 bytes from 192.168.2.14: icmp_seq=21 ttl=62 time=6788.174 ms
64 bytes from 192.168.2.14: icmp_seq=22 ttl=62 time=5785.347 ms
64 bytes from 192.168.2.14: icmp_seq=23 ttl=62 time=4780.493 ms
```

The network is so overwhelmed that it cannot even establish a simple SSH connection.

```
$ ssh 192.168.2.14
```

This connection never establishes as long as the network is under siege! But once you break the loop all of the devices will be reachable again.

CAUTION! Again. *Do not do this on a production network*.

Let's take a look at the interface statistics on ge-0/0/0 of Switch-1 and Switch-4:

```
SWITCH-1> show interfaces ge-0/0/0 detail
Physical interface: ge-0/0/0, Enabled, Physical link is Down
  Interface index: 130, SNMP ifIndex: 502, Generation: 133
  Link-level type: Ethernet, MTU: 1514, LAN-
PHY mode, Speed: Auto, Duplex: Auto, BPDU Error: None, MAC-
REWRITE Error: None, Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled, Auto-
negotiation: Enabled,
  Remote fault: Online, Media type: Copper, IEEE 802.3az Energy Efficient Ethernet: Disabled
  Device flags   : Present Running Down
  Interface flags: Hardware-Down SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Hold-times     : Up 0 ms, Down 0 ms
  Current address: 5c:45:27:e7:9b:03, Hardware address: 5c:45:27:e7:9b:03
  Last flapped   : 2010-01-02 22:48:04 UTC (00:00:25 ago)
  Statistics last cleared: 2010-01-02 22:38:05 UTC (00:10:24 ago)
  Traffic statistics:
   Input  bytes  :          43045419392                 0 bps
   Output bytes  :          43044399296                 0 bps
   Input  packets:            672584678                 0 pps
   Output packets:            672568739                 0 pps
```

Wow! In just over a minute, this interface transmitted and received 672,584,678 packets! That, ladies and gentlemen, is one heck of a storm.

NOTE Good point for a segue. In 1988 they couldn't even fathom speeds greater than 1Gbps (trust me, I was alive then) but now we're looking at standardizing on 40Gbps for the enterprise and 100G for the WAN! This made the old costing of the original spanning tree obsolete for anything over 1Gbps because we started going into fractions. When RSTP was defined, they used a new method for calculating port cost:

- OLD 1Gbps/ Interface Speed in bps
- NEW 20Tbps/ Interface speed in bps

Now we have a more realistic costing. Table 8.1 is what RSTP Port Costs look like.

Table 8.1 *Junos RSTP Port Costs.*

Port Speeds in Gbps	RSTP
1	20000
10	2000
25	800
40	500
50	400
100	200

So, be aware that port costs can impact spanning tree and to make sure that in a mixed vendor environment everyone is on the same costing plan. Now, back to the storm.

Okay, so what have we learned? Loops in Layer 2 networks are *bad!* Now let's apply Rapid Spanning Tree Protocol (RSTP) and see if we can quell the storm. Again, let's clear all of the interface statistics on all of the switches but also add the following RSTP configuration to each switch:

```
SWITCH-1# show protocols rstp
interface all {
    mode point-to-point;
}
```

Here, you can see that RSTP is engaged in protecting the network and interface ge-0/0/0 is still not connected. Again, let's start a ping and then connect the interface. This time however no packets were lost and pings were an average of 3.5ms round-trip:

```
$ ping 192.168.2.14
PING 192.168.2.14 (192.168.2.14): 56 data bytes
64 bytes from 192.168.2.14: icmp_seq=0 ttl=62 time=3.157 ms
64 bytes from 192.168.2.14: icmp_seq=1 ttl=62 time=3.960 ms
64 bytes from 192.168.2.14: icmp_seq=2 ttl=62 time=4.019 ms
64 bytes from 192.168.2.14: icmp_seq=3 ttl=62 time=3.211 ms
64 bytes from 192.168.2.14: icmp_seq=4 ttl=62 time=2.617 ms
64 bytes from 192.168.2.14: icmp_seq=5 ttl=62 time=3.332 ms
64 bytes from 192.168.2.14: icmp_seq=6 ttl=62 time=4.889 ms
64 bytes from 192.168.2.14: icmp_seq=7 ttl=62 time=3.227 ms
^C
--- 192.168.2.14 ping statistics ---
8 packets transmitted, 8 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 2.617/3.551/4.889/0.659 ms
```

Now, the spanning-tree for ge-0/0/0:

```
SWITCH-1> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.cce17f8f7ac1    20000 BLK   ALT
ge-0/0/1.0             128:514   128:515 32768.5c4527e7e801    20000 FWD   ROOT
```

You can also tell that SWITCH-1 is not the root bridge for the network because there's a `ROOT` Role in the output. You can look into this further by issuing the `spanning-tree bridge` statement:

```
SWITCH-1> show spanning-tree bridge
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : RSTP
  Root ID                           : 32768.30:7c:5e:10:86:c1
  Root cost                         : 40000
  Root port                         : ge-0/0/1.0
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Message age                       : 2
  Number of topology changes        : 4
  Time since last topology change   : 105 seconds
  Topology change initiator         : ge-0/0/1.0
  Topology change last recvd. from  : 5c:45:27:e7:e8:05
  Local parameters
    Bridge ID                       : 32768.5c:45:27:e7:9b:01
    Extended system ID              : 0
    Internal instance ID            : 0
```

Based on Figure 8.1 with the annotated chassis mac-addresses you can tell that the root bridge is SWITCH-3. Let's take a look at all of the spanning tree interfaces so you can see which links were cut off:

```
SWITCH-1> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.cce17f8f7ac1    20000 BLK   ALT
ge-0/0/1.0             128:514   128:515 32768.5c4527e7e801    20000 FWD   ROOT

SWITCH-2> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.307c5e1086c1    20000 FWD   ROOT
ge-0/0/2.0             128:515   128:515 32768.5c4527e7e801    20000 FWD   DESG

SWITCH-3> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.307c5e1086c1    20000 FWD   DESG
ge-0/0/1.0             128:514   128:514 32768.307c5e1086c1    20000 FWD   DESG
```

```
SWITCH-4> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                                port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.cce17f8f7ac1    20000 FWD   DESG
ge-0/0/2.0             128:515   128:514 32768.307c5e1086c1    20000 FWD   ROOT
```

From the spanning tree bridge output on Switch-1 you can see that the Root ID is `32768.30:7c:5e:10:86:c1` and the mac address portion of the root ID matches the MAC on Switch-3. You can also tell that the Role is DESG (Designated) which indicates that both interfaces are located on the root bridge.

NOTE The first part of the root ID is the priority. Since we didn't set a bridge priority they will all default to 32K. Switch-3 was chosen as the root bridge because it has the lowest MAC.

To determine the lowest MAC you start with the left-most significant nibble. A full byte is 16 bits represented in hexadecimal from 00-FF. A nibble is half of a byte and based on Figure 8.1 we have the following nibbles:

- Switch-1 = 5
- Switch-2 = 5
- Switch-3 = 3
- Switch-4 = C

You know C represents 12 in HEX, so Switch-3 is the obvious winner for the lowest MAC. If the first nibble were all the same, you would move to the next nibble and so on until you found the lowest MAC.

When you see the Role of `ROOT`, it indicates the path to the root bridge is upstream on this interface. When you see the Role of `DESG`, this indicates the downstream path from the root bridge. You should also notice that there is an interface on SWITCH-1 that indicates State as `BLK`, or blocked, and a Role of `ALT`, or alternate. This is where the loop was broken by spanning tree. Let's map this all out in Figure 8.2 and then add the ge-0/0/11 interface to see what additional information can be gathered.

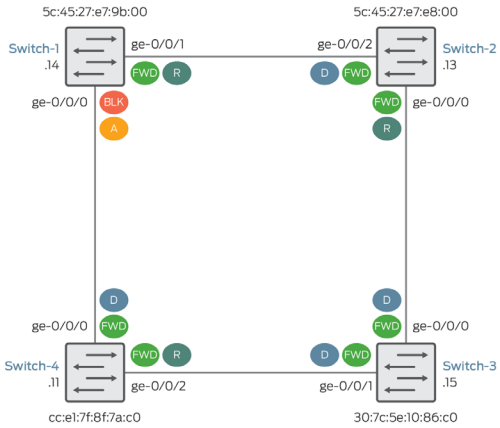


Figure 8.2 Spanning Tree Topology with States

Based on the state and roles of all the links, Switch-3 is indeed the root bridge in the spanning tree topology. You can surmise that if anything happens on the interface between Switch-2 and Switch-3 that the ge-0/0/0 interface will change to `FWD ROOT` for its State and Role, and that ge-0/0/1 on Switch-1 will change to `FWD DESG`, because it will be advertising the route to the root switch, and Switch-2 interface ge-0/0/2 will change to `FWD ROOT`.

Let's disconnect ge-0/0/0 on Switch-2 and see if this is correct:

```
SWITCH-2# set interfaces ge-0/0/0 disable
SWITCH-2# commit
configuration check succeeds
commit complete
```

Now a quick look at Switch-1 and Switch-2 shows:

```
SWITCH-1> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID       bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.cce17f8f7ac1    20000 FWD   ROOT
ge-0/0/1.0             128:514   128:514 32768.5c4527e79b01    20000 FWD   DESG

SWITCH-2> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID       bridge ID        Cost
ge-0/0/2.0             128:515   128:514 32768.5c4527e79b01    20000 FWD   ROOT
```

Just as expected! Let's enable interface ge-0/0/0 on Switch-2 and also add the link inbetween Switch-2 and Switch-4 on ge-0/0/11.

Adding an additional loop in the mix is illustrated in Figure 8.3.

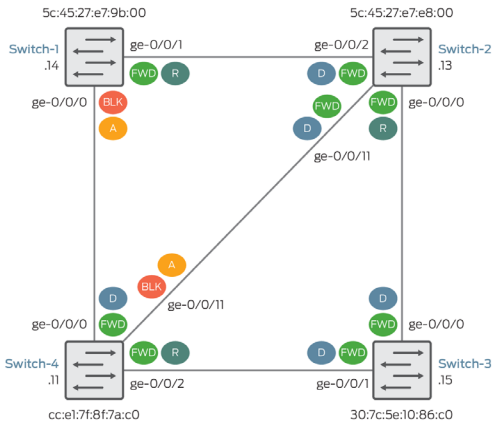


Figure 8.3 *Spanning Tree with Multiple Loops.*

Again, the spanning tree algorithm has found and blocked the loop. It should be noted that this is Rapid Spanning Tree Protocol (RSTP) and not normal STP or IEEE 802.1D.

MORE? The Junos OS defaults to RSTP. If you want to enable basic STP, visit here: https://www.juniper.net/documentation/en_US/junos/topics/concept/spanning-trees-ex-series-basic-stp-understanding.html.

While we have taken time to identify and document the spanning tree topology, we also need to be able to influence our network and also identify the best protocols for our network. That's up next, so if you need more information on BPDU's and Spanning Tree operation refer to the TechLibrary: https://www.juniper.net/documentation/en_US/junos12.3/topics/concept/spanning-trees-ex-series-rstp-understanding.html

Configuring and Monitoring Rapid Spanning Tree Protocol (RSTP)

Enough spanning tree and let's focus on RSTP. The original IEEE 802.1D specification was okay for its time, but it was slow and had some shortcomings. Rapid Spanning Tree Protocol (RSTP) was developed to answer the slowness and also address some of those shortcomings. The original 802.1D specification took up to 50 seconds to converge while RSTP is 6 seconds or three hello BPDU's.

There are four states in STP:

- *Discarding:* Does not learn MAC addresses and discards any Bridge Protocol Data Units (BPDUs) received on the interface.
- *Blocking:* Only listens for BPDUs does not transmit or receive. This stage has a hello timer with a default of 2 seconds.
- *Listening:* If an interesting BPDU is recognized during the blocking state the interface transitions to Listening and starts receiving BPDUs. This stage has a forwarding-delay timer with a default of 15 seconds.
- *Learning:* Receives frames and inspects them to build its forwarding table. This state has a forwarding-delay timer with a default of 15 seconds.
- *Forwarding:* This port is part of the active spanning tree topology and forwards frames. This state has a max age timer and defaults to 20 seconds.

RSTP reduces the number of states to three:

- Discarding
- Learning
- Forwarding

One major update RSTP introduced is two new port states:

- The *alternate port* state indicates that it has an alternate path to the root bridge in the event the primary path, fails.
- The *backup port* indicates that it is a redundant designated port (downstream from root) if a designated port already exists.

There are five roles in RSTP:

- *Designated:* Sends traffic toward the non-root devices (downstream) in the network.
- *Root:* Sends traffic toward the root bridge (upstream).
- *Alternate:* Provides an alternate path to the root bridge in case the root-port is down.
- *Backup:* Provides an alternate path where there are two or more designated ports on the same network. Usually placed in discard state.
- *Disabled:* Not part of the spanning tree topology.

NOTE Discarding is not an actual state in RSTP, it just means the interface is not participating.

NOTE For more information on timers and stages associated with RSTP please see the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/reference/configuration-statement/forward-delay-spanning-trees-ex-series.html.

Another improvement RSTP makes is the marking of edge ports. If you utilize this command you can eliminate BPDUs on access ports. This also enables the access device to connect quicker by eliminating the need to go through the discarding, learning, and forwarding process. Let's run through a quick example of configuring edge for RSTP interfaces:

```
SWITCH-2# set protocols rstp interface ge-0/0/0 edge
SWITCH-2# set protocols rstp interface ge-0/0/0 edge
SWITCH-2# set protocols rstp bpdu-block-on-edge
```

NOTE RSTP is enabled by default on all EX Series Switches to protect the network from loops.

NOTE The `edge` keyword on RSTP interfaces indicates that those interfaces will not send or accept BPDUs. This is the equivalent of "Portfast" to coin another vendor's terminology.

Before getting into the RSTP features that influence the spanning tree topology, let's discuss how RSTP works. BPDU's are sent from all the switches in the network containing information about itself. Those BPDUs contain a ton of information such as bridge priority, mac addresses, ports, path costs, etc.

Step 1 – Select the Root Bridge:

Select the lowest bridge id (default is 32K)

- This is a configurable option under RSTP
- 4k to 60k

If all bridge ids are equal select the lowest MAC address

- Make sure to protect against older devices with lower MAC addresses

Step 2 –Identify Root Ports (non-root devices):

Determine the cost to the Root Bridge.

- Higher speed interfaces have a lower cost
- Lowest cost per interface is preferred
- For ties, the Bridge Priority plus MAC address is used

Step 3 – Identify Designated Ports:

Determine path cost.

- Designated is downstream and should having matching Root
- Designated Ports should always be Forwarding

Here is a packet capture using the `monitor traffic interface` command that allows you to see a lot of the information and how it is being transmitted:

```
SWITCH-3> monitor traffic interface ge-0/0/0 no-resolve extensive
Address resolution is OFF.
Listening on ge-0/0/0, capture size 1514 bytes
15:58:04.865754 Out
           Juniper PCAP Flags [Ext], PCAP Extension(s) total length 16
             Device Media Type Extension TLV #3, length 1, value: Ethernet (1)
             Logical Interface Encapsulation Extension TLV #6, length 1, value: Ethernet (14)
             Device Interface Index Extension TLV #1, length 2, value: 144
             Logical Interface Index Extension TLV #4, length 4, value: 81
           -----original packet-----
           30:7c:5e:10:86:c3 > 01:80:c2:00:00:00, 802.3, length 53: LLC, dsap STP (0x42) Individual,
ssap STP (0x42) Command, ctrl 0x03: STP 802.1w, Rapid STP, Flags [Learn, Forward], bridge-id
8000.30:7c:5e:10:86:c1.8201, length 36
           message-age 0.00s, max-age 20.00s, hello-time 2.00s, forwarding-delay 15.00s
           root-id 8000.30:7c:5e:10:86:c1, root-pathcost 0, port-role Designated
```

You can see under the `original packet` section, that the MAC address of Switch-3 is using the spanning tree multicast address of 01:80:c2:00:00:00 to advertise the MAC, the root-id, the root-pathcost, and with the port-role of `Designated`. Notice that this interface is in `Forward` state and is indeed using Rapid STP.

NOTE The `monitor traffic interface` command is a very powerful tool that is available right in the CLI. It allows you to view the packets in real time or send them to a file or off-box. For more information on the monitor traffic command check out the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/reference/command-summary/monitor-traffic.html.

Now that you are armed with all of that information you should be able to influence the path through the 4-node lab. Let's take on the challenge of making Switch-1 the root bridge, and then using cost and priorities, we should be able to make the path look more like a letter *Z*.

Making sure that Switch-1 is the root bridge for the network is as simple as assigning it the lowest root bridge priority. Set Switch-1 to 4k, Switch-2 to 8k, Switch-3 to 16k, and leave Switch-4 alone as it will default to 32k.

```
SWITCH-1# set protocols rstp bridge-priority 4k
SWITCH-2# set protocols rstp bridge-priority 8k
SWITCH-3# set protocols rstp bridge-priority 16k
```

Now, you can check to see who is the root bridge in the topology:

```
SWITCH-1> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  4096.5c4527e79b01    20000 FWD   DESG
ge-0/0/1.0             128:514   128:514  4096.5c4527e79b01    20000 FWD   DESG

SWITCH-1> show spanning-tree bridge detail
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : RSTP
  Root ID                           : 4096.5c:45:27:e7:9b:01
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Message age                       : 0
  Number of topology changes        : 6
  Time since last topology change   : 254 seconds
  Topology change initiator         : ge-0/0/0.0
  Topology change last recvd. from  : 5c:45:27:e7:e8:05
  Local parameters
    Bridge ID                       : 4096.5c:45:27:e7:9b:01
    Extended system ID              : 0
    Internal instance ID            : 0
    Hello time                      : 2 seconds
    Maximum age                     : 20 seconds
    Forward delay                   : 15 seconds
    Path cost method                : 32 bit
```

And just that simple, we are able to have control over our spanning tree topology. This is an important first step in configuring any type of spanning tree for your network. You don't want to have your root bridge sitting on an access switch somewhere just because it is an old switch with a lower mac address. That could be very bad for your network!

But we aren't done yet. We still want to influence our network and force it into a *Z*-shape topology. The next tool is using path cost:

```
SWITCH-2# set protocols rstp interface ge-0/0/2 cost 10
SWITCH-3# set protocols rstp interface ge-0/0/0 cost 100
SWITCH-3# set protocols rstp interface ge-0/0/1 cost 10

SWITCH-4# set protocols rstp interface ge-0/0/0 cost 100
SWITCH-4# set protocols rstp interface ge-0/0/11 cost 10
SWITCH-4# set protocols rstp interface ge-0/0/2 cost 10
```

Once these are all committed we will have indeed influenced our path so let's map it out again on the diagram.

```
SWITCH-1> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  4096.5c4527e79b01    20000 FWD   DESG
ge-0/0/1.0             128:514   128:514  4096.5c4527e79b01    20000 FWD   DESG
```

```
SWITCH-2# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                                 port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  8192.5c4527e7e801    20000 FWD   DESG
ge-0/0/2.0             128:515   128:514  4096.5c4527e79b01       10 FWD   ROOT
ge-0/0/11.0            128:524   128:524  8192.5c4527e7e801    20000 FWD   DESG

SWITCH-3# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                                 port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  8192.5c4527e7e801      100 BLK   ALT
ge-0/0/1.0             128:514   128:515 32768.cce17f8f7ac1       10 FWD   ROOT

SWITCH-4# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                                 port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  4096.5c4527e79b01      100 BLK   ALT
ge-0/0/2.0             128:515   128:515 32768.cce17f8f7ac1       10 FWD   DESG
ge-0/0/11.0            128:524   128:524  8192.5c4527e7e801       10 FWD   ROOT
```

And now our path looks like Figure 8.4.

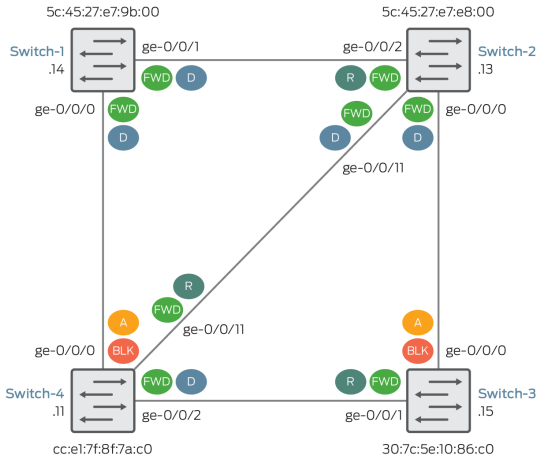


Figure 8.4 *Spanning Tree Topology with Multiple Loops.*

Interface cost has influenced the network into a Z pattern. We could do the same with port cost but you should get the picture by now. If you want to learn more about influencing your spanning tree topology using the RSTP mechanism's see the TechLibary's docs.

With all of the features that are built into RSTP and all the improvements made, RSTP is still not ideal for multiple VLAN networks because it is

not a VLAN-aware protocol. This forces all VLANs to use the same spanning tree topology and may not be the most optimized path for individual VLANs. Good thing the topic of VLAN-aware spanning tree protocols is next.

Configuring and Monitoring Multiple Spanning Tree Protocol (MSTP)

MSTP has the ability to map multiple spanning tree instances (not just a single instance like RSTP) across one physical topology. Each one of those instances can contain one or more VLANs and can be load balanced across the network (unlike RSTP/STP, which cannot load balanced across equal cost paths). MSTP also improves on scaling from RSTP which is limited to 4096 VLANs. With MSTP you can configure up to 64 Multiple Spanning Tree Instances (MSTIs) and each instance can carry VLANs 1-4094.

Common and Internal Spanning Tree (CIST) is used to manage and interconnect all of the different MSTP Regions. It can also manage individual devices running RSTP or STP. CIST treats each of the independent regions as a virtual bridge and allows MSTI's to connect to other MSTP regions.

The CIST is a single autonomous system that connects all of the STP, RSTP, and MSTP devices through a single active topology.

A word of caution – configuring MSTP is not as easy as RSTP or even VSTP but if you break it down into logical chunks, it will make it easier. What do I mean by a logical chunk? Well let's take the topology we've been working with and create two Multiple Spanning Tree Instances (MSTIs).

Instance 1 is illustrated in Figure 8.5.

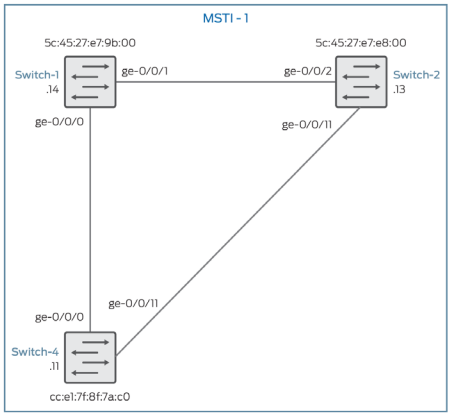


Figure 8.5 *Single Spanning Tree Instance-1*

Instance 2 is illustrated in Figure 8.6.

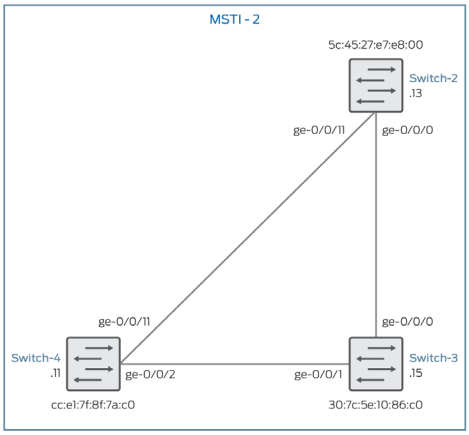


Figure 8.6 *Single Spanning Tree Instance-2*

All the CIST encompasses the entire topology in Figure 8.7.

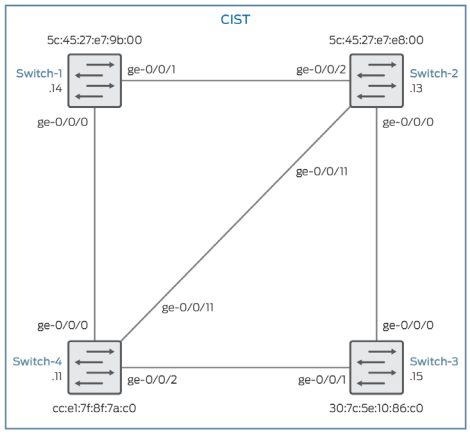


Figure 8.7 *CIST Spanning Tree Topology.*

And then we can break it down a little further:

- We are only going to configure one region and four VLANs for this exercise.
- Instance 1 will carry VLANS 100 and 200.
- Instance 2 will carry VLANS 300 and 400.

- Switch-1 will be the root bridge for Instance 1.
- Switch-3 will be the root bridge for Instance 2.
- Switch-2 will be the root bridge for the CIST.
- The CIST will span the entire topology.

```
SWITCH-1# set vlans V100 vlan-id 100
SWITCH-1# set vlans V200 vlan-id 200
SWITCH-1# set vlans V300 vlan-id 300
SWITCH-1# set vlans V400 vlan-id 400
SWITCH-1# commit

SWITCH-2# set vlans V100 vlan-id 100
SWITCH-2# set vlans V200 vlan-id 200
SWITCH-2# set vlans V300 vlan-id 300
SWITCH-2# set vlans V400 vlan-id 400
SWITCH-2# commit

SWITCH-3# set vlans V300 vlan-id 300
SWITCH-3# set vlans V400 vlan-id 400
SWITCH-3# commit

SWITCH-4# set vlans V100 vlan-id 100
SWITCH-4# set vlans V200 vlan-id 200
SWITCH-4# set vlans V300 vlan-id 300
SWITCH-4# set vlans V400 vlan-id 400
SWITCH-4# commit
```

Once the VLANs are in place you need to create the trunk interfaces and add the VLANs.

ELS ALERT The new ELS command for creating trunk interfaces is `interface-mode` – make sure you're not using the old `port-mode` commands here.

```
SWITCH-1# set interfaces ge-0/0/0.0 family ethernet-switching interface-mode trunk vlan members [V100
V200]
SWITCH-1# set interfaces ge-0/0/1.0 family ethernet-switching interface-mode trunk vlan members [V100
V200]

SWITCH-2# set interfaces ge-0/0/0.0 family ethernet-switching interface-mode trunk vlan members [V300
V400]
SWITCH-2# set interfaces ge-0/0/2.0 family ethernet-switching interface-mode trunk vlan members [V100
V200]
SWITCH-2# set interfaces ge-0/0/11.0 family ethernet-switching interface-mode trunk vlan members [V100
V200 V300 V400]

SWITCH-3# set interfaces ge-0/0/0.0 family ethernet-switching interface-mode trunk vlan members [V300
V400]
SWITCH-3# set interfaces ge-0/0/1.0 family ethernet-switching interface-mode trunk vlan members [V300
V400]

SWITCH-4# set interfaces ge-0/0/0.0 family ethernet-switching interface-mode trunk vlan members [V100
V200]
SWITCH-4# set interfaces ge-0/0/2.0 family ethernet-switching interface-mode trunk vlan members [V300
```

```
V400]
SWITCH-4# set interfaces ge-0/0/11.0 family ethernet-switching interface-mode trunk vlan members [V100
V200 V300 V400]
```

Let's start configuring MSTP on each of the switches. You cannot configure RSTP and MSTP together, only VSTP and RSTP can run simultaneously. Therefore, you need to delete RSTP before committing the MSTP configuration:

```
SWITCH-1# show protocols mstp | display set
set protocols mstp configuration-name region1
set protocols mstp bridge-priority 4k
set protocols mstp interface ge-0/0/0 cost 1000
set protocols mstp interface ge-0/0/0 mode point-to-point
set protocols mstp interface ge-0/0/1 cost 1000
set protocols mstp interface ge-0/0/1 mode point-to-point
set protocols mstp msti 1 bridge-priority 8k
set protocols mstp msti 1 vlan 100
set protocols mstp msti 1 vlan 200

SWITCH-2# show protocols mstp | display set
set protocols mstp configuration-name region1
set protocols mstp interface ge-0/0/0 cost 1000
set protocols mstp interface ge-0/0/0 mode point-to-point
set protocols mstp interface ge-0/0/2 cost 1000
set protocols mstp interface ge-0/0/2 mode point-to-point
set protocols mstp interface ge-0/0/11 cost 1000
set protocols mstp interface ge-0/0/11 mode point-to-point
set protocols mstp msti 1 bridge-priority 8k
set protocols mstp msti 1 vlan 100
set protocols mstp msti 1 vlan 200
set protocols mstp msti 2 bridge-priority 32k
set protocols mstp msti 2 vlan 300
set protocols mstp msti 2 vlan 400

SWITCH-3# show protocols mstp | display set
set protocols mstp configuration-name region1
set protocols mstp interface ge-0/0/0 cost 1000
set protocols mstp interface ge-0/0/0 mode point-to-point
set protocols mstp interface ge-0/0/1 cost 1000
set protocols mstp interface ge-0/0/1 mode point-to-point
set protocols mstp msti 2 bridge-priority 4k
set protocols mstp msti 2 vlan 300
set protocols mstp msti 2 vlan 400

SWITCH-4# show protocols mstp | display set
set protocols mstp configuration-name region1
set protocols mstp interface ge-0/0/0 cost 1000
set protocols mstp interface ge-0/0/0 mode point-to-point
set protocols mstp interface ge-0/0/2 cost 1000
set protocols mstp interface ge-0/0/2 mode point-to-point
set protocols mstp interface ge-0/0/11 cost 1000
set protocols mstp interface ge-0/0/11 mode point-to-point
set protocols mstp msti 1 bridge-priority 32k
set protocols mstp msti 1 vlan 100
set protocols mstp msti 1 vlan 200
set protocols mstp msti 2 bridge-priority 32k
set protocols mstp msti 2 vlan 300
set protocols mstp msti 2 vlan 400
```

Once all the configuration is in place you would expect to see instance 0 (the CIST) across all of the switches. Then you would expect Switch-1 to only have instance 1, Switch-3 should only have instance 2, and Switch-2 and Switch-4 should have instance 0, 1, and 2. Let's take a look at what's on spanning tree now:

```
SWITCH-1# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  4096.5c4527e79b01     1000 FWD   DESG
ge-0/0/1.0             128:514   128:514  4096.5c4527e79b01     1000 FWD   DESG
Spanning tree interface parameters for instance 1
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  8193.5c4527e79b01    20000 FWD   DESG
ge-0/0/1.0             128:514   128:514  8193.5c4527e79b01    20000 FWD   DESG

SWITCH-2# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.5c4527e7e801     1000 FWD   DESG
ge-0/0/2.0             128:515   128:514  4096.5c4527e79b01     1000 FWD   ROOT
ge-0/0/11.0            128:524   128:524 32768.5c4527e7e801     1000 FWD   DESG
Spanning tree interface parameters for instance 1
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/2.0             128:515   128:514  8193.5c4527e79b01    20000 FWD   ROOT
ge-0/0/11.0            128:524   128:524  8193.5c4527e7e801    20000 FWD   DESG
Spanning tree interface parameters for instance 2
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32770.5c4527e7e801    20000 FWD   DESG
ge-0/0/11.0            128:524   128:524 32770.5c4527e7e801    20000 FWD   DESG

SWITCH-3# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.5c4527e7e801     1000 FWD   ROOT
ge-0/0/1.0             128:514   128:515 32768.cce17f8f7ac1     1000 BLK   ALT
Spanning tree interface parameters for instance 2
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  8194.307c5e1086c1    20000 FWD   MSTR
ge-0/0/1.0             128:514   128:514  8194.307c5e1086c1    20000 BLK   ALT

SWITCH-4# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0             128:513   128:513  4096.5c4527e79b01     1000 FWD   ROOT
ge-0/0/2.0             128:515   128:515 32768.cce17f8f7ac1     1000 FWD   DESG
ge-0/0/11.0            128:524   128:524 32768.5c4527e7e801     1000 BLK   ALT
Spanning tree interface parameters for instance 1
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
```

```
ge-0/0/0.0             128:513   128:513  8193.5c4527e79b01    20000 FWD   ROOT
ge-0/0/11.0            128:524   128:524  8193.5c4527e7e801    20000 BLK   ALT
Spanning tree interface parameters for instance 2
Interface            Port ID   Designated    Designated       Port   State Role
                                port ID      bridge ID        Cost
ge-0/0/2.0             128:515   128:515 32770.cce17f8f7ac1    20000 FWD   DESG
ge-0/0/11.0            128:524   128:524 32770.5c4527e7e801    20000 FWD   ROOT
```

NOTE You should have noticed the new role that is unique to MSTP. On the root bridge for each instance, the role of MSTR is present. This indicates the best possible path to reach the CIST from any MSTP region.

On each device, you can get the MSTP configuration and see what VLANs are being carried by each instance:

```
SWITCH-1> show spanning-tree mstp configuration
MSTP information
Context identifier     : 0
Region name             : region1
Revision                : 0
Configuration digest   : 0x669535f92c0ca50b7f33e516f39fe180

MSTI     Member VLANs
   0 0-99,101-199,201-4094
   1 100,200

SWITCH-2> show spanning-tree mstp configuration
MSTP information
Context identifier     : 0
Region name             : region1
Revision                : 0
Configuration digest   : 0x8b5d98ca042bad0d7fa5f18744f4755d

MSTI     Member VLANs
   0 0-99,101-199,201-299,301-399,401-4094
   1 100,200
   2 300,400
We can also get a lot of information from the spanning-tree bridge output.

SWITCH-3> show spanning-tree bridge
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : MSTP
STP bridge parameters for CIST
  Root ID                           : 4096.5c:45:27:e7:9b:01
  Root cost                         : 2000
  Root port                         : ge-0/0/0.0
  CIST regional root                : 32768.30:7c:5e:10:86:c1
  CIST internal root cost           : 0
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Hop count                         : 20
  Message age                       : 2
  Number of topology changes        : 3
  Time since last topology change   : 36035 seconds
  Topology change initiator         : ge-0/0/0.0
```

```
  Topology change last recvd. from  : cc:e1:7f:8f:7a:c5
  Local parameters
    Bridge ID                       : 32768.30:7c:5e:10:86:c1
    Extended system ID              : 0
    Internal instance ID            : 0
STP bridge parameters for MSTI 2
  MSTI regional root                : 4098.30:7c:5e:10:86:c1
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Number of topology changes        : 3
  Time since last topology change   : 36035 seconds
  Topology change initiator         : ge-0/0/0.0
  Local parameters
    Bridge ID                       : 4098.30:7c:5e:10:86:c1
    Extended system ID              : 0
    Internal instance ID            : 1

SWITCH-2> show spanning-tree bridge
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : MSTP
STP bridge parameters for CIST
  Root ID                           : 4096.5c:45:27:e7:9b:01
  Root cost                         : 1000
  Root port                         : ge-0/0/2.0
  CIST regional root                : 32768.5c:45:27:e7:e8:01
  CIST internal root cost           : 0
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Hop count                         : 20
  Message age                       : 1
  Number of topology changes        : 4
  Time since last topology change   : 37211 seconds
  Topology change initiator         : ge-0/0/2.0
  Topology change last recvd. from  : 5c:45:27:e7:9b:04
  Local parameters
    Bridge ID                       : 32768.5c:45:27:e7:e8:01
    Extended system ID              : 0
    Internal instance ID            : 0
STP bridge parameters for MSTI 1
  MSTI regional root                : 8193.5c:45:27:e7:e8:01
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Number of topology changes        : 3
  Time since last topology change   : 37211 seconds
  Topology change initiator         : ge-0/0/2.0
  Topology change last recvd. from  : cc:e1:7f:8f:7a:ce
  Local parameters
    Bridge ID                       : 8193.5c:45:27:e7:e8:01
    Extended system ID              : 0
    Internal instance ID            : 1
STP bridge parameters for MSTI 2
  MSTI regional root                : 32770.5c:45:27:e7:e8:01
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
```

```
Forward delay                     : 15 seconds
Number of topology changes        : 4
Time since last topology change   : 37211 seconds
Topology change initiator         : ge-0/0/0.0
Topology change last recvd. from  : cc:e1:7f:8f:7a:ce
Local parameters
  Bridge ID                       : 32770.5c:45:27:e7:e8:01
  Extended system ID              : 0
  Internal instance ID            : 2
```

You can see here that three instances are running on our small network. Switch-1 is the CIST, and Switch-2 and Switch-3 are the root bridges for their individual instances. You can see that the loop prevention is active and that there is now an MSTP-provisioned network.

While there is a little more configuration with MSTP than standard RSTP, MSTP gives you the flexibility to load balance VLANs across the network and use those paths that may otherwise have gone unused in a standard RSTP-defined network. If you need even more granularity on a per-VLAN basis, then next up is Juniper's VLAN Spanning Tree Protocol (VSTP).

Configuring and Monitoring VLAN Spanning Tree Protocol (VSTP)

VLAN Spanning Tree Protocol (VSTP) is a Juniper Networks proprietary spanning tree protocol that was developed for interoperability with Cisco's Per VLAN Spanning Tree (PVST+) protocol.

We saw from our monitor traffic output earlier that the default spanning tree multicast address is 01:80:c2:00:00:00. Cisco's PVST+ utilizes the standard multicast address but only on VLAN 1. All other VLANs will use the Cisco reserved multicast address of 01:00:0C:CC:CC:CD to carry spanning tree information on a per VLAN basis. Another practice in PVST+ is to use old port cost methods that are not compliant to IEEE standards. To bring Cisco into compliance the `spanning-tree pathcost method long` command has to be entered on all switches.

NOTE For more information on interoperability requirements for Juniper to communicate with Cisco PVST+ please see Juniper's KB: https://kb.juniper.net/InfoCenter/index?page=content&id=KB15138.

You need to be aware that VSTP has a limitation of 253 VLANs with VSTP. You can, however, run RSTP in its default mode to handle all VLANs above the 253 limit. VSTP and RSTP can run simultaneously on all EX Series Switches.

You can configure VSTP at different levels within the configuration:

- All interfaces on a switch
- All interfaces for all VLANs

- All interfaces within a specific VLAN
- All interfaces within a specific VLAN group
- Specific switch interface
- Specific interfaces within a VLAN
- Specific interfaces within a VLAN group

Let's use our current deployed VLANs from the previous MSTP lab (V100/200/300/400) and deploy VSTP across the network.

- Switch-2 will be our root bridge for VLANs V200 and V400.
- Switch-4 will be the backup with a bridge priority of 8k.
- Switch-3 will be the root bridge for V300
- Switch-1 will be the root bridge for V100

```
SWITCH-1# show protocols vstp | display set
set protocols vstp vlan 100 bridge-priority 4k
set protocols vstp vlan 100 interface ge-0/0/0 mode point-to-point
set protocols vstp vlan 200 interface ge-0/0/1 mode point-to-point

SWITCH-2# show protocols vstp | display set
set protocols vstp vlan 100
set protocols vstp vlan 200 bridge-priority 4k
set protocols vstp vlan 400 bridge-priority 4k
set protocols vstp vlan 100 interface ge-0/0/0 mode point-to-point
set protocols vstp vlan 200 interface ge-0/0/2 mode point-to-point
set protocols vstp vlan 400 interface ge-0/0/11 mode point-to-point


SWITCH-3# show protocols vstp | display set
set protocols vstp vlan 300 bridge-priority 4k
set protocols vstp vlan 300 interface ge-0/0/0 mode point-to-point
set protocols vstp vlan 300 interface ge-0/0/1 mode point-to-point

SWITCH-4# show protocols vstp | display set
set protocols vstp vlan 100 interface ge-0/0/0 mode point-to-point
set protocols vstp vlan 200 interface ge-0/0/11 mode point-to-point
set protocols vstp vlan 300 interface ge-0/0/2 mode point-to-point
set protocols vstp vlan 400 interface ge-0/0/11 mode point-to-point
set protocols vstp vlan 100 bridge-priority 32k
set protocols vstp vlan 200 bridge-priority 32k
set protocols vstp vlan 300 bridge-priority 32k
set protocols vstp vlan 400 bridge-priority 32k
```

Now that we have all the configurations in place let's look at the spanning tree topology.

```
SWITCH-1# run show spanning-tree interface
Spanning tree interface parameters for VLAN 100
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID       bridge ID        Cost
```

```
ge-0/0/0.0              128:513   128:513  4196.5c4527e79b01    20000 FWD   DESG
ge-0/0/1.0              128:514   128:514  4196.5c4527e79b01    20000 FWD   DESG
Spanning tree interface parameters for VLAN 200
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0              128:513   128:513 32968.5c4527e79b01    20000 FWD   DESG
ge-0/0/1.0              128:514   128:515  4296.5c4527e7e801    20000 FWD   ROOT

SWITCH-2# run show spanning-tree interface
Spanning tree interface parameters for VLAN 100
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/2.0              128:515   128:514  4196.5c4527e79b01    20000 FWD   ROOT
ge-0/0/11.0             128:524   128:524 32868.5c4527e7e801    20000 FWD   DESG
Spanning tree interface parameters for VLAN 200
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/2.0              128:515   128:515  4296.5c4527e7e801    20000 FWD   DESG
ge-0/0/11.0             128:524   128:524  4296.5c4527e7e801    20000 FWD   DESG
Spanning tree interface parameters for VLAN 300
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0              128:513   128:513  4396.307c5e1086c1    20000 FWD   ROOT
ge-0/0/11.0             128:524   128:524 33068.5c4527e7e801    20000 FWD   DESG
Spanning tree interface parameters for VLAN 400
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0              128:513   128:513  4496.5c4527e7e801    20000 FWD   DESG
ge-0/0/11.0             128:524   128:524  4496.5c4527e7e801    20000 FWD   DESG

SWITCH-3# run show spanning-tree interface
Spanning tree interface parameters for VLAN 300
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0              128:513   128:513  4396.307c5e1086c1    20000 FWD   DESG
ge-0/0/1.0              128:514   128:514  4396.307c5e1086c1    20000 FWD   DESG

SWITCH-4# run show spanning-tree interface
Spanning tree interface parameters for VLAN 100
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0              128:513   128:513  4196.5c4527e79b01    20000 FWD   ROOT
ge-0/0/11.0             128:524   128:524 32868.5c4527e7e801    20000 BLK   ALT
Spanning tree interface parameters for VLAN 200
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/0.0              128:513   128:513 32968.5c4527e79b01    20000 BLK   ALT
ge-0/0/11.0             128:524   128:524  4296.5c4527e7e801    20000 FWD   ROOT
Spanning tree interface parameters for VLAN 300
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/2.0              128:515   128:514  4396.307c5e1086c1    20000 FWD   ROOT
ge-0/0/11.0             128:524   128:524 33068.5c4527e7e801    20000 BLK   ALT
Spanning tree interface parameters for VLAN 400
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID      bridge ID        Cost
ge-0/0/2.0              128:515   128:513  4496.5c4527e7e801    20000 FWD   ROOT
ge-0/0/11.0             128:524   128:524  4496.5c4527e7e801    20000 BLK   ALT
```

As you can see, you are able to run VSTP on specific interfaces or VLANs and you can set the priorities to create numerous spanning tree topologies across the network to influence traffic. This allows you to actually engineer and load balance traffic as necessary.

VSTP is very easy to configure but it has the limitation of 253 VLANs. Those limitations can be overcome by using RSTP configured simultaneously and that will handle any VLANs over 253 that are configured. The nice thing about VSTP is that it is compatible with PVST+ and can be used in mixed-vendor networks or it can stand on its own.

Configuring Loop Protection

Loop protection stops interfaces from going into a forwarding state that would cause loops and be detrimental to the network. Looking back at our network where we had two ALT/BLK interfaces, you could run into a situation where a missed BPDU transitions one or both of those ALT interfaces to start forwarding traffic.

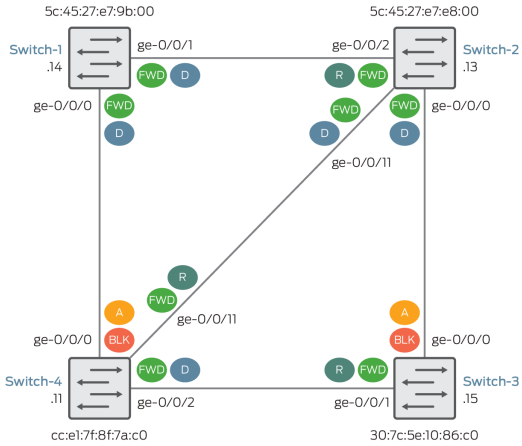


Figure 8.8 *Spanning Tree Topology with Cost.*

Loops can occur when a blocked interface transitions to forwarding because of an interface error. If Switch-4 does not receive a BPDU from Switch-2's designated port in this topology it would transition those BLK interfaces to FWD and cause a loop.

With loop protection, you can prevent that interface from transitioning to a forwarding state and instead transition to a loop-inconsistent state. Once the BPDU is

received from the designated port on Switch-2 the interfaces transition back to BLK state.

Using our topology, you can place loop protection on all of the interfaces but we are primarily concerned with the ALT/BLK interfaces, so let's only perform the commands on Switch-4 for this example:

```
SWITCH-4# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID       bridge ID        Cost
ge-0/0/0.0             128:513   128:513  4096.5c4527e79b01     1000 BLK   ALT
ge-0/0/2.0             128:515   128:515 32768.cce17f8f7ac1    20000 FWD   DESG
ge-0/0/11.0            128:524   128:524  8192.5c4527e7e801      100 FWD   ROOT
```

Loop protection is configured using `bpdu-timeout-action`. There are two actions associated with this key word, you can `block` or `log` and they should be self-explanatory.

```
SWITCH-4# set protocols rstp interface ge-0/0/0 bpdu-timeout-action block
SWITCH-4# commit
```

And that's all there is to it. Now let's disable RSTP on interface ge-0/0/0 on Switch-1 and see what happens on the Switch-4 ge-0/0/0 interface:

```
SWITCH-1# set protocols rstp interface ge-0/0/0 disable
SWITCH-1# commit
configuration check succeeds
commit complete

SWITCH-4> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID       bridge ID        Cost
ge-0/0/0.0             128:513   128:513 32768.cce17f8f7ac1    21000 BLK   DIS (Loop-Incon)
ge-0/0/11.0            128:524   128:524  4096.5c4527e7e801    20000 FWD   ROOT
```

As you can see as soon as the BPDUs were stopped on that interface the port went into BLK DIS (Loop-Incon) state. This indicates that the interface knows it should be receiving BPDUs and refuses to transition to a FWD state (which would cause a spanning tree loop). It will remain in this state until a BPDU is received on that interface and then it will transition back to the ALT state immediately.

Configuring Bridge Protocol Data Unit (BPDU) Protection

BPDU Protection or `bpdu-block` allows you to protect your network by shutting down interfaces if inconsistent BPDUs are received on an interface.

ELS ALERT The `bpdu-block` command has moved from the `ethernet-switching` hierarchy to the `layer2-control` hierarchy in ELS.

Let's test the drop action of `bpdu-block` on ge-1/0/0 on the lab QFX5100:

```
QFX5100-2_3-VC# set protocols layer2-control bpdu-block interface ge-1/0/0 drop
QFX5100-2_3-VC# commit
QFX5100-2_3-VC# run monitor start messages
```

And now turn up the interface on EX2300_16 after giving that switch a 4k bridge priority. Let's take a look at the ethernet switching table. It is indeed discarding:

```
QFX5100-2_3-VC# run show ethernet-switching interface
Routing Instance Name : default-switch
Logical Interface flags (DL - disable learning, AD - packet action drop,
                         LH - MAC limit hit, DN - interface down,
                         SCTL - shutdown by Storm-control,
                         MMAS - Mac-move action shutdown, AS - Autostate-exclude enabled)
Logical          Vlan          TAG     MAC         STP         Logical           Tagging
interface        members                limit       state       interface flags
ge-0/0/4.0                              294912                                     untagged
                 V1000         1000    294912      Forwarding                     untagged
Routing Instance Name : default-switch
Logical Interface flags (DL - disable learning, AD - packet action drop,
                         LH - MAC limit hit, DN - interface down,
                         SCTL - shutdown by Storm-control,
                         MMAS - Mac-move action shutdown, AS - Autostate-exclude enabled)
Logical          Vlan          TAG     MAC         STP         Logical           Tagging
interface        members                limit       state       interface flags
ge-1/0/0.0                              294912                   DN                tagged
                 V100          100     294912      Discarding                     tagged
```

BPDU block at layer2-control is not compatible with RSTP protocols so you have to decide to use one or the other. For more information on using `bpdu-block` please see: https://www.juniper.net/documentation/en_US/junos/topics/concept/spanning-trees-bpdu-protection-understanding-ex-series.html.

Configuring Root Protection

Root protection, or root bridge protection, allows the administrator to enforce root bridge placement in a spanning tree network. This is a common problem especially when older gear, VoIP phones, servers, and anything else that sends a BPDU are attached to a network without root protection.

NOTE The consequences of not protecting your edge ports, not using loop protection, and not using BPDU protection can be detrimental to a Layer 2 network. These are all important and simple configurations to add to your spanning tree protocol.

Based on our four-node topology, you know that through the standard election process, that Switch-3 will become root-based on its MAC address. For this example, let's disable (shutdown) the port between Switch-4 and Switch-3, as well as the port between Switch-2 and Switch-3, to isolate Switch-3 for the time being.

Our topology now looks like Figure 8.9.

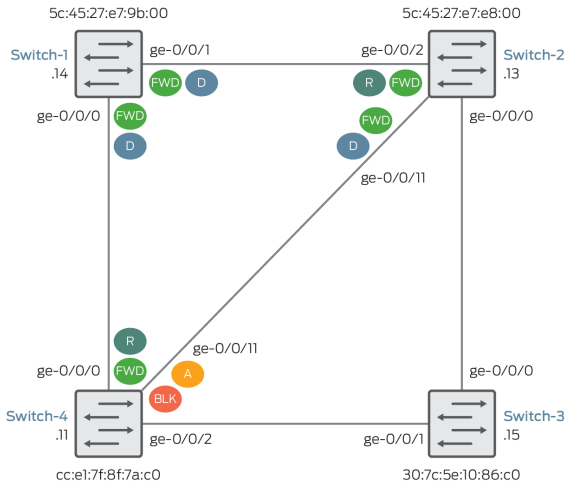


Figure 8.9 *Spanning Tree Topology Excluding Switch-3.*

Without root protection, any device sending BPDUs to the network could potentially take over as root-based solely on the lower MAC address. Imagine someone plugging in a Cisco VoIP phone to this network what would happen? The OUI for Cisco phones is 00-03-68. Take the lessons learned about MAC addresses and let that sink in for a minute.

Okay, so as soon as you enable the ge-0/0/0.0 interface on Switch-2 what is going to happen to the topology? It will change, considerably, right? Because Switch-3 is the lowest MAC address it is going to send BPDUs to the unprotected uplink interface and then it ends up like the following.

Before:

```
SWITCH-2# run show spanning-tree bridge
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : RSTP
  Root ID                           : 32768.5c:45:27:e7:9b:01
  Root cost                         : 20000
  Root port                         : ge-0/0/2.0
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Message age                       : 1
  Number of topology changes        : 25
  Time since last topology change   : 1503 seconds
  Topology change initiator         : ge-0/0/2.0
  Topology change last recvd. from  : 5c:45:27:e7:9b:04
  Local parameters
```

```
    Bridge ID                       : 32768.5c:45:27:e7:e8:01
    Extended system ID              : 0
    Internal instance ID            : 0
```

After:

```
SWITCH-2# run show spanning-tree bridge
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : RSTP
  Root ID                           : 32768.30:7c:5e:10:86:c1
  Root cost                         : 20000
  Root port                         : ge-0/0/0.0
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Message age                       : 1
  Number of topology changes        : 27
  Time since last topology change   : 110 seconds
  Topology change initiator         : ge-0/0/11.0
  Topology change last recvd. from  : cc:e1:7f:8f:7a:ce
  Local parameters
    Bridge ID                       : 32768.5c:45:27:e7:e8:01
    Extended system ID              : 0
    Internal instance ID            : 0
```

Just like that, our entire network is directing spanning tree Topology Change Notification (TCN) to Switch-3 which had no business being the root. This could as easily have been a phone or server. Let's reset everything and then add root protection and try again.

After setting interface ge-0/0/0 to disable on Switch-2 you can see that the network thinks Switch-1 is the root bridge:

```
SWITCH-1> show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface           Port ID   Designated    Designated       Port   State Role
                              port ID       bridge ID        Cost
ge-0/0/0.0            128:513   128:513 32768.5c4527e79b01    20000 FWD   DESG
ge-0/0/1.0            128:514   128:514 32768.5c4527e79b01    20000 FWD   DESG

SWITCH-1> show spanning-tree bridge
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : RSTP
  Root ID                           : 32768.5c:45:27:e7:9b:01
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Message age                       : 0
  Number of topology changes        : 36
  Time since last topology change   : 8336 seconds
  Topology change initiator         : ge-0/0/0.0
  Topology change last recvd. from  : cc:e1:7f:8f:7a:c3
  Local parameters
    Bridge ID                       : 32768.5c:45:27:e7:9b:01
    Extended system ID              : 0
    Internal instance ID            : 0
```

This time, though, add root protection to our RSTP configuration on ge-0/0/1 on Switch-2 as well as ge-0/0/2 on Switch-4:

```
SWITCH-2# set protocols rstp interface ge-0/0/1 no-root-port

SWITCH-4# set protocols rstp interface ge-0/0/2 no-root-port
```

And when you enable those ports and check the root bridge:

```
SWITCH-2# delete interfaces ge-0/0/0 disable
SWITCH-2# commit
configuration check succeeds
commit complete

SWITCH-2# run show lldp neighbors
Local Interface    Parent Interface    Chassis Id          Port info          System Name
ge-0/0/0.0         -                   5c:45:27:b1:72:00   ge-0/0/0.0         SWITCH-3
ge-0/0/2.0         -                   5c:45:27:e7:9b:00   ge-0/0/1.0         SWITCH-1
ge-0/0/11.0        -                   cc:e1:7f:8f:7a:c0   ge-0/0/11.0        SWITCH-4

SWITCH-2# run show spanning-tree bridge
STP bridge parameters
Context ID                          : 0
Enabled protocol                    : RSTP
  Root ID                           : 32768.5c:45:27:e7:9b:01
  Root cost                         : 20000
  Root port                         : ge-0/0/2.0
  Hello time                        : 2 seconds
  Maximum age                       : 20 seconds
  Forward delay                     : 15 seconds
  Message age                       : 1
  Number of topology changes        : 28
  Time since last topology change   : 8678 seconds
  Topology change initiator         : ge-0/0/2.0
  Topology change last recvd. from  : 5c:45:27:b1:72:03
  Local parameters
    Bridge ID                       : 32768.5c:45:27:e7:e8:01
    Extended system ID              : 0
    Internal instance ID            : 0

SWITCH-2# run show spanning-tree interface
Spanning tree interface parameters for instance 0
Interface            Port ID   Designated    Designated       Port   State Role
                               port ID       bridge ID        Cost
ge-0/0/0.0             128:514   128:513 16384.5c4527b17201    20000 BLK   ALT (Root-Incon)
ge-0/0/2.0             128:515   128:514 32768.5c4527e79b01    20000 FWD   ROOT
ge-0/0/11.0            128:524   128:524 32768.5c4527e7e801    20000 FWD   DESG
```

You can see that the root bridge is still Switch-1 and life is good. The same with enabling the port on Switch-4. Now you have the full topology back and the root bridge is protected. Using the `no-root-port` is another way to protect the network from rogue devices sending BPDUs that could possibly influence the network's spanning tree topology. You can also see in the spanning tree interface output that the state is showing `Root-Incon`, indicating that it is blocking BPDUs on this interface.

For more information on configuring root protection please refer to the documents in the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/example/spanning-trees-root-protection-ex-series.html.

Configuring Redundant Trunk Group (RTG) Protection

RTG assists network convergence by having a hot standby link ready and waiting. If one of the trunk links go down, then traffic is simply diverted to the redundant trunk and traffic continues to flow.

RTG is configured on the access switch and is made up of a primary and a secondary link.

The key to RTG is that if the primary link fails, the secondary link starts forwarding traffic and bypasses the spanning tree convergence algorithms (they can take time).

Another benefit of RTG is that Layer 2 control information can be passed, such as LLDP, while normal data traffic is discarded.

Let's use our simple four-node lab topology but in this scenario just use Switches 1, 2, and 4, with Switch-2 being our access layer device and Switches 1 and 4 being our aggregation layer. The primary link will be to Switch-1 and the redundant link will be to Switch-4.

The first thing to do is disable RSTP on the RTG links on Switch-2:

```
SWITCH-2# set protocols rstp interface ge-0/0/2 disable
SWITCH-2# set protocols rstp interface ge-0/0/11 disable
SWITCH-2# commit
configuration check succeeds
```

And then disable RSTP on Switch-1:

```
SWITCH-1# set protocols rstp interface ge-0/0/1 disable
SWITCH-1# commit
```

Configuration check succeeds. And on Switch-4:

```
SWITCH-4# set protocols rstp interface ge-0/0/11 disable
SWITCH-4# commit
```

Configuration check succeeds again. And now we can create the RTG on Switch-2.

ELS ALERT The `switch-options` hierarchy is an ELS change to the old `ethernet-switching-options` hierarchy. Just be aware of the ELS change if you are familiar with the NON-ELS, and need to easily switch to the ELS commands.

```
SWITCH-2# show switch-options | display set
set switch-options redundant-trunk-group group RTG-0 preempt-cutover-timer 5
set switch-options redundant-trunk-group group RTG-0 interface ge-0/0/2.0 primary
```

```
set switch-options redundant-trunk-group group RTG-0 interface ge-0/0/11.0
SWITCH-2# commit
configuration check succeeds
commit complete
```

Everything is now set up for the `redundant-trunk-group` configuration and that the switch knows which interface is primary and which interfaces make up the RTG:

```
SWITCH-2# run show redundant-trunk-group
Group      Interface   State       Time of last flap                     Flap
name                                                                      count
RTG-0      ge-0/0/2.0  Up/Pri/Act  2017-08-01 23:56:55 UTC (00:00:59 ago)     2
           ge-0/0/11.0 Up          2017-08-01 23:57:06 UTC (00:00:48 ago)     2
```

Let's ping sourcing Switch-2 irb.100 gateway for V100 to the gateway on Switch-1. Once that is started you can immediately shut down interface ge-0/0/1 on Switch-1.

```
SWITCH-2> ping 172.16.14.1
PING 172.16.14.1 (172.16.14.1): 56 data bytes
64 bytes from 172.16.14.1: icmp_seq=0 ttl=64 time=4.246 ms
64 bytes from 172.16.14.1: icmp_seq=1 ttl=64 time=3.283 ms
64 bytes from 172.16.14.1: icmp_seq=2 ttl=64 time=3.606 ms
64 bytes from 172.16.14.1: icmp_seq=3 ttl=64 time=4.255 ms
64 bytes from 172.16.14.1: icmp_seq=4 ttl=64 time=3.290 ms
64 bytes from 172.16.14.1: icmp_seq=5 ttl=64 time=3.763 ms
64 bytes from 172.16.14.1: icmp_seq=6 ttl=64 time=3.288 ms
64 bytes from 172.16.14.1: icmp_seq=7 ttl=64 time=3.287 ms
64 bytes from 172.16.14.1: icmp_seq=8 ttl=64 time=34.698 ms
64 bytes from 172.16.14.1: icmp_seq=9 ttl=64 time=4.502 ms
64 bytes from 172.16.14.1: icmp_seq=10 ttl=64 time=4.067 ms
64 bytes from 172.16.14.1: icmp_seq=11 ttl=64 time=3.311 ms
64 bytes from 172.16.14.1: icmp_seq=12 ttl=64 time=4.251 ms
64 bytes from 172.16.14.1: icmp_seq=13 ttl=64 time=3.756 ms
64 bytes from 172.16.14.1: icmp_seq=14 ttl=64 time=3.759 ms
64 bytes from 172.16.14.1: icmp_seq=15 ttl=64 time=42.102 ms
64 bytes from 172.16.14.1: icmp_seq=16 ttl=64 time=3.762 ms
^C
--- 172.16.14.1 ping statistics ---
17 packets transmitted, 17 packets received, 0% packet loss
round-trip min/avg/max/stddev = 3.283/7.837/42.102/11.239 ms

SWITCH-2> show redundant-trunk-group
Group      Interface   State       Time of last flap                     Flap
name                                                                      count
RTG-0      ge-0/0/2.0  Dwn/Pri     2017-08-02 01:01:02 UTC (00:00:08 ago)     3
           ge-0/0/11.0 Up/Act      2017-08-01 23:57:06 UTC (01:04:04 ago)     2
```

Would you look at that! We never even took a hit. You can see that at sequence 8 there was a little more delay but the pings kept going and we never dropped a packet! That's pretty impressive for failover and it's a pretty simple configuration.

Summary

This chapter has discussed several features to help maintain and protect topologies in a spanning tree network. Understanding the BPDU mechanism and how spanning tree works across the different protocols to include RSTP, MSTP, and VSTP is a must in any Layer 2 network. These are all working examples of how to configure and protect your network.

The Juniper TechLibrary has lots of pages about every aspect of spanning tree. Try starting here: http://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/ex4300/spanning-tree-protocols.pdf.

Chapter 9

Configuring EX Series Protocols

Configuring LLDP

The Link Layer Discovery Protocol (LLDP), defined as the IEEE 802.1AB standard, allows network devices to advertise their identity and capabilities on the LAN. In particular, this advertised information allows EX Series switches to identify a variety of devices that can interoperate efficiently in a LAN.

LLDP-capable devices, called *agents,* per standard, transmit information in the form of type, length, and value (TLV) messages, called Link Layer Discovery Protocol Data Units (LLD PDUs), to neighboring devices. These messages can include device specific information such as chassis and port identification, and system name and capabilities. The LLD PDU is sent from each agent and is stored on the receiving agent. It must be refreshed periodically to remain valid.

To enable LLDP on EX switches:

```
switch1# set protocols lldp interface all
```

You can also enable LLDP on a per-interface basis by specifying the interface:

```
switch1# set protocols lldp interface ge-0/0/0
```

Configuring LLDP-MED

LLDP-Media Endpoint Discovery (LLDP-MED) is an extension of the LLDP (IEEE 802.1AB) standard that supports interoperability between voice over IP (VoIP) endpoint devices and other networking end devices. LLDP-MED is commonly used for discovering VoIP phones connected to networked devices such as switches.

In addition to the TLV information that is transmitted on the LLDP agents, LLDP-MED includes additional information such as network policy discovery and Power over Ethernet (PoE) management.

The network policy TLV advertises the VLAN information (see *Voice VLAN* section) for which the interface is configured, as well as associated Layer 2 and Layer 3 attributes such as 802.1Q tagging, and QoS information such as DSCP. The switch uses this TLV to ensure that voice traffic gets treated with appropriate priority by advertising this information to the IP phone.

For supported devices, the PoE management TLV lets the switch advertise the power level and PoE priority required. For example, the switch can compare the power required by an IP telephone connected to a PoE interface with available resources. If the switch cannot deliver the resources required by the IP phone, the switch could negotiate with the IP phone until a compromise on power is reached.

And the location information advertises the configured physical location of the endpoint. This can be determined either by physical location or by Emergency Line Identification Number (ELIN).

You can enable LLDP-MED on EX switches using the following configuration:

```
switch1# set protocols lldp-med interface all
```

LLDP-MED can also be enabled on a per-interface basis by using this configuration:

```
switch1# set protocols lldp-med interface ge-0/0/0
```

LLDP and LLDP-MED Interaction

By default, interfaces configured with both LLDP and LLDP-MED will only advertise TLVs defined in LLDP. Once the interface detects an LLDP-MED-capable device by receiving LLDP-MED TLVs, the interface will toggle to send LLDP-MED TLVs out on the interface.

For verifying LLDP status on EX Ethernet switches, use the `show lldp` command:

```
switch1> show lldp
LLDP                      : Enabled
Advertisement interval    : 30 seconds
Transmit delay            : 2 seconds
Hold timer                : 120 seconds
Notification interval     : 0 Second(s)
Config Trap Interval      : 0 seconds
Connection Hold timer     : 300 seconds
LLDP MED                  : Enabled
MED fast start count      : 3 Packets
Port ID TLV subtype       : locally-assigned
Port Description TLV type : interface-alias (ifAlias)
In/terface      Parent Interface    LLDP        LLDP-MED       Power Negotiation
all             -                  Enabled     Enabled        Enabled
```

One of the most useful aspects of LLDP information is the list of neighbors on the database of the EX Ethernet switch. Use the show lldp neighbors command:

```
user@switch> show lldp neighbors
Local Interface    Parent Interface    Chassis Id          Port info          System Name
xe-0/2/1           ae1                 10:0e:7e:b0:66:80   xe-0/0/1           FABRIC-leaf-1
me0                    -                       78:fe:3d:38:02:00   ge-0/0/45.0         switch05
xe-0/2/0           ae1                 dc:38:e1:e2:4c:c0   xe-0/0/0           FABRIC-leaf-0
```

In the event an existing LLDP neighbor list needs to be cleared, you can clear it using the following:

```
user@switch1> clear lldp neighbors
```

Individual interfaces can be specified if it is not desirable to clear the entire database:

```
user@switch1> clear lldp neighbors interface ge-0/0/0
```

And it is also useful to see what information is being advertised to the neighbors, as shown here with the show lldp local-information command:

```
user@switch1> show lldp local-information
LLDP Local Information details
Chassis ID : 12:34:56:78:9a:bc System name : switch System descr : Juniper Networks, Inc. ex4200- 448t
, version 12.3R6.6 Build date: 2014-xx-xx 08:38:30 UTC
System Capabilities
   Supported
Enabled
: Bridge Router
 : Bridge Router
Parent Interface SNMP Index
Interface description
Status
- 36 - 34 - 503
- 505
-
- - -
Up Up Up
Up
Disabled
Disabled
Disabled
Disabled
```

Collected statistics on EX4200 Ethernet switches can be viewed by using the `statistics` keyword:

```
user@switch> show lldp statistics
Interface Parent Interface Received Unknown TLVs With Errors
ge-0/0/0.0   -
ge-0/0/1.0   -
ge-0/0/2.0   -
158502    0             0
158510    0             0
158517    0             0
```

```
Discarded TLVs  Transmitted  Untransmitted
0               158502       1
0               158510       1
0               158517       1
```

Finally, use the `clear` keyword to clear the collected LLDP statistics on the EX4200 switch:

```
user@switch1> clear lldp statistics
```

TIP Individual interfaces can also be specified if necessary (similar to the `clear lldp neighbors interface ge-0/0/0` command).

Configuring LLDP Management Address

In an effort to make LLDP information more useful on ELS supported switches, by default, when LLDP is enabled, LLDP advertises the out of band management interface address of the switch. This is very handy to have if you need to know the neighboring switch's IP address so you can log into it for management or troubleshooting. You can find this information by running the `show lldp neighbors interface` command and appending the associated interface. For example: `show lldp neighbors interface ge-3/0/3`. Here is example output of the command. It's showing the management IP address of 192.168.56.41:

```
> show lldp neighbors interface ge-0/0/0
LLDP Neighbor Information:
Local Information:
Index: 1 Time to live: 120 Time mark: Fri Aug  4 15:34:26 2017 Age: 4 secs
Local Interface    : ge-0/0/0
Parent Interface   : -
Local Port ID      : 515
Ageout Count      : 0
Neighbour Information:
Chassis type       : Mac address
Chassis ID         : 00:05:86:71:e7:c0
Port type          : Locally assigned
Port ID            : 515
Port description   : ge-0/0/0
System name        : VEX1

System Description : Juniper Networks, Inc. ex9214 Ethernet Switch, kernel Junos 14.2-
20150201.0, Build date: 2015-02-01 07:08:41 UTC Copyright (c) 1996-2015 Juniper Networks, Inc.

System capabilities
        Supported: Bridge Router
        Enabled  : Bridge Router
Management address
        Address Type      : IPv4(1)
        Address           : 192.168.56.41
        Interface Number  : 1
        Interface Subtype : ifIndex(2)
        OID               : 1.3.6.1.2.1.31.1.1.1.1.1.
Organization Info
       OUI      : IEEE 802.3 Private (0x00120f)
       … Truncated for brevity
```

```
> show lldp neighbors interface ge-0/0/0
LLDP Neighbor Information:
Local Information:
Index: 1 Time to live: 120 Time mark: Fri Aug  4 15:34:26 2017 Age: 4 secs
Local Interface    : ge-0/0/0
Parent Interface   : -
Local Port ID      : 515
Ageout Count      : 0
Neighbour Information:
Chassis type       : Mac address
Chassis ID         : 00:05:86:71:e7:c0
Port type          : Locally assigned
Port ID            : 515
Port description   : ge-0/0/0
System name        : VEX1

System Description : Juniper Networks, Inc. ex9214 Ethernet Switch, kernel Junos 14.2-
20150201.0, Build date: 2015-02-01 07:08:41 UTC Copyright (c) 1996-2015 Juniper Networks, Inc.

System capabilities
        Supported: Bridge Router
        Enabled  : Bridge Router
Management address
        Address Type      : IPv4(1)
        Address           : 192.168.56.41
        Interface Number  : 1
        Interface Subtype : ifIndex(2)
        OID               : 1.3.6.1.2.1.31.1.1.1.1.1.
Organization Info
       OUI      : IEEE 802.3 Private (0x00120f)
       Subtype  : MAC/PHY Configuration/Status (1)
       Info     : Autonegotiation [supported, enabled (0x3)], PMD Autonegotiation Capability
(0x1d6c), MAU Type (0x0)
     Index    : 1
Organization Info
       OUI      : IEEE 802.3 Private (0x00120f)
       Subtype  : Link Aggregation (3)
       Info     : Aggregation Status (0x1), Aggregation Port ID (0)
       Index    : 2

Organization Info
       OUI      : IEEE 802.3 Private (0x00120f)
       Subtype  : Maximum Frame Size (4)
       Info     : MTU Size (1514)
       Index    : 3
```

But in non-ELS switches, by default, LLDP does NOT advertise the management interface information. In addition, if you are using in band management instead of out of band management, the default action of advertising the out of band management IP address may not be useful. In the next example, which is the output of a neighboring, non-ELS switch, an EX2200-C, there's no field showing the management information:

```
> show lldp neighbors interface ge-0/0/0
LLDP Neighbor Information:
Local Information:
```

```
Index: 1 Time to live: 120 Time mark: Fri Aug  4 16:23:16 2017 Age: 10 secs
Local Interface    : ge-0/0/0
Parent Interface   : -
Local Port ID      : 512
Ageout Count      : 0
Neighbour Information:
Chassis type       : Mac address
Chassis ID         : 5c:45:27:b6:3c:40
Port type          : Locally assigned
Port ID            : 568
Port description   : RAD-SRX300-1 Management Connection to ge-0/0/0
System name        : RAD-EX2200-C-1

System Description : Juniper Networks, Inc. ex2200-c-12p-
2g Ethernet Switch, kernel Junos 15.1R1.8, Build date: 2015-05-29 07:28:49 UTC Copyright (c) 1996-
2015 Juniper Networks, Inc.

System capabilities
      Supported: Bridge Router
      Enabled  : Bridge Router
Media endpoint class: Network Connectivity
<<<<Output should be here
Organization Info
     OUI     : IEEE 802.3 Private (0x00120f)
     Subtype  : MAC/PHY Configuration/Status (1)
     Info    : Autonegotiation [supported, enabled (0x3)], PMD Autonegotiation Capability (0x6c11),
MAU Type (0x0)
     Index   : 1
… Truncated for brevity
```

Fortunately, you can change this default behavior and it works on both non-ELS and ELS switches. Use the `set protocols lldp management-address` command, and the management IP address of your choice, such as the loopback address for in band management, and you'll get your desired result. Here on the lab EX-2200-C, we put the loopback in as the management interface and commited the configuration:

```
# set management-address 192.168.3.1
```

Then re-running `show lldp neighbors interface`, you can now see the correct management interface specified:

```
> show lldp neighbors interface ge-0/0/0
LLDP Neighbor Information:
Local Information:
Index: 1 Time to live: 120 Time mark: Fri Aug  4 18:03:12 2017 Age: 14 secs
Local Interface    : ge-0/0/0
Parent Interface   : -
Local Port ID      : 512
Ageout Count      : 0
Neighbour Information:
Chassis type       : Mac address
Chassis ID         : 5c:45:27:b6:3c:40
Port type          : Locally assigned
Port ID            : 568
Port description   : RAD-SRX300-1 Management Connection to ge-0/0/0
System name        : RAD-EX2200-C-1
```

```
System Description : Juniper Networks, Inc. ex2200-c-12p-
2g Ethernet Switch, kernel Junos 15.1R1.8, Build date: 2015-05-29 07:28:49 UTC Copyright (c) 1996-
2015 Juniper Networks, Inc.

System capabilities
        Supported: Bridge Router
        Enabled  : Bridge Router
Management address
        Address Type      : IPv4(1)
        Address           : 192.168.3.1
        Interface Number  : 0
        Interface Subtype : Unknown(1)
        OID               : 1.3.6.1.2.1.31.1.1.1.1.0.
Media endpoint class: Network Connectivity
… Truncated for brevity
```

Routing Protocols (OSPF)

OSPF is an expansive topic and we are only going to be able to scratch the surface on a few commands in this section due to time and space restraints. Please be sure to visit the Juniper Technical Library to get a very good overview on OSPF, syntax, link state packet breakdown, operational output, and troubleshooting. The Technical Library will provide much more information than this book can provide here: https://www.juniper.net/documentation/en_US/junos/topics/concept/ospf-configuration-overview.html.

OSPF is a two-tier hierarchical link-state routing protocol. Each router builds a routing database based on the OSPF link state advertisement (LSA). Similar to other Junos OS-based platforms, routing protocol configuration is performed under the protocols stanza in Junos.

NOTE OSPF requires an Enhanced Feature License (EFL) on most EX platforms. Please refer to the *Understanding Software Licensing for EX Series Switches* page in the Juniper Technical Library: https://www.juniper.net/documentation/en_US/junos/topics/concept/ex-series-software-licenses-overview.html.

This *Day One* book is going to use the following topology, made up of five switches, to discuss OSFP. Switches A and C are in area 0, B is in both areas 0 and 1, D and E are in area 1.

To configure a multi-area OSPF network, the first step is to configure a single-area OSPF network by specifying the area ID and associated interface.

Device A:

```
[edit]
# set protocols ospf area 0.0.0.0 interface ge-0/0/0
# set protocols ospf area 0.0.0.0 interface ge-0/0/1
```

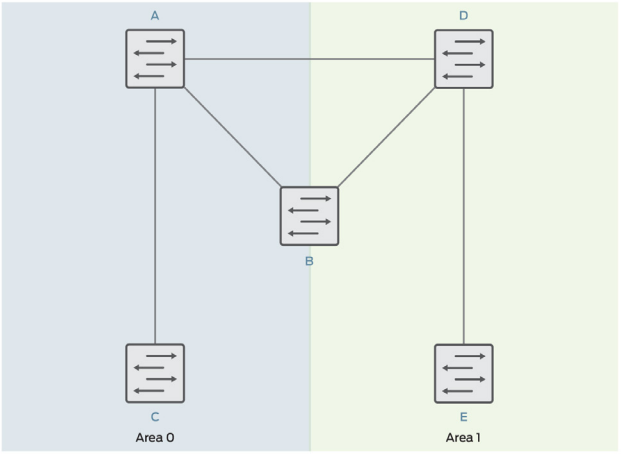


Figure 9.1 *OSPF Area Example.*

Next configure loopback 0 as a passive interface so that you can advertise the interface addresses without actually running OSPF on that interface. The address information is advertised as an internal route:

```
# set protocols ospf area 0.0.0.0 interface lo0 passive
```

MORE? For more information on the passive command please refer to: https://www.juniper.net/documentation/en_US/junos/topics/example/ospf-passive-interface-configuring.html.

Device C:

```
[edit]
# set protocols ospf area 0.0.0.0 interface ge-0/0/0
# set protocols ospf area 0.0.0.0 interface lo0 passive
```

Device B:

```
[edit]
# set protocols ospf area 0.0.0.0 interface ge-0/0/0
# set protocols ospf area 0.0.0.2 interface ge-0/0/2
# set protocols ospf area 0.0.0.0 interface lo0 passive
```

Device D:

```
[edit]
# set protocols ospf area 0.0.0.2 interface ge-0/0/0
# set protocols ospf area 0.0.0.2 interface ge-0/0/2
# set protocols ospf area 0.0.0.2 interface lo0 passive
```

Device E:

```
[edit]
# set protocols ospf area 0.0.0.2 interface ge-0/0/2
```

If you want to configure ospfv3, replace `ospf` with `ospf3`, for example: `set protocols ospf3 area 0.0.0.0 interface <number>`.

The `show ospf neighbor` command provides a good OSPF summary between adjacencies, such as the local interface, the IP address that OSPF is enabled on, the respective adjacency state, and the neighbor's information:

```
user@switch1> show ospf neighbor
Address Interface 172.16.31.2 ge-0/0/23.0 172.16.3.2 irb.10
State     ID
Full      10.0.0.2
Full      10.0.0.3
Pri    Dead
128      32
16
```

Use the `show ospf route` command to view the OSPF routes learned from other OSPF-enabled routers or the `show route` command to view all of the routing tables:

```
user@switch> show ospf route
Topology default Route Table:
Prefix
1.0.0.1
1.0.0.2
172.16.3.2
192.0.0.1
10.0.0.1/32
172.16.3.0/24
172.16.31.0/24
172.16.81.0/24
172.16.82.0/24
192.168.150.0/24
Path  Route
Type  Type
Intra Area/AS
Intra Area/AS
Intra Router
Intra Router
Intra Network
Intra Network
Intra Network
Intra Network
Intra Network
Intra Network
NH       Metric NextHop       Nexthop
Type            Interface     Address/LSP
BR IP           2 ge-0/0/0.0
BR IP           2 ge-0/0/0.0
   IP            1 vlan.1
   IP            1 ge-0/0/0.0
   IP            0 lo0.0
```

```
   IP            1 vlan.1
   IP            1 ge-0/0/23.0
   IP            3 ge-0/0/0.0
   IP            3 ge-0/0/0.0
   IP            1 ge-0/0/0.0
192.168.150.2
192.168.150.2
  172.16.3.2
 192.168.150.2
 192.168.150.2
 192.168.150.2
```

NOTE To get more information on operational output and troubleshooting commands for OSPF please visit the Juniper Technical Library: https://www.juniper.net/documentation/en_US/junos/topics/concept/ospf-configuration-overview.html.

Configuring Import Policy

Now let's talk about how you can configure import policies. Import policies only apply to external routes. Any route that is outside the OSPF autonomous system is an external route.

MORE? Please refer to the Technical Library for more information on import policy in OSPF: https://www.juniper.net/documentation/en_US/junos/topics/concept/policy-routing-policies-overview.html).

OSPF import policy allows you to prevent external routes from being added to the routing tables of OSPF neighbors. The import policy does not impact the OSPF database. This means that the import policy has no impact on the link state advertisements. The filtering is done only on external routes in OSPF.

In this example, we will use following OSPF policy settings:

- *Policy-statement*: Defines the routing policy. You specify the name of the policy and further define the elements of the policy. The policy name must be unique and can contain letters, numbers, hyphens, and be up to 255 characters long.
- *Export*: Applies the export policy you created to be evaluated when network summary LSAs are flooded into an area. In this example, the export policy is named export_static.
- *Import*: Applies the import policy you created to prevent external routes from being added to the routing table. In this example, the import policy is named filter_routes.

We will use Switch A and Switch C in this section.

Switch A Configuration

The Export policy redistributes the static routes from Switch A routing table to switch A OSPF database. Static route is in Switch A OSPF database the route is advertised in an LSA to switch A's OSPF neighbor which is Switch C:

```
[edit]
# set interfaces ge-0/0/0 unit 0 family inet address 20.0.3.1/30
# set protocols ospf export export_static
# set protocols ospf area 0.0.0.0 interface ge-0/0/0
```

Redistribute the static route into OSPF:

```
# set policy-options policy-statement export_static from protocol static
# set policy-options policy-statement export_static then accept
```

Switch C Configuration

```
[edit]
# set interfaces ge-0/0/0 unit 0 family inet address 20.0.3.2/30
```

Switch C has an OSPF import policy configured that matches the static route to the 20.0.16.0/30 network and prevents the static route from being installed in switch C's routing table. Here are the configuration steps to configure the OSPF import policy:

```
# set protocols ospf import filter_routes
# set protocols ospf area 0.0.0.0 interface ge-0/0/0
# set policy-options policy-statement filter_routes from route-filter 20.0.16.0/30 exact
# set policy-options policy-statement filter_routes then reject
```

Configure OSPF Interfaces

There are different types of OSPF interfaces. You have to enable the OSPF protocol on one or more interfaces on each device within the network to activate OSPF. Configuration depends on whether the interface is connected to a broadcast or point-to-point network, a point-to-multipoint network, or a non-broadcast multiaccess (NBMA) network. (For more information on OSPF interfaces please refer to the Juniper TechLibrary at: https://www.juniper.net/documentation/en_US/junos/topics/concept/ospf-interfaces-overview.html.)

- A broadcast interface behaves as if the routing device is connected to a LAN.
- A point-to-point interface provides a connection between a single source and a single destination (there is only one OSPF adjacency).
- A point-to-multipoint interface provides a connection between a single source and multiple destinations.
- An NBMA interface behaves in a similar fashion to a point-to-multipoint interface, but you might configure an NBMA interface to interoperate with other equipment.

Configuring an Interface on a Broadcast or Point-to-Point Network

If the interface that you are configuring OSPF upon supports broadcast mode, or if the interface supports point-to-point mode, you specify the interface by including the IP address or the interface name for OSPFv2, or only the interface name for OSPFv3. First step is to configure the interface:

```
[edit]
# set interfaces ge-0/0/0 unit 0 family inet address 10.0.0.1/16
```

Then assign the interface to an OSPF area:

```
[edit]
# set protocols ospf area 0.0.0.0 interface ge-0/0/0
```

TIP If the interface is an ethernet link between two devices, and is truly a point-to-point connection, specifying the ethernet `interface-type` as `p2p` (point-to-point) eliminates the DR/BDR election process. It also eliminates the Type 3 LSA and will improve OSPF convergence by eliminating the exstart and exchange mechanism.

```
[edit]
# set protocols ospf area 0.0.0.0 interface ge-0/0/0.0 interface-type p2p
```

Configuring an Interface on a Nonbroadcast Multiaccess Network

For NBMA networks you have to use nonbroadcast mode rather than point-to-multipoint mode. Nonbroadcast mode treats the NBMA network as a partially connected LAN, electing designated and backup designated routers. All routing devices must have a direct connection to both the designated and backup designated routers, or unpredictable results occur.

1. First step is to configure an interface:

```
[edit]
# set interfaces ge-0/1/0 unit 0 family inet address 192.168.1.1
```

2. Then Create an OSPF area.

```
[edit]
# set protocols ospf area 0.0.0.0
```

3. Assign the interface to the area. In this example, include the `eligible` keyword to allow the neighbor to be a designated router.

```
[edit ]
# set protocols ospf area 0.0.0.0
# set interface ge-0/0/0 interface-type nbma neighbor 192.0.2.2 eligible
```

4. Configure the poll interval.

```
[edit protocols ospf area 0.0.0.0]
# set interface ge-0/0/0 poll-interval 130
```

Configuring an OSPFv2 Interface on a Point-to-Multipoint Network:

OSPFv2 operates by default in point-to-multipoint mode. OSPFv2 treats the network as a set of point-to-point links. Because there is no auto discovery mechanism, you must configure each neighbor.

1. To configure an OSPFv2 interface on a point-to-multipoint network you have to first Configure the interface.

```
[edit]
# set interfaces ge-0/0/0 unit 0 family inet address 192.168.1.1/24
```

2. Then Create an OSPF area.

```
[edit]
# edit protocols ospf area 0.0.0.0
```

3. Assign the interface to the area and specify the neighbor.

```
[edit protocols ospf area 0.0.0.1]
# set interface ge-0/0/0 neighbor 192.168.1.2
```

NOTE We have only touched the surface of OSPF and all of the capabilities. If you are new to OSPF or new to configuring this protocol on Junos OS devices, please refer to the Juniper TechLibrary documentation that covers this topic in detail. The OSPF Overview along with configuration, troubleshooting, and operational commands is a book in and of itself, and covers more interface configurations, as well as the nuts and bolts of OSPF, than what is presented here. To access the OSPF TechLibrary go to: https://www.juniper.net/documentation/en_US/junos/topics/concept/ospf-configuration-overview.html.

Configuring BFD for OSPF

To detect failures in a network, use the Bidirectional Forwarding Detection (BFD). BFD works with a wide variety of network environments and topologies. A pair of routing devices exchange BFD packets. Hello packets are sent at a specified, regular interval. A neighbor failure is detected when the routing device stops receiving a reply after a specified interval. The BFD failure detection timers have shorter time limits than the OSPF failure detection mechanisms, so they provide faster detection. (Please refer to the Juniper TechLibrary for more detailed information on BFD for OSPF: https://www.juniper.net/documentation/en_US/junos/topics/concept/ospf-bfd-overview.html).

There are three types of BFD sessions listed in Table 9.1, based on the source from which BFD packets are sent to the neighbors.

Table 9.1 BFD Table

Type of BFD Session	Description
Non-distributed BFD	BFD sessions running completely on the Routing Engine.
Distributed BFD	BFD sessions running on the Packet Forwarding Engine.
Inline BFD	BFD sessions running on the FPC hardware.

You can configure the following BFD settings:

- `detection-time threshold`: Threshold for the adaptation of the detection time.
- `full-neighbors-only`: Ability to establish BFD sessions only for OSPF neighbors with full neighbor adjacency.
- `minimum-interval`: Minimum transmit and receive interval for failure detection.
- `minimum-receive-interval`: Minimum receive interval for failure detection.
- `multiplier`: Multiplier for hello packets.
- `transmit-interval minimum-interval`: Minimum transmit interval for failure detection.

To configure BFD:

```
[edit]
# set protocols ospf area 0.0.0.0 interface geg-0/0/1 bfd-liveness-detection minimum-interval 300
# set protocols ospf area 0.0.0.0 interface geg-0/0/1 bfd-liveness-detection multiplier 4
# set protocols ospf area 0.0.0.0 interface geg-0/0/1 bfd-liveness-detection full-neighbors-only
```

And use the `show` command to verify the BFD session:

```
user@host> show bfd session detail
                                                Transmit
Address              State     Interface     Detect Time  Interval  Multiplier
10.9.1.33            Up        ge-0/0ge00/0.0          0.600     0.200      3
  Client OSPF, TX interval 0.200, RX interval 0.200, multiplier 3, Authenticate
  Session up time 3d 00:34:18
  Local diagnostic None, remote diagnostic None
  Remote state Up, version 1
  Replicated
10.9.1.29            Up        ge-4/0/0.0          0.600     0.200      3
  Client ISIS L2, TX interval 0.200, RX interval 0.200, multiplier 3
  Session up time 3d 00:29:12, previous down time 00:00:01
  Local diagnostic NbrSignal, remote diagnostic AdminDown
  Remote state Up, version 1
2 sessions, 2 clients
Cumulative transmit rate 10.0 pps, cumulative receive rate 10.0 pps
```

Multicast and Multicast Routing

Multicast allows delivery of packets from a single source to a specific subset of users or many destination members.

NOTE Multicast is another one of those topics that are very expansive. To get more detailed information than what we can cover in this section please refer to the Juniper TechLibrary on Multicast for EX Switches: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/ex4300/multicast.html.

The EX Series switches support three different Protocol Independent Multicast (PIM) modes (PIM is a family of multicast routing protocols for IP networks):

- *PIM DM (dense mode, flood, and prune):* Multicast join requests are initially flooded to all PIM DM-enabled routers. If there are no downstream members, then the router prunes towards the source.
- *PIM SM (sparse mode, explicit join):* The destination/receiver member must send an explicit "join" request to the rendezvous point (RP) router.
- *PIM SSM (source-specific):* One-to-many model; receiving hosts must join with either Internet Group Management Protocol version 3 (IGMPv3) or Multicast Listener Discovery version 2 (MLDv2).

All multicast routing configuration is done under the `pim` stanza in Junos.

In shared tree, RP is the root of the multicast distribution tree. Initially, the source of the multicast and PIM join requests from the last hop router, first converge at the RP. The RP needs to be reachable by all multicast routers. The following command should be configured on the router designated as the RP:

```
user@switch# set protocols pim rp local address <ip_address>
```

TIP The loopback 0 interface is recommended to be the RP interface.

For all other routers, configure:

```
user@switch# set protocols pim rp static address <ip_address>
```

Any routed interface, including RP interface, that will be routing multicast traffic, needs to be enabled for PIM-SM:

```
user@switch# set protocols pim interface <interface_name> mode sparse
```

The `show pim rps` command is to verify the RP. Its output provides a RP address, how the RP is learned, number of active multicast groups, and the multicast group the RP can forward:

```
user@swtich> show pim rps
```

```
Instance: PIM.master Address family INET RP address Type Holdtime Timeout Groups Group prefixes
10.1.1.1 static 0 None 1 224.0.0.0/4
```

This `show pim neighbors` command is used to validate PIM neighbors:

```
user@switch> show pim neighbors
Instance: PIM.master B = Bidirectional Capable, G = Generation Identifier, H = Hello Option Holdtime, L
= Hello Option LAN Prune Delay, P = Hello Option DR Priority
Interface         IP V Mode       Option     Uptime Neighbor addr
ge-1/0/23.0        4 2           HPLG     02:18:42 10.1.2.2
```

The `show multicast route` command displays the multicast route for a given multicast group, as well as the multicast source and the upstream and downstream multicast path:

```
user@switch> show multicast route
 Family: INET
Group: 224.0.1.39
   Source: 1.1.1.2/32
   Upstream interface: ge-0/1/0.0
   Downstream interface list:
      local ge-1/0/23.0
```

Multicast Switching

By default, a switch treats a multicast packet much like a broadcast packet – it floods to all ports within the VLAN with the exception of the source port. IGMP snooping regulates multicast traffic by monitoring the IGMP transmission between the router and host to build a table, associating the Layer 3 multicast group and the switch port on a per-VLAN basis. The switch knows which port to forward the multicast packet. IGMP snooping is enabled by default.

For hosts that do not support IGMP, the group can be manually configured using:

```
user@switch# set protocols igmp-snooping vlan <vlan_name> interface <interface_ name> static group
<multicast_ip_group_address>
```

The `show igmp-snooping membership` command is to view the IGMP snooping table that was built by the switch. The output provides all the multicast groups on a per-VLAN basis:

```
user@switch> show igmp-snooping membership VLAN: v2
   225.1.1.1     *            199 secs
      Interfaces: xe-0/0/1.0, xe-0/0/2.0, xe-0/0/3.0
```

Bidirectional Forwarding Detection (BFD)

The BFD protocol is used in a wide variety of network environment and topologies to detect failure.

In BFD operation, switches exchange BFD hello packets at a specified interval and detect a neighbor failure if they do not receive a reply after a specified interval. The BFD failure detection timers support shorter time limits than the static route failure detection mechanisms, so they can provide faster detection of failures.

Try to configure lower BFD timer values for fast failure detection. If an adjacency fails or a BFD session flaps more than three times in 15 seconds, the timers can automatically adjust to a higher value.

There are three types of BFD sessions listed in Table 9.1, based on the source from which BFD packets are sent to the neighbors.

BFD for Static Routes

To specify the hold down interval, include the `hold-down-interval` statement in the BFD configuration. You can configure a number in the range from 0 through 255,000 milliseconds. The default is 0. If the BFD session goes down and then comes back up during the hold down interval, the timer is restarted.

To specify the minimum, transmit, and receive intervals for failure detection, include the `minimum-interval` statement in the BFD configuration.

This value represents both the minimum interval after which the local routing device transmits hello packets and the minimum interval after which the routing device expects to receive a reply from the neighbor with which it has established a BFD session.

You can configure a number in the range from 1 through 255,000 milliseconds. Optionally, instead of using this statement, you can configure the minimum transmit and receive intervals separately using the `transmit-interval minimum-interval` and `minimum-receive-interval` statements:

```
user@root# set routing-options static route <destination> bfd-liveness-detection minimum-interval
<value>
user@root# set protocols bfd traceoptions file bfd-trace
user@root# set protocols bfd traceoptions flag all
```

OAM Link-Fault Management (802.3ah)

IEEE 802.3ah is a standards-based feature that encompasses operation, administration, and maintenance (OAM) to help increase reliability and streamline administration and maintenance for Ethernet. The 802.3ah standard is a link-layer and point-to-point protocol, thus, it does not extend beyond the local link. While the 802.3ah standard provides remote failure indication, remote loopback, link monitoring, and discovery, let's focus on how it can be used to detect a unidirectional link, which occurs when a link between two devices is still up, but one device is no longer receiving traffic because of a hardware or software error. The

802.3ah standard needs to be supported and enabled on the interfaces of both devices. Through discovery – OAM protocol data unit (PDU) – the two endpoints will establish adjacencies and learn each other's capabilities. If one end loses adjacency at any time, then the interface can be forced down.

Note that 802.3ah is configured under the `oam` stanza in Junos. The first step is to configure the OAM action profile for loss adjacency – when adjacency is lost, it brings the link down:

```
user@switch# set protocols oam ethernet link-fault-management action-profile <action- profile-name>
event link-adjacency-loss
user@switch# set protocols oam ethernet link-fault-management action-profile <action- profile-name>
action link-down
```

Next, enable 802.3ah on the interfaces:

```
user@switch# set protocols oam ethernet link-fault-management interface ge- 0/1/0.0 link-discovery
active
```

And the last step is to bind the action profile to the interface:

```
user@switch# set interface ge-0/1/0.0 apply-action-profile action-profile-name
```

You can use the `show oam ethernet link-fault-management` command to validate 802.3ah. The output provides information on the neighboring capabilities as well as the action profile that has been invoked. When the output displays a MAC address for the peer address, and the discovery state is `Send Any`, then OAM `link-fault-management` is configured correctly:

```
user@switch> show oam ethernet link-fault-management Interface: ge-0/0/23.0
   Status: Running, Discovery state: Send Any
   Peer address: 00:11:22:33:44:55
   Flags:Remote-Stable Remote-State-Valid Local-Stable 0x50
   Remote entity information:
     Remote MUX action: forwarding, Remote parser action: forwarding
     Discovery mode: active, Unidirectional mode: unsupported
     Remote loopback mode: unsupported, Link events: supported
     Variable requests: unsupported
Application profile statistics:
  Profile Name
  down-link
Invoked    Executed
        0       0
```

Configuring BGP

BGP is an exterior gateway protocol that is used to exchange routing information among switches in different autonomous system. The routing information includes the complete route to each destination.

BGP allows for policy-based routing. You can use routing policies to choose among multiple paths to a destination and to control the redistribution of routing information.

Internal BGP Session

So, first let's talk about how you can configure IBGP on Juniper switches. In this example, there are three switches: A, B, and C, and they are all part of autonomous system 20. We'll use loopback addresses to establish connection between IBGP peers as shown in Figure 9.2.

The loopback interface is always up, that's why it's preferred to use it for IBGP peering. If you use a physical address and that interface goes up and down, the IBGP peer session also goes up and down.

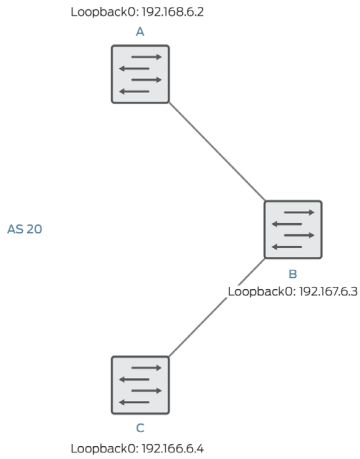


Figure 9.2 Configure IBGP

Here's how to configure IBGP between these three switches.

Switch A:

```
# set interfaces ge-0/1/0 unit 0 description to-B
# set interfaces ge-0/1/0 unit 0 family inet address 20.20.20.1/30
# set interfaces lo0 unit 1 family inet address 192.168.6.2/32
# set protocols bgp group internal-peers type internal
# set protocols bgp group internal-peers description "connections to B and C"
```

The `local-address` statement enables you to specify the source information in BGP update messages. If you don't add the `local-address` statement, the source of BGP update messages is based on the device's source address selection rules. When this happens, the peer session is not established because a mismatch exists between the

expected source address (the egress interface of the peer) and the actual source (the loopback interface of the peer). To make sure that the expected source address matches the actual source address, specify the loopback interface address in the `local-address` statement:

```
# set protocols bgp group internal-peers local-address 192.168.6.2
# set protocols bgp group internal-peers export send-direct
# set protocols bgp group internal-peers neighbor 192.167.6.3
# set protocols bgp group internal-peers neighbor 192.166.6.4
# set protocols ospf area 0.0.0.0 interface lo0.1 passive
# set protocols ospf area 0.0.0.0 interface ge-0/1/0.1
```

This configures a policy that accepts direct routes but it would also will be useful for this scenario to accept routes that are learned from OSPF or local routes.

```
# set policy-options policy-statement send-direct term 2 from protocol direct
# set policy-options policy-statement send-direct term 2 then accept
# set routing-options router-id 192.168.6.2
# set routing-options autonomous-system 20
```

Device B:

```
# set interfaces ge-0/1/0 unit 0 description to-A
# set interfaces ge-0/1/0 unit 0 family inet address 20.20.20.2/30
# set interfaces ge-0/1/1 unit 0 description to-C
# set interfaces ge-0/1/1 unit 0 family inet address 20.20.20.5/30
# set interfaces lo0 unit 2 family inet address 192.167.6.3/32
# set protocols bgp group internal-peers type internal
# set protocols bgp group internal-peers description "connections to A and C"
# set protocols bgp group internal-peers local-address 192.167.6.3
# set protocols bgp group internal-peers export send-direct
# set protocols bgp group internal-peers neighbor 192.168.6.2
# set protocols bgp group internal-peers neighbor 192.166.6.4
# set protocols ospf area 0.0.0.0 interface lo0.2 passive
# set protocols ospf area 0.0.0.0 interface ge-0/1/0.0
# set protocols ospf area 0.0.0.0 interface ge-0/1/1.0
# set policy-options policy-statement send-direct term 2 from protocol direct
# set policy-options policy-statement send-direct term 2 then accept
# set routing-options router-id 192.167.6.3
# set routing-options autonomous-system 20
```

Device C:

```
# set interfaces ge-0/1/0 unit 0 description to-B
# set interfaces ge-0/1/0 unit 0 family inet address 10.10.10.6/30
# set interfaces lo0 unit 3 family inet address 192.166.6.4/32
# set protocols bgp group internal-peers type internal
# set protocols bgp group internal-peers description "connections to A and B"
# set protocols bgp group internal-peers local-address 192.166.6.4
# set protocols bgp group internal-peers export send-direct
# set protocols bgp group internal-peers neighbor 192.168.6.2
# set protocols bgp group internal-peers neighbor 192.167.6.3
# set protocols ospf area 0.0.0.0 interface lo0.3 passive
# set protocols ospf area 0.0.0.0 interface ge-0/1/0.0
# set policy-options policy-statement send-direct term 2 from protocol direct
# set policy-options policy-statement send-direct term 2 then accept
```

```
# set routing-options router-id 192.166.6.4
# set routing-options autonomous-system 20
```

You can use `show interfaces`, `show policy-options`, `show protocol`, and `show routing-option` commands to confirm the configuration:

```
user@C# show interfaces
   ge-0/1/0 {
unit 0 {
description to-B;
family inet {
address 10.10.10.6/30;
}
}
}
lo0 {
unit 3 {
family inet {
address 192.166.6.4/32;
}
}

user@C# show policy-options
policy-statement send-direct {
term 2 {
from protocol direct;
then accept;
}
}
```

You can use `show bgp neighbors` command to make sure peering is up between the switches:

```
user@A> show bgp neighbor
Peer: 192.167.6.3+179 AS 20    Local: 192.168.6.2+58852 AS 20
  Type: Internal    State: Established    Flags: Sync
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Export: [ send-direct ]
  Options: Preference LocalAddress Refresh
  Local Address: 192.168.6.2 Holdtime: 90 Preference: 170
 Number of flaps: 0
  Peer ID: 192.167.6.3    Local ID: 192.168.6.2       Active Holdtime: 90
  Keepalive Interval: 30         Peer index: 0
  BFD: disabled, down
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  NLRI for this session: inet-unicast
  Peer supports Refresh capability (2)
  Restart time configured on the peer: 120
  Stale routes from peer are kept for: 300
  Restart time requested by this peer: 120
  NLRI that peer supports restart for: inet-unicast
  NLRI that restart is negotiated for: inet-unicast
  NLRI of received end-of-rib markers: inet-unicast
  NLRI of all end-of-rib markers sent: inet-unicast
  Peer supports 4 byte AS extension (peer-as 20)
  Peer does not support Addpath
  Table inet.0 Bit: 10000
```

```
  RIB State: BGP restart is complete
  Send state: in sync
  Active prefixes:              0
  Received prefixes:            3
  Accepted prefixes:            3
  Suppressed due to damping:    0
  Advertised prefixes:          2
Last traffic (seconds): Received 25   Sent 19   Checked 67
Input messages:  Total 2420   Updates 4       Refreshes 0     Octets 46055
Output messages: Total 2411   Updates 2       Refreshes 0     Octets 45921
Output Queue[0]: 0
```

Configuring Exterior BGP

In this section, we'll show you how you can configure EBGP sessions between Juniper EX Series switches. In the sample network shown in Figure 9.3, switch A in AS20 has BGP sessions to a group of peers called external-peers. Peer B resides in AS80 and Peer C resides in AS22.

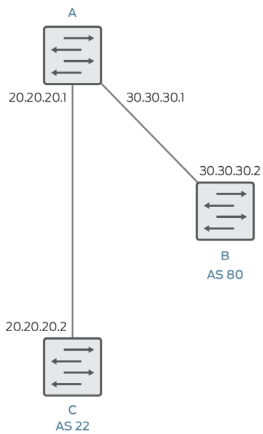


Figure 9.3 *Configure EBGP*

Here is the configuration to establish EBGP sessions between A, B, and C switches.

Switch A:

```
# set interfaces ge-1/2/0 unit 0 description to-B
# set interfaces ge-1/2/0 unit 0 family inet address 30.30.30.1/30
# set interfaces ge-0/0/1 unit 0 description to-C
# set interfaces ge-0/0/1 unit 0 family inet address 20.20.20.1/30
# set protocols bgp group external-peers type external
# set protocols bgp group external-peers neighbor 30.30.30.2 peer-as 80
# set protocols bgp group external-peers neighbor 20.20.20.2 peer-as 22
# set routing-options autonomous-system 20
```

If you want to send direct routes configure below routing policy

```
# set policy-options policy-statement send-direct term 1 from protocol direct
# set policy-options policy-statement send-direct term 1 then accept
```

Switch B:

```
# set interfaces ge-1/2/0 unit 0 description to-A
# set interfaces ge-1/2/0 unit 0 family inet address 30.30.30.2/30
# set protocols bgp group external-peers type external
# set protocols bgp group external-peers neighbor 30.30.30.1 peer-as 20
# set routing-options autonomous-system 80
# set policy-options policy-statement send-direct term 1 from protocol direct
# set policy-options policy-statement send-direct term 1 then accept
```

Switch C:

```
# set interfaces ge-1/2/0 unit 0 description to-A
# set interfaces ge-1/2/0 unit 0 family inet address 20.20.20.2/30
# set protocols bgp group external-peers neighbor 20.20.20.1 peer-as 20
# set routing-options autonomous-system 22
# set policy-options policy-statement send-direct term 1 from protocol direct
# set policy-options policy-statement send-direct term 1 then accept
```

You'll need more information on BGP because of its possibilities, so check out the TechLibrary at: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/config-guide-routing/config-guide-routing-bgp.pdf.

Summary

This section introduced you to a lots of dynamic routing protocols and features that may be new to anyone just picking up this book, and yet we only scratched the surface of it all. Please be sure to take what you have learned here and use the Juniper TechLibrary to expand your knowledge of OSPF, BFD, PIM, and BGP. These are all important to enterprise routing. The detailed information for all of these protocols can be found here: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/ex4300/index.html.

Chapter 10

Configuring EX Series System

Configuring Local Users

One of the first things you will do when you are configuring Junos is to create users. While Junos supports both RADIUS and TACACS+, you will almost always have one local user other than *root*. The root user is built into the box and if you remember from Chapter 3, setting the `root-authentication` was mandatory before the system would take a commit for the first time.

Local user accounts are just that. They are static accounts on Junos that are always accessible. A typical local account may be for security or tier-1 with limited access to specific commands, or tier-2 accounts more access than tier-1 but unable to create additional accounts. You can limit local users access by specifying allow or deny-commands, and allow or deny-configuration. The difference between deny-commands and deny-configuration is that deny-commands are for operational commands and deny-configuration commands are for, you guessed it, configuration commands. You can also limit a user's access by specifying classes with specific access areas.

First let's build a typical local user account, and then build a user named LAB with a password of Juniper123. You can perform a quick `show` command:

```
switch1# show | display set | match user
set system login user junspace uid 2000
set system login user junspace class super-user
set system login user junspace authentication encrypted-password "$5$TYz4tAbe$e8FfaEVoIltX5XmBolayuiS
NZK73wxjs/dfDXKeA6Z0"
```

TIP Notice that we used the pipe command to display the configuration as set-based. The default show command is the ASCII curly brace configuration. You can also see that we used a second pipe to match on user. You could have specified `system login user` but knowing that user is unique within the string is easy to match on. As you progress with your Junos knowledge you will be able to pick out specific strings to match on to find specific information within the configuration.

The normal ASCII output of the `show` command:

```
switch1# show system login
user junspace {
    uid 2000;
    class super-user;
    authentication {
        encrypted-password "$5$TYz4tAbe$e8FfaEVoIltX5XmBolayuiSNZK73wxjs/dfDXKeA6Z0"; ## SECRET-DATA
    }
}
```

Here is the minimum requirement to create a local user: user name, a class, and authentication. You can assign the user id UID manually, which is an extra step, or the system will automatically assign the user id.

Let's go ahead and create our user and give them a limited login class. The super-user class is basically the root user. Using context sensitive help, you can see that there are four built-in classes:

```
switch1# set system login user LAB class ?
Possible completions:
  <class>              Login class
  operator             permissions [ clear network reset trace view ]
  read-only            permissions [ view ]
  super-user           permissions [ all ]
  unauthorized         permissions [ none ]
```

Let's give our LAB user, class read-only, and then see if we can use that account to edit the configuration:

```
switch1# set system login user LAB class read-only authentication plain-text-password
New password:
Retype new password:

{master:0}[edit]
switch1# show | compare
[edit system login]
+    user LAB {
+        class read-only;
+        authentication {
+            encrypted-password "$5$sVprU.Wk$BpsoZzuuf.
sPXQj6gQacJY8hBeFD7WOiS2ZRO3O9hQ8"; ## SECRET-DATA
+        }
+    }

{master:0}[edit]
switch1# commit
```

```
configuration check succeeds
commit complete

switch1# show | display set | match LAB
set system login user LAB uid 2002
set system login user LAB class read-only
set system login user LAB authentication encrypted-password "$5$sVprU.Wk$BpsoZzuuf.
sPXQj6gQacJY8hBeFD7WOiS2ZRO3O9hQ8"
```

So far we've created the user LAB with class read-only and authentication "`plain-text-password`" and then press Enter, so the system will prompt us to enter the password manually on the CLI, and the retype the password for verification. You do not see the password in clear text and you will notice in the `show | compare` command that it now states `encrypted-password`. So as soon as you type it in, it is hashed. Remember, we are still dealing with a candidate configuration, and to make it active, you have to commit. Once the candidate configuration has been committed you can see that the system assigned the user id to the user.

Let's log out and see if the class allows us to do anything but read the configuration.

TIP Since Junos is built on FreeBSD you can guarantee that all commands will be case sensitive. If you tried logging in with *lab* or *Lab*, it would fail because our user is all upper-case: *LAB*.

```
$ ssh LAB@192.168.2.6
Password:
--- Junos 15.1X53-D51 Kernel 32-bit  JNPR-11.0-20160921.337570_build
{master:0}
switch1> edit
              ^
unknown command.

switch1> show configuration
version /* ACCESS-DENIED */;
system { /* ACCESS-DENIED */ };
chassis { /* ACCESS-DENIED */ };
interfaces { /* ACCESS-DENIED */ };
snmp { /* ACCESS-DENIED */ };
forwarding-options { /* ACCESS-DENIED */ };
routing-options { /* ACCESS-DENIED */ };
protocols { /* ACCESS-DENIED */ };
vlans { /* ACCESS-DENIED */ };
poe { /* ACCESS-DENIED */ };

{master:0}
switch1> show route
inet.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
192.168.2.0/24     *[Direct/0] 00:00:00  > via irb.2
192.168.2.6/32     *[Local/0] 00:00:00 Local via irb.2
192.168.5.0/24     *[Static/5] 00:00:00  > to 192.168.2.1 via irb.2
```

```
{master:0}
switch1> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
ge-0/0/0                up    up
ge-0/0/0.0              up    up   eth-switch
          …
{master:0}
switch1> show cli authorization
Current user: 'LAB         ' class 'read-only'
Permissions:
    view        -- Can view current values and statistics
Individual command authorization:
    Allow regular expression: none
    Deny regular expression: none
    Allow configuration regular expression: none
    Deny configuration regular expression: none
```

Well there you have it. The output from the `show cli authorization` command tells us that you can only view current values and statistics. You can see that the LAB user cannot enter edit mode or even read the configuration. They can still see interface information, routes, etc., which would be enough for someone in the NOC, or a new user with little Junos experience.

We're not done with the local user, yet. There are cases where you may want to create a password-less login using Secure Shell (SSH) RSA encrypted key pair. Automation systems sometimes require quick access to the devices in the network. With RSA key pairs, you rely heavily on the security of the server or workstation you are accessing the network devices from. If the server gets compromised, then it's possible your public and private keys could get compromised as well. Therefore, best practice is to have very strict security protocols on any server running with RSA keys that are able to access the network. For lab access, or staging equipment, you can't beat the thriftiness of an RSA key. Let's go ahead and generate an SSH-RSA private and public key pair on the server and then take the public key and place it on the EX2300. Let's go:

```
$ ssh 192.168.2.5 -l LAB
192.168.2.5's password:

$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/LAB/.ssh/id_rsa):
Created directory '/home/LAB/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/LAB/.ssh/id_rsa.
Your public key has been saved in /home/LAB/.ssh/id_rsa.pub.
The key fingerprint is:
8b:9a:99:c9:1c:7f:19:f5:f0:c5:c6:9f:3c:f2:2b:44 AUTOMATE
The key's randomart image is:
```

```
+--[ RSA 2048]----+
|              |
|              |
|           o  |
|         o  E=  |
|       S. +.o...|
|      ...  oo +.|
|   . . .o  . o .|
|  o O  o   . . |
|   O ..     ...|
+-----------------+
[AUTOMATE ~]$
```

You can see that we logged into the LAB users account on the CentOS 7 server. Next, we issued the `ssh-keygen` command with type `-t rsa`. You will be immediately notified that the RSA key pair is being generated. The defaults were all accepted and no passphrase is entered. If you entered a passphrase it would prompt you for it every time you used the key, and are basically back to a password-protected login.

```
[AUTOMATE ~]$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EDAQABAAABAQC4Zll4iKukuS52ds5cXKE4cvC4tdn7ktB5feIXy48s/uscZGOphCIWEJQzp696y/
JWYRpU1GAgImZZPbedC2w6wcqC34YTSf1OpcK81C7xBhmcnAp6DQmjzh5Amim7bo/
bjuKhATNxYhwymaWCBpv6bzUkBQmHcxmGjEAep4zdgbQrte/im1p6od//1AfOEu/
nhOktmeywY7YM4DT1QOXVVL9TFXYB701t1KMUtwjZdL5JLlQk2RjtFkJCIXtvmfkwD1UW3AllJa51+mOERds+/
spiwbQZy0Z7yukxTCKflWec6hI/Bvrd62yQ2pkgdRreGzZtoueiiIXMAYNdcz3P AUTOMATE

switch1> edit
Entering configuration mode

switch1# set system login user LAB authentication ssh-rsa "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC4Zll4iKu kuS52ds5cXKE4cvC4tdn7ktB5feIXy48s/uscZGOphCIWEJQzp696y/
JWYRpU1GAgImZZPbedC2w6wcqC34YTSf1OpcK81C7xBhmcnAp6DQmjzh5Amim7bo/
bjuKhATNxYhwymaWCBpv6bzUkBQmHcxmGjEAep4zdgbQrte/im1p6od//1AfOEu/
nhOktmeywY7YM4DT1QOXVVL9TFXYB701t1KMUtwjZdL5JLlQk2RjtFkJCIXtvmfkwD1UW3AllJa51+mOERds+/
spiwbQZy0Z7yukxTCKflWec6hI/Bvrd62yQ2pkgdRreGzZtoueiiIXMAYNdcz3P AUTOMATE"

[AUTOMATE ~]$ ssh 192.168.2.6
Last login: Sat May  6 03:42:58 2017 from 192.168.2.5
--- Junos 15.1X53-D51 Kernel 32-bit  JNPR-11.0-20160921.337570_build
{master:0}
switch1>
```

Create the RSA key-pair and then display it on the server CLI using `cat`. Then copy the command and pasted it into `set system login user LAB` and you are no longer prompted for a password.

This section introduced you to a few different concepts in creating a local user account setting the class and authentication method. If you want to learn more about denying access to specific parts of the configuration or specific commands, please visit the Juniper TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/concept/access-privileges-allow-deny-regexps-overview.html.

Configuring RADIUS

Most networks have a Remote Authentication Dial-In User Service (RADIUS) or a Terminal Access Control Access-Control System Plus (TACACS+) server to control the Authentication, Authorization and Accounting (AAA) for network access.

NOTE RADIUS uses UDP port 1812 (see RFC 2865), and for authentication and authorization it's UDP port 1813 (see RFC 2866), and for accounting. RADIUS only encrypts the password for the session. TACACS+ uses TCP port 49 for AAA and encrypts the entire payload. For more information on these, see: https://www.juniper.net/documentation/en_US/junos/topics/concept/remote-authentication-servers-understanding.html.

One of the advantages of RADIUS is that it supports IEEE 802.1x and carries Extensible Authentication Protocol (EAP), not from the network device, but from the end-user. IEEE 802.1x will be discussed in-depth later on. For now, you need to know that Junos supports both RADIUS and TACACS+ for AAA.

Another important concept to understand with RADIUS is the use of Vendor Specific Attributes (VSAs). Vendor Specific Attributes start on page 42 of RFC2138, and Table 10.1 lists the most commonly used VSA's for Juniper Networks EX Series.

Table 10.1 RADIUS Vendor Specific Attributes (VSA) Table.

Name	Description	Type	Length	String
Juniper-Local-User-Name	Indicates the name of the user template used by this user when logging in to a device. This attribute is used only in Access-Accept packets.	1	≥3	One or more octets containing printable ASCII characters.
Juniper-Allow-Commands	Contains an extended regular expression that enables the user to run operational mode commands in addition to the commands authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	2	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression. See Regular Expressions for Allowing and Denying Junos OS Operational Mode Commands, Configuration Statements, and Hierarchies.
Juniper-Deny-Commands	Contains an extended regular expression that denies the user permission to run operation mode commands authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	3	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression. See Regular Expressions for Allowing and Denying Junos OS Operational Mode Commands, Configuration Statements, and Hierarchies.

Juniper-Allow-Configuration	Contains an extended regular expression that enables the user to run configuration mode commands in addition to the commands authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	4	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression. See Regular Expressions for Allowing and Denying Junos OS Operational Mode Commands, Configuration Statements, and Hierarchies.
Juniper-Deny-Configuration	Contains an extended regular expression that denies the user permission to run configuration commands authorized by the user's login class permission bits. This attribute is used only in Access-Accept packets.	5	≥3	One or more octets containing printable ASCII characters, in the form of an extended regular expression. See Regular Expressions for Allowing and Denying Junos OS Operational Mode Commands, Configuration Statements, and Hierarchies.
Juniper-Interactive-Command	Indicates the interactive command entered by the user. This attribute is used only in Accounting-Request packets.	8	≥3	One or more octets containing printable ASCII characters.
Juniper-Configuration-Change	Indicates the interactive command that results in a configuration (database) change. This attribute is used only in Accounting-Request packets.	9	≥3	One or more octets containing printable ASCII characters.
Juniper-User-Permissions	Contains information the server uses to specify user permissions. This attribute is used only in Access-Accept packets.	10	≥3	One or more octets containing printable ASCII characters.
	Note: When the Juniper-User-Permissions attribute is configured to grant the Junos OS `maintenance` or `all` permissions on a RADIUS server, the UNIX wheel group membership is not automatically added to a user's list of group memberships. Some operations such as running the `su root` command from a local shell require wheel group membership permissions. However, when a user is configured locally with the permissions `maintenance` or `all`, the user is automatically granted membership to the UNIX wheel group. Therefore, it's recommend that you create a template user account with the required permissions and associate individual user accounts with the template user account.			The string is a list of permission flags separated by a space. The exact name of each flag must be specified in its entirety. See Understanding Junos OS Access Privilege Levels.

Juniper-Authentication-Type	Indicates the authentication method (local database, or RADIUS server) used to authenticate a user. If the user is authenticated using a local database, the attribute value shows 'local'. If the user is authenticated using RADIUS server, the attribute value shows 'remote'.	11	≥5	One or more octets containing printable ASCII characters.
Juniper-Session-Port	Indicates the source port number of the established session.	12	Integer	

Let's put some thought into what is needed to get the remote users to authenticate against a remote server.

1. You need a RADIUS server.
2. You need a shared key between the network devices and the RADIUS server.
3. Users have to be configured on the RADIUS server.
4. A remote profile has to be configured in Junos.
5. You need to test the access using a remote user and the RADIUS server.
6. You need a way to monitor and troubleshoot RADIUS authentication to the Junos device.

This lab uses freeRadius from the CentOS yum repository. The shared key is *Juniper123* and the users are going to be labuser1– 3. Here is a quick look at the users' file under /etc/raddb:

```
labuser1   Cleartext-Password := "labP@$$1"
             Service-Type = Login-User,
             Juniper-Local-User-Name := "remote-su"
labuser2   Cleartext-Password := "labP@$$2"
             Service-Type = Login-User,
             Juniper-Local-User-Name := "remote-su"
labuser3   Cleartext-Password := "labP@$$3"
             Service-Type = Login-User,
             Juniper-Local-User-Name := "remote-su"
```

NOTE The `Juniper-Local-User-Name := "remote-su"` tells RADIUS to use the remote-su configuration on the actual device being logged in to. (You have to have a remote user profile setup for RADIUS.)

The RADIUS configuration looks like this:

```
$ ssh labuser1@192.168.56.11
labuser1@VEX1> show configuration | display set | match radius
set system authentication-order radius
set system radius-server 192.168.56.1 secret "$9$COIylMNdsEcds24DjCtuOEcrevL7-"
set system radius-server 192.168.56.1 retry 3
```

```
set system radius-server 192.168.56.1 source-address 192.168.56.11
set system radius-options password-protocol mschap-v2
set system radius-options attributes nas-ip-address 192.168.56.11
```

As you can see the switch has already been configured to contact the RADIUS server that is currently running at 192.168.56.1. The secret key of *Juniper123* was provided and the source address of our device. Now let's take a look at the `remote-su` profile:

```
# set system login user remote-su uid 2002
# set system login user remote-su class super-user
```

NOTE You don't have to set the User ID (uid) unless you just want to have complete control, otherwise Junos will automatically set the uid per user. Otherwise this is a single command assigning the `remote-su` profile to the class of super user. There is no password on this profile because RADIUS handles the authentication from here.

If you are building a network in your lab and you are the only user, then you can probably justify assigning yourself to the super user class. However, in most organizations, you will have several different departments such as security, network operations center, engineering, operations, and performance, who all want to be in the network to check their specific area. So. you might want to restrict certain people based on their experience level and only give them access to the areas of the network that they really need. Junos has a predefined permission list that will allow you to get very granular with what class of user has permission to see or edit. You can think of this as Role Based Access Control (RBAC) and it is spelled out in Table 10.2.

Table 10.2 Role-Based Access Control (RBAC)

Permission Flag	Description
access	Can view the access configuration in configuration mode and with the show configurationoperational mode command.
access-control	Can view and configure access information at the [edit access] hierarchy level.
admin	Can view user account information in configuration mode and with the show configuration operational mode command.
admin-control	Can view user accounts and configure them at the [edit system login] hierarchy level.
all	Can access all operational mode commands and configuration mode commands. Can modify configuration in all the configuration hierarchy levels.
clear	Can clear (delete) information learned from the network that is stored in various network databases by using the clear commands.
configure	Can enter configuration mode by using the configure command.

control	Can perform all control-level operations—all operations configured with the -control permission flags.
field	Can view field debug commands. Reserved for debugging support.
firewall	Can view the firewall filter configuration in configuration mode.
firewall-control	Can view and configure firewall filter information at the [edit firewall] hierarchy level.
floppy	Can read from and write to the removable media.
flow-tap	Can view the flow-tap configuration in configuration mode.
flow-tap-control	Can view the flow-tap configuration in configuration mode and can configure flow-tap configuration information at the [edit services flow-tap] hierarchy level.
flow-tap-operation	Can make flow-tap requests to the router or switch. For example, a Dynamic Tasking Control Protocol (DTCP) client must have flow-tap-operation permission to authenticate itself to the Junos OS as an administrative user.
	Note: The flow-tap-operation option is not included in the all-control permissions flag.
idp-profiler-operation	Can view profiler data.
interface	Can view the interface configuration in configuration mode and with the show configuration operational mode command.
interface-control	Can view chassis, class of service (CoS), groups, forwarding options, and interfaces configuration information. Can edit configuration at the following hierarchy levels: [edit chassis] [edit class-of-service] [edit groups] [edit forwarding-options] [edit interfaces]
maintenance	Can perform system maintenance, including starting a local shell on the router or switch and becoming the superuser in the shell by using the su root command, and can halt and reboot the router or switch by using the request system commands.
network	Can access the network by using the ping, ssh, telnet, and traceroute commands.
pgcp-session-mirroring	Can view the pgcp session mirroring configuration.
pgcp-session-mirroring-control	Can modify the pgcp session mirroring configuration.
reset	Can restart software processes by using the restart command and can configure whether software processes are enabled or disabled at the [edit system processes] hierarchy level.
rollback	Can use the rollback command to return to a previously committed configuration other than the most recently committed one.
routing	Can view general routing, routing protocol, and routing policy configuration information in configuration and operational modes.

routing-control	Can view general routing, routing protocol, and routing policy configuration information and can configure general routing at the [edit routing-options] hierarchy level, routing protocols at the [edit protocols] hierarchy level, and routing policy at the [edit policy-options] hierarchy level.
secret	Can view passwords and other authentication keys in the configuration.
secret-control	Can view passwords and other authentication keys in the configuration and can modify them in configuration mode.
security	Can view security configuration in configuration mode and with the show configuration operational mode command.
security-control	Can view and configure security information at the [edit security] hierarchy level.
shell	Can start a local shell on the router or switch by using the start shell command.
snmp	Can view Simple Network Management Protocol (SNMP) configuration information in configuration and operational modes.
snmp-control	Can view SNMP configuration information and can modify SNMP configuration at the [edit snmp] hierarchy level.
system	Can view system-level information in configuration and operational modes.
system-control	Can view system-level configuration information and configure it at the [edit system] hierarchy level.
trace	Can view trace file settings and configure trace file properties.
trace-control	Can modify trace file settings and configure trace file properties.
view	Can use various commands to display current system-wide, routing table, and protocol-specific values and statistics. Cannot view the secret configuration.
view-configuration	Can view all of the configuration excluding secrets, system scripts, and event options. Note: Only users with the maintenance permission can view commit script, op script, or event script configuration

It takes just a few steps to create a class with only specific permissions and apply that to a specific user in a RADIUS configuration. Let's say you wanted to create a `VIEW-ONLY` class with read permissions for interface, firewall, network, and SNMP. The configuration would look like this:

```
# set system login class VIEW-ONLY permissions firewall
# set system login class VIEW-ONLY permissions interface
# set system login class VIEW-ONLY permissions network
# set system login class VIEW-ONLY permissions snmp
```

Now let's apply this class to the `remote-su` account instead of the `super-user` class:

```
# set system login user remote-su class VIEW-ONLY
# commit
```

Using our RADIUS login of `labuser1` we should now see a different set of authorized commands than we had previously seen as `super-user`:

```
labuser1@EX2300-16> show cli
                    ^
syntax error, expecting <command>.
labuser1@EX2300-16> show ?
Possible completions:
  firewall             Show firewall information
  host                  Show hostname information from domain name server
  interfaces            Show interface information
  multicast             Show multicast information
  policer               Show interface policer counters and information

labuser1@EX2300-16> edit
                    ^
unknown command.
```

You can see we are unable to issue a `show cli authorization` to view what we are authorized for because it wasn't included in the permissions. You can look at firewall information, interfaces, snmp, policers, and statistics for those areas but you cannot enter edit mode. You should note from Table 10.2, that to actually edit any information you would use the permission keywords with `–control` such as `interface-control`, `routing-control`, or `system-control`.

You can even get more granular with classes and user accounts by denying and allowing only specific commands through the use of Vendor Specific Attributes (VSA's) in RADIUS from Table 10.2 or you can use the same attributes in the Junos configuration:

```
# set system login class VIEW-ONLY permissions all deny-commands "^show interfaces"
[edit]
# set system login class VIEW-ONLY permissions all deny-configuration-
regex [ "interfaces" "protocols" "system"]
```

Once this is committed let's log out and log back in with the LAB account, then do a quick `show cli authorization` output to see that the deny commands and `deny-configuration-regex` are there just as configured:

```
EX2300-16> show cli authorization
Current user: 'LAB       ' class 'VIEW-ONLY'
Permissions:
…
Individual command authorization:
   Allow regular expression: none
   Deny regular expression: ^show interfaces
   Allow configuration regular expression: none
   Deny configuration regular expression: "system" "protocols" "interfaces"
Now let's test to see if our regular expressions will stop our LAB user from actually issuing those
commands.
EX2300-16> show int
                   ^
syntax error, expecting <command>.
 EX2300-16> edit
Entering configuration mode
[edit]
 EX2300-16# set int
                  ^
```

```
syntax error.
 EX2300-16# set protocols
                  ^
syntax error.
[edit]
 EX2300-16# set syste
                  ^
syntax error.
```

And there you have it – even though you can set all permission bits `on` for the `VIEW-ONLY` class you can limit users from issuing specific commands using regular expressions. You can do the same thing for `allow-commands` and `allow-configuration-regex` as well, but let's stop here for now.

MORE? To read more about using the regular expressions for allow and deny configurations please check out: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/security/security-swconfig-initial-device-config.pdf.

We tested the RADIUS configuration, and that RADIUS is working properly, by logging into the device successfully. However, if you were having issues with your RADIUS authentication you could use traceoptions to capture the session, or freeRadius has a command line option to debug using `radiusd -X`, that will provide information on the authentication sessions and point out where things are specifically working and where they are failing. It's a very useful command line tool for the RADIUS administrator.

MORE? Configuring the actual RADIUS server for your particular operating system is beyond the source of this book. Please review the *Juniper Networks User Guide: Configuring RADIUS Server Authentication at URL*: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/radius-authentication-configuring.html. Also, check out the freeRadius WIKI at: http://wiki.freeradius.org for more detail on how to configure the freeRadius server.

Accounting is part of the Authentication Authorization and Accounting (AAA) model and it applies to either RADIUS or TACACS+ configurations. You can log accounting information locally to the Junos device, or you can send the accounting information back to the AAA server.

Here is a quick example of logging accounting information to a RADIUS server:

```
EX3400-10_11-VC# set system accounting events [change-log interactive-
commands login] destination radius server 172.16.0.17 secret JUNIPER123
```

To send to TACACS instead of RADIUS, specify `tacplus` versus `radius` and don't forget the secret key.

Configuring TACACS+

Terminal Access Controller Access-Control System Plus (TACACS+) is an authentication protocol developed by Cisco and released as an open standard in the early 1990's. There are some major differences between TACACS+ and RADIUS, so let's do a quick side-by-side to point those out:

TACACS+	RADIUS
TCP port 49	UDP ports 1812/1813
Encrypts the entire Payload	Encrypts only the Password for the session
Separate Authentication and Authorization	Combined Authorization and Authentication
Designed for Device Administration	Created to control Network Access

In the lab, we have loaded tac_plus-4.0.4.26-1.el6.nux.x86_64.rpm on CentOS 7. The same user: labuser1 with password: labP@$$1 has been created in the /etc/tac_plus.conf file:

```
$ sudo rpm -ivh tac_plus-4.0.4.26-1.el6.nux.x86_64.rpm
[sudo] password for labuser:
warning: tac_plus-4.0.4.26-1.el6.nux.x86_64.rpm: Header V4 RSA/
SHA1 Signature, key ID 85c6cd8a: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:tac_plus-4.0.4.26-1.el6.nux      ################################# [100%]

# tac_pwd
Password to be encrypted: labP@$$1
tfIupE.ZfiQK2

# vi tac_plus.conf
key = "JUNIPER123"
accounting file = /var/log/tac.log
user = labuser1 {
          #login = cleartext labP@$$1
          login = des tfIupE.ZfiQK2
          service = junos-exec {
                local-user-name = remote-su
              }
}
```

The tac_plus configuration is pretty simple. You have a Shared Secret just like RADIUS and you tell it where to find the accounting logs. Then you specify the user, labuser1, and provide a clear text password for the user.

NOTE You should note in the previous example, we left the cleartext password commented out in the configuration, but used the `tac_pwd` command on the server to generate a DES-hashed password. Normally we would remove the clear text password but we wanted to leave it there to show you can use either method.

Next, you assign a service such as junos-exec and then the local-user-name (on the Junos device) as remote-su. Once the rpm is loaded and the /etc/tac_plus.conf file is updated you simply start the service on the server:

```
# service tac_plus start
Starting tac_plus (via systemctl):                         [  OK  ]
```

Now let's focus on the Junos configuration to support a tacplus server. Log into your Junos device and add the following commands to configure tacplus. For this example, let's use the EX4600 VC consisting of EX4600-4/5:

```
$ ssh 172.16.0.4
Password:
--- Junos 14.1X53-D40.8 built 2016-11-09 02:13:22 UTC
{master:0}
EX4600-4_5-VC> edit
Entering configuration mode
{master:0}[edit]
EX4600-4_5-VC# set system authentication-order [tacplus password ]

{master:0}[edit]
EX4600-4_5-VC# set system tacplus-server 172.16.0.17 secret "JUNIPER123"

{master:0}[edit]
EX4600-4_5-VC# set system tacplus-server 172.16.0.17 port 49
{master:0}[edit]
EX4600-4_5-VC# set system tacplus-server 172.16.0.17 source-address 172.16.0.4

{master:0}[edit]
EX4600-4_5-VC# set system login user remote-su class super-user

{master:0}[edit]
EX4600-4_5-VC# commit and-quit
fpc0:
configuration check succeeds
fpc1:
commit complete
fpc0:
commit complete
Exiting configuration mode
```

A couple of items need to be pointed out with the configuration. The secret tacplus key is entered in clear text but is immediately hashed. We specified the port to use for TACACS+ and also specified the source-address to use with TACACS+. The authentication order is also specified. Without including the authentication order of tacplus and placing it before the password, it would never be called. Now you can test our server and configuration using the labuser1 configured earlier:

```
$ ssh labuser1@172.16.0.4
Password:
--- Junos 14.1X53-D40.8 built 2016-11-09 02:13:22 UTC
{master:0}
labuser1@EX4600-4_5-VC>
```

Just like RADIUS you can see that TACACS+ can be used to authenticate, authorize, and provide accounting for Junos systems. You can also use the permission flags in the remote-su user to limit the command set available as well as the regular expression for allowing and denying configuration commands.

MORE? Configuring TACACS+ is just as simple as configuring RADIUS and some customers may even want to add multiple RADIUS and multiple TACACS+ servers to their configuration and that is perfectly acceptable. To learn more about configuring TACPLUS on the Junos OS. visit the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/tacacs-authentication-configuring.html.

In review, we loaded the available rpm for CentOS to create a TACPLUS server on CentOS for our lab, then covered the basics of creating the tac_plus.conf file as well as the Junos configuration. Once we committed the configuration we logged in with labuser1 and successfully tested the configuration of the TACPLUS server in Junos.

Configuring Syslog

According to Wikipedia (https://en.wikipedia.org/wiki/Syslog): "Syslog was created in the 1980's, by Eric Allman, as part of the Sendmail project."

The Syslog protocol was later standardized in 2009 by the IETF through RFC5424. It is one of the most important Fault Management (FM) tools for a network engineer. Syslog allows all the devices in your network to send event and fault information back to a centralized server where you can easily identify when an event took place and your FM systems can provide alerts to you and your NOC.

Syslog has come a long way from its original design when it used UDP port 514 as its transport mechanism and just dumped plaintext to a file. In the last decade, we have seen the rise of Syslog-NG, an open-source application that builds on the old syslog daemon and provides a couple of new features such as filtering and database population, but it is much more complex to get up and running then rsyslog. For our purposes rsyslog will work just fine for the small lab and that's what we will be running on the CentOS server.

The rsyslog package is available in the yum repository as *rsyslogd.x86* as shown in this output from our server:

```
$ sudo yum search rsyslog
[sudo] password for user:
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: repos-va.psychz.net
 * epel: mirror.us-midwest-1.nexcess.net
```

```
 * extras: mirrors.liquidweb.com
 * updates: ftpmirror.your.org
=================================== N/S matched: rsyslog ===================================
pcp-pmda-rsyslog.x86_64 : Performance Co-Pilot (PCP) metrics for Rsyslog
rsyslog-doc.x86_64 : HTML Documentation for rsyslog
rsyslog-elasticsearch.x86_64 : ElasticSearch output module for rsyslog
rsyslog-gnutls.x86_64 : TLS protocol support for rsyslog
rsyslog-gssapi.x86_64 : GSSAPI authentication and encryption support for rsyslog
rsyslog-libdbi.x86_64 : Libdbi database support for rsyslog
rsyslog-mmnormalize.x86_64 : Log normalization support for rsyslog
rsyslog-mysql.x86_64 : MySQL support for rsyslog
rsyslog-pgsql.x86_64 : PostgresSQL support for rsyslog
rsyslog-relp.x86_64 : RELP protocol support for rsyslog
rsyslog-snmp.x86_64 : SNMP protocol support for rsyslog
uwsgi-logger-rsyslog.x86_64 : uWSGI - rsyslog logger plugin
rsyslog.x86_64 : Enhanced system logging and kernel message trapping daemon
rsyslog-crypto.x86_64 : Encryption support
rsyslog-mmaudit.x86_64 : Message modification module supporting Linux audit format
rsyslog-mmjsonparse.x86_64 : JSON enhanced logging support
rsyslog-mmsnmptrapd.x86_64 : Message modification module for snmptrapd generated messages
rsyslog-udpspoof.x86_64 : Provides the omudpspoof module
```

To install rsyslog on the server using the yum repository, simply install it like this:

```
$ sudo yum install rsyslog
Loaded plugins: fastestmirror, langpacks
base                                          | 3.6 kB  00:00:00
epel/x86_64/metalink                          |  13 kB  00:00:00
epel                                          | 4.3 kB  00:00:00
extras                                        | 3.4 kB  00:00:00
updates                                       | 3.4 kB  00:00:00
Loading mirror speeds from cached hostfile
 * base: repos-va.psychz.net
 * epel: mirror.us-midwest-1.nexcess.net
 * extras: mirrors.gigenet.com
 * updates: ftpmirror.your.org
Package rsyslog-7.4.7-16.el7.x86_64 already installed and latest version
Nothing to do
```

As you can see, the server already has rsyslog installed and it is the latest version. Before spinning up the service, let's take a look at the /etc/rsyslog.conf file and make a couple of improvements. The area we are interested is the RULES area:

```
#### RULES ####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages
# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure
# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog

# Log cron stuff
```

```
cron.*                                                  /var/log/cron
# Everybody gets emergency messages
*.emerg                                                 :omusrmsg:*
# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler
# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

Basically the output means that anything coming into the system, with `info` or higher, will get logged to the /var/log/messages file. You don't want to put all of your network device logs in the messages file, so you are going to specify a specific facility and break out all the error levels. Therefore, let's add the following to the rules area:

```
local1.emerg          /var/log/junos/rtr-0-emerg.log
local1.alert          /var/log/junos/rtr-1-alert.log
local1.crit             /var/log/junos/rtr-2-crit.log
local1.err            /var/log/junos/rtr-3-err.log
local1.warn             /var/log/junos/rtr-4-warn.log
local1.notice           /var/log/junos/rtr-5-notice.log
local1.info             /var/log/junos/rtr-6-info.log
local1.debug          /var/log/junos/rtr-7-debug.log
```

With this configuration, you can combine the facility code key word `local1` and break out each of the severity levels from `emerg` to `debug`. The Facility and Severity Codes are defined in RFC3164. This is important to understand because you are going to use these codes in your Junos configuration:

Numerical Code	Facility
0	kernel messages
1	user-level messages
2	mail system
3	system daemons
4	security/authorization messages (note 1)
5	messages generated internally by syslogd
6	line printer subsystem
7	network news subsystem
8	UUCP subsystem
9	clock daemon (note 2)
10	security/authorization messages (note 1)
11	FTP daemon
12	NTP subsystem
13	log audit (note 1)
14	log alert (note 1)
15	clock daemon (note 2)
16	local use 0 (local0)
17	local use 1 (local1)
18	local use 2 (local2)
19	local use 3 (local3)
20	local use 4 (local4)
21	local use 5 (local5)
22	local use 6 (local6)
23	local use 7 (local7)

There are also seven severity levels defined in RFC3164:

```
Numerical     Severity
Code

0               Emergency: system is unusable
1               Alert: action must be taken immediately
2               Critical: critical conditions
3               Error: error conditions
4               Warning: warning conditions
5               Notice: normal but significant condition
6               Informational: informational messages
7               Debug: debug-level messages
```

As you can see, we are taking the local1 facility and breaking out each of the severities in the `rsyslog.conf` file so that when we send syslogs from our Juniper devices, it will break them down by severity. In addition, let's say you want to log your core, aggregation, and access layers separately, so you can easily identify them in the logs. No problem, just assign a different facility to those groups such as local2 and local3 respectively, add them to the rsyslog.conf file, and touch the files, and it will all work. To start the syslog daemon in CentOS 7 start the service:

```
$ sudo service rsyslog start
Redirecting to /bin/systemctl start  rsyslog.service
$ sudo service rsyslog status
Redirecting to /bin/systemctl status  rsyslog.service
 rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2017-05-19 13:44:08 CDT; 12s ago
 Main PID: 30756 (rsyslogd)
   CGroup: /system.slice/rsyslog.service
           └─30756 /usr/sbin/rsyslogd -n
```

Now, let's get to the fun stuff and configure the Junos devices for syslog using `local1` as our facility. The following is a basic syslog configuration that will use local1 as the facility and then send specific information off-box, via UDP/514, to the syslog server:

```
# set system syslog host 172.16.0.17 facility-override local1
# set system syslog host 172.16.0.17 log-prefix EX4300-6
# set system syslog host 172.16.0.17 source-address 172.16.0.6
# set system syslog host 172.16.0.17 kernel info
# set system syslog host 172.16.0.17 interactive-commands any
# set system syslog host 172.16.0.17 structured-data
# commit comment "adding syslog configuration to the device//SER"
# exit
```

There are a couple of things to point out here. First is `facility-override local1`. If you do not specify a facility override it will default to local7. Remember local7 will default to /var/log/messages and you don't want that. Next is `log-prefix`. This command will prefix all the log entries with whatever you put there, so if you wanted it to say LAB or STOREX or some other keyword, you can enter it here and it will show up in the logs.

NOTE The source address is on our out-of-band network but best practice is to utilize the loopback interface lo0.0 as your source address, for not only syslog but, ntp, ssh, tacacs, radius, snmp, etc.

Specifying `interactive-commands any` ensures that you capture all commands entered into the device while `kernel info` states that you only want to see info level messages or higher.

While we have discussed how to build the syslog server using rsyslog, and identified some neat ways to use facilities and severity levels, and even successfully configured syslog on our Junos device, we are not yet done with logging. Since Junos is built on FreeBSD, you have an amazing logging infrastructure and can actually get a lot of information right on the box itself. Let's configure local logging for our Junos device and take a look at the underlying system:

```
# set system syslog file messages any notice
# set system syslog file messages authorization none
# set system syslog file interactive-commands interactive-commands any
# set system syslog file default-log-messages any any
# set system syslog file default-log-messages structured-data
# set system syslog file User-Auth authorization any
# set system syslog file User-Auth interactive-commands any
# set system syslog console any any
```

A quick look at this configuration and you can see that the `/var/log/messages` file on our device will contain any message with severity level `notice` or higher. We will not log authorization messages, as indicated by the keyword `none`. `Interactive-commands` will be placed in a file called interactive-commands and will log any interaction with the user interface.

NOTE Logging interactive commands to the device, and in a separate file, will allow you to quickly see what commands have been entered into the device.

`Default-log-messages` will include all messages at any severity level and use the structured data format. `User-Auth` will contain the authorization messages and all interactive commands from the UI. Finally, the `console` is set to log any message at any severity level.

NOTE Unless you are directed to do so by some security guideline, it is best practice to *never* log all messages to console. *Why?* If your device is having a hardware issue or just a flapping interface it will spew messages on the console, and your console user experience will be absolutely miserable. If you are logging in via the RS232 console, then you are already there because of an issue, and the last thing you want is to have to wade through tons of logs showing up on your CLI as you try to get in and fix a problem. So, while we showed you it is possible *please don't do this* unless you absolutely have to, and even then, try to fight it.

MORE? There's a lot of information on Syslog in this chapter but there is so much more that you can do with it. If you want to get more granular, or find something more specific, please visit the Juniper TechLibrary to learn more: https://www.juniper.net/documentation/en_US/junos/topics/concept/syslog-messages-overview-qfx-series.html.

There are numerous logs that are maintained on the device itself. To see a list of all the available logs simply use the operational `> show log ?` command and this will show all logs available on the system. You will notice that the messages file will be compressed using gzip and the logs will roll over once the reach a specific size. You can still view the contents of those logs, but you need to include the [0-9].gz on the end of the file, for example:

```
> show log messages?
Possible completions:
  <filename>           Name of log file
  messages              Size: 205369, Last changed: May 20 15:39:30
  messages.0.gz         Size: 29312, Last changed: May 18 22:15:00
  messages.1.gz         Size: 35085, Last changed: May 10 02:30:00
  messages.2.gz         Size: 35029, Last changed: May 01 13:00:00
  messages.3.gz         Size: 54145, Last changed: Apr 20 03:45:00
```

As you can see there are several message files and they are being rolled and compressed about every nine days or when they reach a certain file size. You can specify the number of files to maintain and the file size in the syslog configuration. For now, let's take a look at one of the compressed messages files:

```
> show log messages.0.gz | last
          May 18 21:19:25  switch1 xntpd[18569]: NTP Server 192.168.2.5 is Unreachable
          May 18 21:36:29  switch1 xntpd[18569]: NTP Server 192.168.2.5 is Unreachable
          May 18 21:53:34  switch1 xntpd[18569]: NTP Server 192.168.2.5 is Unreachable
          May 18 22:06:43  switch1 last message repeated 2 times
          May 18 22:10:38  switch1 xntpd[18569]: NTP Server 192.168.2.5 is Unreachable
          May 18 22:15:00 switch1 newsyslog[41779]: logfile turned over due to size>1024K
```

TIPS The `|` "pipe" command can be used to take you to the last line of the file by using `| last`. You can also use the pipe command to exclude specific information, for example, `show log messages | except repeated`, or if you want, to only search the log for a specific time, such as `show log messages.0.gz | match "May 18 21:36"`. You can also search without using the pipe command. For example, if you run `show log messages` it will give you the paged version of the log and you can then use the `/` (forward slash) with a key word and it will search the entire file (or you can press Shift+G and it will take you to the end of the file). To get out of the paging mode simply press `q` to end the session.

This device is not synching with the NTP server for some reason, so let's take a look.

```
> show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================================
 192.168.2.5      .NKEY.          16 -    -   64    0    0.000    0.000 4000.00
```

The messages logs were correct. If we hadn't been advised in the logs this could have gone unnoticed and caused us some issues. The NKEY reference usually points to not having the trusted key setting in the configuration, so let's check that out:

```
> show configuration | display set | match ntp
set system ntp boot-server 192.168.2.5
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value "$9$hG5yeM7-boJUfT1EcyvMaJGDqm5QF/Ap"
set system ntp server 192.168.2.5 key 1
set system ntp source-address 192.168.2.6
```

Sure enough, it's missing the trusted-key statement, so let's put it in there and see what happens:

```
# set system ntp trusted-key 1
[edit]
# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode

> show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================
 192.168.2.5      66.96.99.10      3 -    1   64    1    4.385  -14.946   0.007

> show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================
*192.168.2.5      66.96.99.10      3 -    9   64    3    3.428  -15.455   0.472

> show system uptime
fpc0:
--------------------------------------------------------------------------
Current time: 2017-05-20 16:05:32 UTC
Time Source:  NTP CLOCK
System booted: 2017-04-20 02:05:36 UTC (4w2d 13:59 ago)
Protocols started: 2017-04-20 02:08:29 UTC (4w2d 13:57 ago)
Last configured: 2017-05-20 16:01:08 UTC (00:04:24 ago) by user
 4:05PM  up 30 days, 14 hrs, 1 users, load averages: 0.41, 0.44, 0.40
```

Excellent! Thanks to the messages log we have identified and corrected an NTP server issue that was being reported. Now, let's check the messages log to see what it says about the ordeal:

```
> show log messages | match ntp
May 20 15:26:25  switch1 xntpd[18569]: NTP Server 192.168.2.5 is Unreachable
May 20 15:43:31  switch1 xntpd[18569]: NTP Server 192.168.2.5 is Unreachable
May 20 16:06:59  switch1 xntpd[18569]: ntpd 4.2.0-a Fri Sep 23 12:29:28  2016 (1)
May 20 16:07:12  switch1 xntpd[18569]: kernel time sync enabled 2001
```

You can see in the messages log file that NTP is now synchronized and enabled.

Being able to monitor the logs while you are troubleshooting is an important tool and we would be amiss if you weren't introduced to it here. By using the `monitor start` command, you can get a real time feed of the logs as they are coming into the system. Let's use it to take down the interface, and then watch it come back up. In the lab let's disable the xe-1/01 interface on QFX5100-2 and then watch the logs on EX3400-10:

```
EX3400-10_11-VC> monitor stop
EX3400-10_11-VC> monitor start messages
May 20 12:29:31  EX3400-10_11-VC rpd[2921]: EVENT <UpDown> xe-0/2/2.0 index 557 <Broadcast Multicast>
address #0 30.b6.4f.c6.27.22
May 20 12:29:31  EX3400-10_11-VC rpd[2921]: RPD_OSPF_NBRDOWN: OSPF neighbor 10.1.0.0 (realm ospf-v2
xe-0/2/2.0 area 0.0.0.0) state changed from Full to Down due to KillNbr (event reason: local router ID
changed)
May 20 12:29:31  EX3400-10_11-VC rpd[2921]: EVENT UpDown xe-0/2/2.0 index 557 10.1.0.1/31 -> zero-len
<Broadcast Multicast Localup>
May 20 12:29:31  EX3400-10_11-VC rpd[2921]: RPD_OSPF_NBRDOWN: OSPF neighbor 10.1.0.7 (realm ospf-v2
xe-1/2/1.0 area 0.0.0.0) state changed from Full to Down due to KillNbr (event reason: local router ID
changed)
May 20 12:29:31  EX3400-10_11-VC rpd[2921]: EVENT <UpDown> xe-0/2/2 index 700 <Broadcast Multicast>
address #0 30.b6.4f.c6.27.22
May 20 12:29:31  EX3400-10_11-VC rpd[2921]: STP handler: IFD=xe-0/2/2, op=change, state=Discarding,
Topo change generation=0
May 20 12:29:31  EX3400-10_11-VC rpd[2921]: *STP Change*, notify to other modules
May 20 12:29:31  EX3400-10_11-VC mib2d[2920]: SNMP_TRAP_LINK_DOWN: ifIndex 597, ifAdminStatus up(1),
ifOperStatus down(2), ifName xe-0/2/2
```

As you can see, all the events of the interface are going down – spanning tree updates and OSPF status. You should also note that before we issued the `monitor start messages` command, the `monitor stop` command was issued. This is an experience tip – issue the `monitor stop`, then press the Enter key. The reason: trying to type while the logs are scrolling is a bit difficult sometimes. Another way to stop the monitor command is by using the ESC-q sequence.

TIP You can monitor any of the active logs using this method and even your locally defined logs such as traceoptions files.

Another often used command for logs is the `file archive` command. This command is used to Tape Archive (TAR) and (GNU zip or gzip) compress all of the logs into a single file so that they can be sent off to JTAC.

NOTE A few years ago Juniper enabled automatic log parsing output on any logs that are sent to a case. There are a couple of requirements and you can get more information on that process here: https://www.juniper.net/documentation/en_US/junos/topics/task/troubleshooting/troubleshooting-logs-compressing.html.

To collect and compress all of the files on your EX Series device you can run the following command on a standalone device, or if you have a virtual chassis (VC) with dual routing-engines (REs), you can collect the logs off of both using the following operational commands:

```
EX3400-10_11-VC> file list /var/tmp/
/var/tmp/:
bcast.bdisp.log
bcast.disp.log
bcast.rstdisp.log
bcast.undisp.log
ex_autod_config
ex_autod_rollback_cfg
…
EX3400-10_11-VC> file archive source /var/log/ destination /var/tmp/EX3400_LOGS_0520171155.
logs compress
tar: Removing leading '/' from member names
{master:0}

EX3400-10_11-VC> file list detail /var/tmp/
/var/tmp/:
total blocks: 8440
-rw-------  1 labuser wheel   2087200 May 20 12:55 EX3400_LOGS_0520171155.logs.tgz
```

To collect the logs from the backup routing engine, perform the following steps:

```
EX3400-10_11-VC> request routing-engine login backup
--- Junos 15.1X53-D50.2 Kernel 32-bit  JNPR-11.0-20160614.329646_build
warning: This chassis is operating in a non-master role as part of a virtual-chassis (VC) system.
warning: Use of interactive commands should be limited to debugging and VC Port operations.
warning: Full CLI access is provided by the Virtual Chassis Master (VC-M) chassis.
warning: The VC-M can be identified through the show virtual-
chassis status command executed at this console.
warning: Please logout and log into the VC-M to use CLI.
{backup:1}
EX3400-10_11-VC>

{backup:1}
EX3400-10_11-VC> file archive source /var/log/ destination /var/tmp/EX3400_RE1_LOGS_0520171156.
logs compress
tar: Removing leading '/' from member names

{backup:1}
EX3400-10_11-VC> file list /var/tmp/
/var/tmp/:
EX3400_RE1_LOGS_0520171156.logs.tgz
```

Now that the logs are all TAR'd and compressed on the backup routing engine (note the indicator changed from master to backup:1), you need to copy it to the primary routing engine to SCP it off the system. To do that, exit the backup and then use the `file copy` command to pull it from the backup to the primary:

```
{backup:1}
labuser@EX3400-10_11-VC> exit
rlogin: connection closed
```

```
{master:0}
labuser@EX3400-10_11-VC> file copy re1:/var/tmp/*.tgz /var/tmp/

{master:0}
labuser@EX3400-10_11-VC> file list /var/tmp/EX34* detail
-rw-------  1 labuser wheel   2087200 May 20 12:55 /var/tmp/EX3400_LOGS_0520171155.logs.tgz
-rw-------  1 labuser wheel   2114254 May 20 21:44 /var/tmp/EX3400_RE1_LOGS_0520171156.logs.tgz
total files: 2
```

All of the log files in the /var/log/ directory are now rolled up into the `EX4300_LOGS_0520171155.logs.tgz` file and can easily be secure copied (SCP'd) off the box and attached to a case. As you can see from the command, the source is `/var/log/` and the destination is the `/var/tmp/` directory with the filename `EX4300_LOGS_0520171155.logs`. Once the command is completed you will see that it has been compressed just like it was told. A quick `file list /var/tmp` detail will show the files that exist in that directory. For more information on this feature visit: https://www.juniper.net/documentation/en_US/junos/topics/task/troubleshooting/troubleshooting-logs-compressing.html.

NOTE You can also archive the logs from the line cards in a virtual chassis by logging into the member with `> request session member N` (*N* being the member number) and then issuing the command `> file archive source /var/logs/ destination re0:/var/tmp/MEM4.LOGS compress`.

TIP If for any reason you get a connection closed and cannot access another member of the chassis, you can use the following command from the shell to access that member: `root% rlogin -Ji fpc<member num>`.

This will collect all the logs and copy them to the RE so you can SCP them off. A quick `file list/var/tmp` will verify that the logs were copied off the line card:

```
> file list /var/tmp/MEM4.logs.tgz detail
-rw-------  1 lab wheel   2036892 Jun 22 15:58 /var/tmp/MEM4.logs.tgz
total files: 1
```

This section demonstrated how to build a simple syslog server using rsyslog and UDP port 514, as well as some improvements to syslog such as syslog-ng. We configured syslog on a Junos device for remote logging and for local logging, and then went over several features within those configurations.

Also discusses was how to look into the logs and use the pipe command to filter on specific information. You were introduced to the `monitor` command to watch real time log entries, and then you saw how to archive all of the files and copy them off-box to send to JTAC.

At this point you should have a pretty good handle on configuring syslog for your devices. If you need more information try these pathway pages at the Juniper TechLibrary: https://www.juniper.net/documentation/en_US/junos12.3/information-products/pathway-pages/system-basics/system-logs.html#tab=undefined.

Configuring Timezone

The default time zone for Junos is UTC (Universal Time Coordinated or Coordinated Universal Time, formally known as GMT or Greenwich Mean Time). Let's modify the local time zone under the `edit system` hierarchy and then set it to America/Chicago:

```
# set system time-zone America/C?
Possible completions:
  <time-zone>          Time zone name or POSIX-compliant time zone string (<continent>/<major-
city> or <time-zone>)
  America/Cambridge_Bay
  America/Campo_Grande
  America/Cancun
  America/Caracas
  America/Catamarca
  America/Cayenne
  America/Cayman
  America/Chicago
  America/Chihuahua
  America/Cordoba
  America/Costa_Rica
  America/Cuiaba
  America/Curacao
# set system time-zone America/Chicago
# commit
```

Now you can check the system clock to make sure that you are on Central Daylight Time (CDT).

```
# run show system uptime
Current time: 2017-05-19 15:30:42 CDT
System booted: 2017-05-18 16:56:11 CDT (22:34:31 ago)
Protocols started: 2017-05-18 16:56:59 CDT (22:33:43 ago)
Last configured: 2017-05-19 15:30:33 CDT (00:00:09 ago) by labuser1
 3:30PM  up 22:35, 2 users, load averages: 0.00, 0.00, 0.00
```

There are over a hundred different time zones to set your device to, including Zulu. If you don't want to use a location-based time zone, you can choose to use offset instead. GMT or UTC is five hours ahead of CDT, so instead of specifying CDT you could use an offset from GMT and still get the same local time. Let's take a look:

```
# set system time-zone GMT?
Possible completions:
  <time-zone>          Time zone name or POSIX-compliant time zone string (<continent>/<major-
city> or <time-zone>)
  GMT
  GMT+1
  GMT+10
  GMT+11
  GMT+12
  GMT+2
  GMT+3
  GMT+4
  GMT+5
```

```
  GMT+6
  GMT+7
…
  GMT-9
# set system time-zone GMT-5
[edit]
labuser@VEX1# commit
```

Now let's check the system uptime to see if properly reflects the GMT-5 time:

```
> show system uptime
Current time: 2017-05-19 15:43:41 GMT-5
System booted: 2017-05-18 16:56:08 GMT-5 (22:47:33 ago)
Protocols started: 2017-05-18 16:56:56 GMT-5 (22:46:45 ago)
Last configured: 2017-05-20 01:42:17 GMT-5 (-9:-58:-36 ago) by labuser1
```

Good timing. For further information on time zone, please refer to: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/time-zone-configuring.html.

Configuring NTP

The Network Time Protocol (NTP) is essential to provide the correct time in syslog messages and SNMP traps. There is nothing worse than looking at your logs and seeing a device that is sending messages with a date of 1 June 1970. That kind of time is just not helpful, especially when you are trying to correlate the exact time an interface went down or some other event that took place on the network. NTP is an important component and should be configured on all of your network devices. You can build an NTP server, easily and quickly, on CentOS 7, by simply loading NTP from the CentOS yum repository. That's exactly what we did for this book's lab, so let's do a quick walk through to show how easy it is to get a stratum level 2 clock for your network.

First pull down the NTP package on the server and install it using yum:

```
# yum install ntp
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.trouble-free.net
 * epel: mirror.us.leaseweb.net
 * extras: repos-va.psychz.net
 * updates: mirror.math.princeton.edu
Resolving Dependencies
--> Running transaction check
---> Package ntp.x86_64 0:4.2.6p5-25.el7.centos.2 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
================================================================================
 Package           Arch        Version                   Repository    Size
================================================================================
Installing:
 ntp             x86_64       4.2.6p5-25.el7.centos.2    updates       547 k
Transaction Summary
================================================================================
```

```
Install  1 Package
Total download size: 547 k
Installed size: 1.4 M
Is this ok [y/d/N]: y
Downloading packages:
ntp-4.2.6p5-25.el7.centos.2.x86_64.rpm                    | 547 kB  00:00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : ntp-4.2.6p5-25.el7.centos.2.x86_64                    1/1
  Verifying  : ntp-4.2.6p5-25.el7.centos.2.x86_64                    1/1
Installed:
  ntp.x86_64 0:4.2.6p5-25.el7.centos.2
Complete!
```

Now you need to edit the /etc/ntp.conf file and add some publicly accessible NTP servers. To get a list of servers go to http://www.pool.ntp.org/en/use.html and follow their *join and use* processes:

```
# cat ntp.conf | grep -v "#"
driftfile /var/lib/ntp/drift
restrict default nomodify notrap nopeer noquery
restrict 127.0.0.1
restrict ::1
restrict 192.168.56.0 mask 255.255.255.0 nomodify notrap
restrict 172.16.0.0 mask 255.255.255.0 nomodify notrap
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
includefile /etc/ntp/crypto/pw
keys /etc/ntp/keys
disable monitor
```

The highlighted section above is what was added to the default ntp.conf file via vi. You may notice that we read the file out and used GREP to remove all the comments. This makes the file much easier to read. However, you should read the entire file and understand it if you are administrating an NTP server on your own as there are some security vulnerabilities that you need to lock down.

You also need to set a key to use for NTP authentication. To do that you need to edit the key file under /etc/ntp/. Here we add two keys to be used with NTP:

```
# more keys
# For more information about this file, see the man page ntp_auth(5).
#
# id       type key
1          M JUNIPER123
2          M WATERMELON123
```

Let's make sure the server has started:

```
# service ntpd start
```

Now check your connectivity to the NTP server by using the `ntpq -p` command:

```
$ ntpq -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 192.168.56.1    .XFAC.          16 u    - 1024    0    0.000    0.000   0.000
*66.96.99.10     204.9.54.119     2 u  309  512  377   43.795    5.533   1.404
+66.96.98.9      64.250.105.227   2 u  261  512  377   43.901    5.952   1.437
 38.229.71.1     .STEP.          16 u    - 1024    0    0.000    0.000   0.000
 159.203.158.197 .STEP.          16 u    - 1024    0    0.000    0.000   0.000
```

The output shows that we are synched on 66.96.99.10 and have a stratum level 2 clock. It also shows delay, offset, and jitter. If you don't see the * and + characters then you are not synced, and if you don't see any delay offset or jitter then you more than likely are not synchronized either. Fortunately, we have two servers that are providing us a stratum level 2 clock.

NOTE NTP has 16 stratum levels. Stratum level 0 is the master clock and cannot be used on the network. Our server has a stratum level 2 so we are 3 stratums away from a master clock. The further you are away from the master clock (stratum-0) the more delay offset and jitter you will see.

Now that we have a good clock source and our server is synchronized we can provide timing to our network devices. You need to think about a couple of things, so let's make a list:

1. Identify the NTP server
2. Establish a secure key using MD5 to hash it
3. Identify the network that NTP allows for access
4. Set our Junos device to be within 30 seconds of the actual time
5. Verify the device is synched with the NTP server.

Using the lab, let's point to 172.16.0.17 for the NTP server that we just configured. (The server is authorized to serve time on this network as seen in the ntp.conf file above.)

```
# set system ntp server 172.16.0.17 prefer
```

The `prefer` keyword says to prefer this server over any other servers listed. Now let's add a secret key and hash it using Message Digest 5 (MD5):

```
# set system ntp authentication-key 1 type md5 value "JUNIPER123"
# set system ntp trusted-key 1
# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value "$9$tXwY01ElKW-VsUj/Ap0REdVwYaZDikPTz"
set system ntp trusted-key 1
set system ntp server 172.16.0.17 prefer
```

Next, tell the device which key to use because you can have multiple keys with different md5 values to switch out whenever you want. For now, let's just focus on one key:

```
# set system ntp server 172.16.0.17 key 1
```

That is the basic configuration for NTP on a Junos device to connect it to the secure NTP server in your network. However, in step 3 you want to specify a source address for NTP:

```
# set system ntp source-address 172.16.0.6
```

Now go ahead and commit what you have so far and then you will set your system time:

```
# commit
# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value "$9$tXwY01ElKW-VsUj/Ap0REdVwYaZDikPTz"
set system ntp server 192.168.56.1 key 1
set system ntp server 192.168.56.1 prefer
set system ntp source-address 192.168.56.11
```

Excellent, so now let's go ahead and set the date from operational mode:

```
> set date ?
Possible completions:
  <time>               New date and time (YYYYMMDDhhmm.ss)
  ntp                  Set system date and time using Network Time Protocol servers
```

As you can see, there are two ways to set the date. Type it in manually or point to the NTP server. Let's do it both ways just for practice:

```
> set date 201705191724
Fri May 19 17:24:00 GMT-5 2017
```

Now let's try it with the NTP server:

```
> set date ntp 172.16.0.17
20 May 03:25:58 ntpdate[12591]: step time server 172.16.0.17 offset 36009.441756 sec
```

The last thing in the to-do list was to check the status of NTP. There are two commands that provide information and let us know if we are synchronized with the server. First, the operational command, `show ntp status`, provides a lot of information:

```
# run show ntp status
status=0664 leap_none, sync_ntp, 6 events, event_peer/strat_chg,
version="ntpd 4.2.0-a Sun Feb  1 06:55:42 UTC 2015 (1)",
processor="i386", system="Junos14.2-20150201.0", leap=00, stratum=4,
precision=-21, rootdelay=66.036, rootdispersion=512.466, peer=2604,
refid=172.16.0.17,
reftime=dcc9f2d4.3b1c7c55  Fri, May 19 2017 17:40:20.230, poll=6,
clock=dcc9f2e6.910b23bd  Fri, May 19 2017 17:40:38.566, state=4,
offset=39.711, frequency=-499.674, jitter=1.676, stability=0.085
```

You want the reftime and clock to have the same date. If the reftime is 7 FEB 2036, then it is *not* synchronized. The output from the `status` command shows that it is in sync. Now let's use the other operational command, `show ntp associations`, to see the NTP connection:

```
> show ntp associations
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*172.16.0.17    66.96.99.10      3 -   62   64   17    0.283   72.534  36.735
```

The asterisk shows that you are synced to the remote device and that you have a stratum level 3 connection with the server. The secret key is hashed using MD5 to prevent someone from spoofing our NTP server address and sending fake time to our network. For more in-depth information on configuring NTP on Junos devices, see the Juniper TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/network-time-protocol-time-server-time-services-configuring.html.

Configuring EX Switches to Act as NTP Servers

There are times when you may want your EX Series switches to provide time to the rest of the network. Juniper EX Series switches support four different NTP operating modes:

- `Client`: Relies on a server; keyword: `server`.
- `Symmetric Active`: Allows the switch to operate as an NTP server with primary and backup designation: Keyword: `peer`.
- `Broadcast`: The EX Series acts as an NTP server: Keyword: `broadcast`.
- `Server`: The EX Series serves time to the other network devices: Keyword: `server`.

For this section, let's go back to the book's simple four-node topology and addi an NTP server into the mix as shown in Figure 10.1.

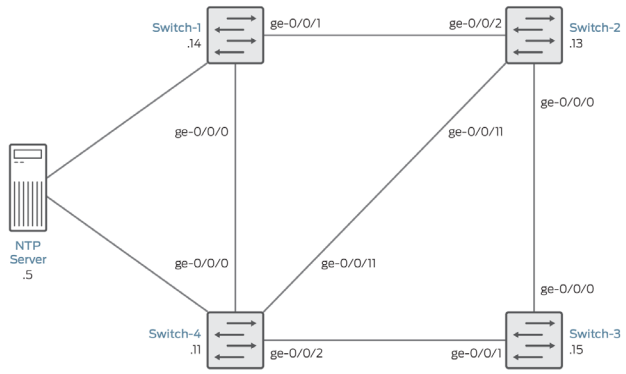


Figure 10.1 *Network Time Protocol (NTP) Diagram.*

Currently none of the devices in Figure 10.1 are configured for NTP and they are not able to reach the NTP server.

```
SWITCH-1> show ntp associations
localhost: timed out, nothing received
***Request timed out
SWITCH-1> show system uptime
fpc0:
--------------------------------------------------------------------------
Current time: 2010-01-03 20:17:01 UTC
Time Source:  LOCAL CLOCK
```

Our first step is to become a client to the NTP source (this is usually a separate appliance or a source available over the Internet). The command to become a client is a little counter-intuitive because it is actually `server`:

```
SWITCH-1# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value «$9$wn2oZHqP36CRh-bs2JZn69AO1EcyKWL»
set system ntp server 192.168.2.5 key 1
set system ntp trusted-key 1

SWITCH-1# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================================
*192.168.2.5      96.244.96.19     3 -    4   64    1    0.833   -0.584   0.127

SWITCH-1# run show ntp status
status=0664 leap_none, sync_ntp, 6 events, event_peer/strat_chg,
version=»ntpd 4.2.0-a Thu Jul 20 08:16:47  2017 (1)», processor=»arm»,
system=»Junos15.1R6-S2.1», leap=00, stratum=4, precision=-16,
rootdelay=43.024, rootdispersion=1.560, peer=41532, refid=192.168.2.5,
reftime=dd27367c.03c04f9d  Sat, Jul 29 2017 16:29:48.014, poll=6,
clock=dd27368e.6f230f9b  Sat, Jul 29 2017 16:30:06.434, state=3,
offset=0.000, frequency=0.000, jitter=0.110, stability=0.000
```

Just as in the previous chapter, add the NTP configuration and point the device to the *server* of 192.168.2.5. That makes Switch-1 a client of 192.168.2.5.

Now configure this device to also *serve* time to the rest of the network. You can use either broadcast or peer (Symmetric Active) to accomplish this. Let's do broadcast first:

```
SWITCH-1# set vlans V100 l3-interface irb.100
SWITCH-1# set interfaces irb.100 family inet address 172.16.14.1/24
SWITCH-1# set system ntp broadcast 172.16.14.0
SWITCH-1# commit

SWITCH-1# show | display set | match ntp
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value "$9$wn2oZHqP36CRh-bs2JZn69AO1EcyKWL"
set system ntp server 192.168.2.5 key 1
set system ntp broadcast 172.16.14.0
set system ntp trusted-key 1

SWITCH-1# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
*192.168.2.5      96.244.96.19     3 -   56   64   17    0.808   -3.449   0.542
 172.16.14.1      .BCST.          16 -    -   64    0    0.000    0.000 4000.00
```

You can see a configured a Layer 3 interface for VLAN 100 and a configured broadcast for that address so the rest of the network can point to 172.16.14.1 as the broadcast server.

The next step, configure clients Switch-2 and Switch-3:

```
SWITCH-2# set vlans V100 l3-interface irb.100
SWITCH-2# set interfaces irb.100 family inet address 172.16.14.2/24
SWITCH-2# commit

SWITCH-2# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value «$9$DmH.f36CBIhWLJUjHPf1IEcev8X7Vs2»
set system ntp server 172.16.14.1
set system ntp broadcast-client

SWITCH-2# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================================
*172.16.14.1      192.168.2.5      4 -   15   64   17    3.704   -0.658   0.483

SWITCH-2# run show ntp status
status=0664 leap_none, sync_ntp, 6 events, event_peer/strat_chg,
version=»ntpd 4.2.0-a Thu Mar 24 17:43:25  2016 (1)», processor=»arm»,
system=»Junos15.1R3.6», leap=00, stratum=5, precision=-16,
rootdelay=66.661, rootdispersion=4.515, peer=59540, refid=172.16.14.1,
reftime=dd273fc6.fa255fa9  Sat, Jul 29 2017 17:09:26.977, poll=6,
clock=dd273fdb.d3f20198  Sat, Jul 29 2017 17:09:47.827, state=3,
offset=0.000, frequency=0.000, jitter=0.411, stability=0.000
```

And now onto Switch-3.

NOTE VLAN V100 was added to this switch and so also was the VLAN to the trunked interface, ge-0/0/0, between Switch-2 and Switch-3 to allow V100 on Switch-3:

```
SWITCH-3# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value "$9$DmH.f36CBIhWLJUjHPf1IEcev8X7Vs2"
set system ntp server 172.16.14.1
set system ntp broadcast-client

SWITCH-3# run set date ntp
29 Jul 17:22:59 ntpdate[14778]: step time server 172.16.14.1 offset 25.585421 sec

SWITCH-3# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================================
*172.16.14.1     192.168.2.5      4 -   62   64    3    3.759    0.138   1.272
Switch-4 already has VLAN 100 so all we have to do is copy and paste the NTP configuration used on
Switch-2 or Switch-3, set the date manually and then we should get NTP synch.
```

TIP When trying to synchronize with an NTP server you have to be within 128 seconds of the time being served. Once you are in that window it will STEP you up in 128ms increments to get the correct time.

```
SWITCH-4# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value "$9$DmH.f36CBIhWLJUjHPf1IEcev8X7Vs2"
set system ntp server 172.16.14.1
set system ntp broadcast-client

SWITCH-4# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
================
 172.16.14.1     .STEP.          16 -   32   64    0    0.000    0.000 4000.00
```

STEP indicates that you are in the window and that the server is stepping up 128ms at a time to bring us in synch.

Now the lab entire network is synchronized off of our NTP server. You should note that the stratum level for Switch-1 is Stratum 3. This is because the local NTP server is Stratum 2. This makes our broadcast clients Stratum 4 as you can see in the associated output.

Since you have direct connections to Switch-1 and Switch-4, you could also make these primary and backup servers for the network. To accomplish that use the `prefer` keyword.

NOTE When viewing the output of the `ntp association` command you will see symbols in front of the IP addresses: * master (synced), # master (unsynced), + selected, - candidate, and ~ configured.

First, configure Switch-4 with server statements and `prefer` the NTP appliance over Switch-1:

```
SWITCH-4# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value "$9$wn2oZHqP36CRh-bs2JZn69AO1EcyKWL"
set system ntp server 192.168.2.5 prefer
set system ntp server 172.16.14.1
set system ntp trusted-key 1
```

By using the `prefer` keyword you can specify which device you want to be primary. That way you can have more than one NTP source for failover. To test this, let's disable the interface that 192.168.2.5 is reachable (emulating a real-world scenario). Check the current associations first:

```
SWITCH-4# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================================
*192.168.2.5      96.244.96.19     3 -   39   64  377    0.742   -2.644   1.146
+172.16.14.1      192.168.2.5      4 -   31   64  377    3.760  -22.486   2.095
```

This output clearly show that 192.168.2.5 is the preferred master and is synced (identified by *). Now let's disable that interface and see what happens:

```
SWITCH-4# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================================
 192.168.2.5      45.127.112.2     3 -   53   64  375    0.739   -1.684   0.720
*172.16.14.1      192.168.2.5      4 -   45   64  377    5.162  -20.013  98.945
```

You can see that 172.16.14.1 has become the master synched (*) transitioning from selected (+) and is still synched. You can also make Switch-2 and Switch-3 clients of Switch-1 and Switch-4, respectively, and then have backup connections to the main two NTP sources:

```
SWITCH-2# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value «$9$DmH.f36CBIhWLJUjHPf1IEcev8X7Vs2»
set system ntp server 172.16.14.4
set system ntp server 172.16.14.1 prefer
set system ntp broadcast-client

SWITCH-2# run show ntp associations
   remote         refid           st t when poll reach   delay   offset  jitter
==============================================================================
+172.16.14.4      172.16.14.1      4 -   51   64  377    5.160    7.470   3.244
*172.16.14.1      192.168.2.5      3 -   53   64  377    4.516   -6.597   1.158

SWITCH-3# show system ntp | display set
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value «$9$DmH.f36CBIhWLJUjHPf1IEcev8X7Vs2»
set system ntp server 172.16.14.4 prefer
set system ntp server 172.16.14.1
set system ntp broadcast-client
{master:0}[edit]
```

```
SWITCH-3# run show ntp associations
   remote          refid           st t when poll reach   delay   offset  jitter
==============================================================================
*172.16.14.4      172.16.14.1      4 -   34   64  377    4.711   -3.337   8.753
+172.16.14.1      192.168.2.5      3 -   32   64  377    5.432  -19.796   6.898
If we take away NTP on 172.16.14.4 then Switch-3 should pick up on 172.16.14.1 without any issues.

SWITCH-4# deactivate system ntp
SWITCH-4# commit

SWITCH-3# run show ntp associations
   remote          refid           st t when poll reach   delay   offset  jitter
==============================================================================
 172.16.14.4      .INIT.          16 -    -   64    0    0.000    0.000 4000.00
*172.16.14.1      192.168.2.5      3 -   32   64    1    3.663    0.567   0.171
```

And once we rollback the change made to Switch-4, then Switch-3 will pick back up on 172.16.14.4 because it is preferred:

```
SWITCH-4# rollback 1
load complete
SWITCH-4# commit
SWITCH-4# run show ntp associations
   remote          refid           st t when poll reach   delay   offset  jitter
==============================================================================
 192.168.2.5      .STEP.          16 -  502   64    0    0.000    0.000 4000.00
 172.16.14.1      .STEP.          16 -   65   64    0    0.000    0.000 4000.00

SWITCH-4# run show ntp associations
   remote          refid           st t when poll reach   delay   offset  jitter
==============================================================================
*192.168.2.5      132.163.4.101    2 -    1   64    1    1.225   14.808   0.269
+172.16.14.1      192.168.2.5      3 -    1   64    1    3.987   -2.778   0.462
```

Once everything is back up on Switch-4 let's check Switch-3 and see that it indeed has regained the preferred master clock on the network which is an EX Series switch:

```
SWITCH-3# run show ntp associations
   remote          refid           st t when poll reach   delay   offset  jitter
==============================================================================
*172.16.14.4      192.168.2.5      3 -    1   64    1    4.219    2.698   0.330
+172.16.14.1      192.168.2.5      3 -    2   64    1    4.726    0.462   0.326
```

The last NTP setting covered in this chapter is using the `peer` statement. This configuration is used to synchronize clocks *between* EX switches acting as NTP servers in the network. This is slightly confusing given the description that it allows a switch to act as an NTP server with primary and backup designation. You can already do that with both broadcast and server statement, so what does Symetric Active do using the `peer` keyword?

Basically this command is used between the two devices acting as NTP servers on the network to synchronize their clocks so that they are in lock step with each other and not serving different time to their clients on the network.

Let's do a quick configuration and check of the peer statement on the network and see what it gives us that `broadcast` or `server` keywords do not:

```
EX4600-4_5-VC# show | display set | match ntp
set system ntp authentication-key 1 type md5
set system ntp authentication-key 1 value «$9$XP4NVsaZDmfQtuMLxNwsPfTz/Cp0BESr»
set system ntp peer 172.16.0.2 key 1
set system ntp peer 172.16.0.17
set system ntp server 172.16.0.17
set system ntp trusted-key 1
set system ntp source-address 172.16.0.2
set system ntp source-address 172.16.0.4
```

Here you can see that we entered `peer` statements from the EX4600 Series in our lab topology to the NTP server 172.16.0.17 as well as to the QFX5100 Series which is acting as a server on the network. Now let's show the associations and verify that the `peer` statement is showing up as configured and will take over in case the NTP appliance fails:

```
EX4600-4_5-VC# run show ntp associations
   remote          refid           st t when poll reach   delay   offset  jitter
==============================================================================
+172.16.0.2       172.16.0.17      4 -   42   64  175    0.844   -0.179   1.191
+172.16.0.17      204.2.134.163    3 -   47  512   37    0.429   -0.445   0.078
*172.16.0.17      204.2.134.163    3 -   32   64  377    0.400   -0.449   0.114
```

As you can see things are peered and configured with the QFX5100 VC and have a stratum level 4 clock. The QFX5100 Series device also has a corresponding peer statement so that these two devices can synchronize their time for the network.

If you are wanting to set up a primary and backup NTP server, use the `server` keyword because it serves time as well as acts like a client. Use the `peer` statement to synchronize time between the primary and alternate devices acting like clients. You can have `server`, `peer`, and `broadcast` keywords all in one NTP configuration if you desire, but that is probably not best practice!

Configuring Name Server

This is a very simple task but is important if you want to be able to use Domain Name Servers (DNS) in your production network to reference devices by name instead of IP addresses:

```
EX3400-10_11-VC# set system name-server 8.8.8.8
EX3400-10_11-VC# commit
EX3400-10_11-VC# run ping juniper.net
PING juniper.net (66.129.230.17): 56 data bytes
```

You can see we used Google's name server 8.8.8.8 in our configuration and then ran a quick ping to Juniper.net from edit mode and it is able to translate the IP for us to 66.129.230.17.

We should also set our `domain-name` if we are going to be using DNS:

```
EX3400-10_11-VC # set system domain-name juniper.net
```

Once that is committed to the active configuration the domain will be appended to the host name for those host-names that are not fully qualified.

Summary

This chapter has covered a lot of information, geared around the management of the Junos device. It discussed creating local users on the device as well as RADIUS and TACACS using the remote profile. It also discussed the configuration of actual RADIUS and TACPLUS servers on CentOS 7. It configured a syslog server using rsyslog and then configured our device to point syslog to the server and verified the receipt of messages. Finally, this chapter configured name server and domain name.

That's a lot! But let's lock it all down with security next.

Chapter 11

Configuring EX Series Port Security

Configuring MAC Limiting

Within a VLAN, MAC limiting is used to limit the number of MAC addresses that can be learned. This helps to prevent devices from flooding or overflowing the Ethernet switching table and purging previously learned MAC addresses.

MAC limiting is applied to Layer 2 interfaces by specifying the number of allowed MAC addresses to a specific Layer 2 interface, all Layer 2 interfaces, or a specific VLAN. By enabling MAC limiting you now have control on how to treat new MAC addresses being learned by the system. You can take the following action on any incoming packets with new MAC address information:

- *drop:* Drop the packet, but do not generate an alarm.
- *drop-and-log:* Drop the packet and generate an alarm, an SNMP trap, or system log entry.
- *log:* Do not drop the packet but generate an alarm, an SNMP trap, or a system log entry.
- *none:* Forward packets with new source MAC addresses, and learns the new source MAC address.
- *shutdown:* Disable the interface in the VLAN and generates an alarm, an SNMP trap, or a system log entry.
- *vlan-member-shutdown:* (EX9200 only) Starting in Junos OS release 15.1 for MAC limiting and MAC move limiting on EX9200 switches, the `vlan-member-shutdown` statement is supported to block an interface on the basis of its membership in a specific VLAN and generate an alarm, an SNMP trap, or a system log entry.

Let's take a look at configuring MAC limiting on an ELS device.

NOTE If you have static MAC addresses applied to a specific interface they do not count toward the MAC limit for dynamically learned addresses.

Before getting started let's log into an EX2300-C running ELS code and check what MAC addresses are being learned by viewing the `ethernet-switching` table:

```
EX2300-C# run show ethernet-switching table
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static, C - Control MAC
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)
Ethernet switching table : 11 entries, 11 learned
Routing instance : default-switch
    Vlan            MAC                 MAC       Age    Logical            NH        RTR
    name            address             flags            interface          Index     ID
    OOB             08:00:27:a4:0a:eb   D           -    ge-0/0/11.0        0         0
    OOB             2c:21:72:ce:8f:88   D           -    ge-0/0/11.0        0         0
    OOB             30:7c:5e:10:86:ff   D           -    ge-0/0/3.0         0         0
    OOB             5c:45:27:b1:72:3f   D           -    ge-0/0/5.0         0         0
    OOB             5c:45:27:e7:9b:3f   D           -    ge-0/0/0.0         0         0
    OOB             5c:45:27:e7:e8:3f   D           -    ge-0/0/4.0         0         0
    OOB             78:fe:3d:e4:01:bf   D           -    ge-0/0/2.0         0         0
    OOB             a8:20:66:27:06:19   D           -    ge-0/0/11.0        0         0
    OOB             cc:e1:7f:8f:7a:ff   D           -    ge-0/0/1.0         0         0
    OOB             d4:ae:52:ba:65:95   D           -    ge-0/0/11.0        0         0
    OOB             ec:b1:d7:4c:24:53   D           -    ge-0/0/11.0        0         0

EX2300-C # show interfaces ge-0/0/11 | display set
set interfaces ge-0/0/11 unit 0 family ethernet-switching vlan members 2
set interfaces ge-0/0/11 unit 0 family ethernet-switching storm-control default
```

There are a couple of rules when using MAC limiting on interfaces:

- You cannot enable MAC limiting on trunked interfaces.
- You cannot enable MAC limiting on an interface that is enabled for 802.1x.
- You cannot enable MAC limiting on an interface that is part of a Redundant Trunk Group (RTG).
- You cannot enable MAC limiting on an interface that has `no-mac-learning` configured.

You can see from the output of interface ge-0/0/11 that none of the rules are broken here and ge-0/0/11 is a candidate for MAC limiting. Right now, ge-0/0/11 has five MAC addresses associated with the interface. Let's limit the number of MACs to four to see how to apply it:

```
# set switch-options interface ge-0/0/11 interface-mac-limit 4 packet-action drop-and-log
# commit
configuration check succeeds
commit complete
```

NOTE When you commit the `mac-limit` configuration, it will clear all of the MAC addresses from the forwarding table associated with the interface.

As you can see we are under the `switch-options` hierarchy in Junos and set the specific interface to limit to four MAC addresses and to drop and log any additional packets containing new MAC addresses. Now let's look at the `ethernet-switching table` that we looked at previously:

```
# run show ethernet-switching table interface ge-0/0/11
MAC database for interface ge-0/0/11
MAC database for interface ge-0/0/11.0
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static, C - Control MAC
SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)
Ethernet switching table : 9 entries, 9 learned
Routing instance : default-switch
    Vlan                MAC                 MAC         Age    Logical                 NH        RTR
    name                address             flags              interface               Index     ID
    OOB                 08:00:27:a4:0a:eb   D             -    ge-0/0/11.0             0         0
    OOB                 2c:21:72:ce:8f:88   D             -    ge-0/0/11.0             0         0
    OOB                 a8:20:66:27:06:19   D             -    ge-0/0/11.0             0         0
    OOB                 ec:b1:d7:4c:24:53   D             -    ge-0/0/11.0             0         0
```

It's a quick look in the logs to see that we are limiting to four addresses on that interface:

```
May 20 19:13:02  switch1 l2ald[3080]: L2ALD_MAC_LIMIT_REACHED_IF: Limit on learned MAC addresses
reached for ge-0/0/11.0; current count is 4
```

Now let's do the same thing for the entire VLAN. Here is the configuration for the VLAN OOB (vlan-id 2):

```
# set vlans OOB vlan-id 2
# set vlans OOB l3-interface irb.2
# set vlans OOB switch-options interface-mac-limit 4
# set vlans OOB switch-options interface-mac-limit packet-action drop-and-log
```

Again, all of the MAC addresses on the VLAN will be cleared and it will go through the learn process when you apply the `interface-mac-limit`. Once it hits four MAC addresses you should see a log entry indicating the max limit has been reached and then if the threshold of four is crossed, you'll get a message stating that the `mac-limit` has been exceeded and the action of `dropped` has taken place.

```
May 20 21:15:31  switch1 l2ald[3080]: L2ALD_MAC_LIMIT_REACHED_IFBD: Limit on learned MAC addresses
reached for ge-0/0/11.0\OOB+2 flags [0x  3b] state [0x     0]; current count is 4
May 20 21:15:32  switch1 l2ald[3080]: L2ALD_MAC_LIMIT_EXCEEDED_IFBD: Limit on learned MAC addresses
exceeded for ge-0/0/11.0\OOB+2 flags [0x  3b] state [0x     0]; current count is 4 DROPPING THE PACKET
with mac address: d4:ae:52:ba:65:95
```

TIP By using the pipe command you can quickly and easily filter the dropped MAC addresses from the messages log. Simply enter `show log messages | match L2ALD | except Child` and you will get all of the MAC addresses that were dropped or logged and what actions were taken with the `ethernet-switching table`. If you are looking for a specific MAC, you can `| match` that MAC address, or if you are looking for a specific interface, you can locate that by appending `| match ge-0/0/11`, for instance.

To clear the actions that have been taken by MAC limiting actions, such as drop or shutdown, use the `clear ethernet-switching mac-learning-log` command:

```
> clear ethernet-switching mac-learning-log
May 20 21:37:46  switch1 l2ald[80]: L2ALD_MAC_LIMIT_RESET_IF: Resumed adding MAC addresses learned by
ge-0/0/11.0\OOB+2 flags[0x3b]state[0x0];current count is 3
```

MORE? For more information on MAC limiting on Junos please go to: https://www.juniper.net/documentation/en_US/junos/topics/concept/port-security-mac-limiting-and-mac-move-limiting.html.

With MAC limiting you can limit the number of MAC address that are allowed on a per interface basis or even a VLAN basis. The configuration for MAC limiting is performed under the `switch-options` hierarchy for interfaces and then under the `VLAN` hierarchy. There is also a `switch-options` section.

Configuring MAC Move Limiting

A MAC address move occurs when a previously learned MAC address shows up on a different interface. When `mac-move-limit` is configured it tracks these movements and can act on them accordingly. If the mac address moves more than the specified number of times, in 1 second, the packet takes the configured action:

- *drop:* (EX2300, 3400, 4300) Drop the packet, but do not generate an alarm.
- *drop-and-log:* (EX2300,3400 4300) Drop the packet and generate an alarm, an SNMP trap, or system log entry.
- *log:* (EX4300, EX9200) Do not drop the packet but generate an alarm, an SNMP trap, or a system log entry.
- *none:* (EX4300, EX9200) Forward packets with new source MAC addresses, and learn the new source MAC address.
- *shutdown:* Disable the interface in the VLAN and generate an alarm, an SNMP trap, or a system log entry.
- *vlan-member-shutdown:* (EX9200 only) Starting in Junos OS Release 15.1 for MAC Limiting and MAC Move Limiting on EX9200 Switches, the `vlan-member-shutdown` statement is supported to block an interface on the basis of its membership in a specific VLAN and generate an alarm, an SNMP trap, or a system log entry.

The following command initiates MAC move limitations on a VLAN:

```
# set vlans OOB switch-options mac-move-limit 4 packet-action log
```

Once the limit of four is crossed it will generate a log entry for the offending MAC. You can also perform the other actions such as `drop-and-log`, `drop` or `shutdown` but be careful when you are placing this on a VLAN.

Configuring Persistent MAC

Persistent MAC or "Sticky MAC" allows the switch to retain the dynamically learned MAC address through switch reboots. This port security feature is disabled by default so it requires explicit configuration. Persistent MAC learning prevents unauthorized devices from connecting to the interface and it also prevents data loss because devices do not have to go through the learning process when attached if they have a persistent MAC assigned to that interface. Combining Persistent MAC with MAC Limiting helps insure against new or unknown devices being able to connect to the network.

```
# set switch-options interface ge-0/0/4 persistent-learning
# run show ethernet-switching mac-learning-log | match ge-0/0/4
Sat May 20 22:44:25 2017 vlan_name OOB+2 mac 5c:45:27:e7:e8:3f was deleted from ge-0/0/4.0 with flags:
0x1080
Sat May 20 22:44:25 2017 vlan_name OOB+2 mac 5c:45:27:e7:e8:3f was learned on ge-0/0/4.0 with flags:
0x2001f
Sun May 21 00:44:12 2017 vlan_name OOB+2 mac 5c:45:27:e7:e8:3f was changed on ge-0/0/4.0 with flags:
0x2103f
```

Once `persistent-learning` is enabled on an interface it will maintain that MAC address. The output from `show ethernet-switching mac-learning-log` is very valuable as you can match on a specific interface and see what MAC addresses have been assigned and any actions taken.

Again, combining `persistent-learning` with `interface-mac-limit` provides a good security posture for dynamically learning MAC addresses and limiting the number of persistent mac addresses on an interface. For more information on persistent learning see the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/concept/port-security-persistent-mac-learning.html.

If you want to move a device from a port with persistent learning use the

`clear ethernet-switching persistent-mac {interface | mac-address}` command to clear the MAC entry and then move the device to another port.

Configuring DHCP local server

Providing Dynamic Host Connection Protocol (DHCP) service from the local switch is a must have. It has been a go-to feature for quite some time. So just imagine configuring a local DHCP server on any of the new ELS software devices, just like you have done a thousand time before, and getting this awesome message when you start troubleshooting the reason it doesn't work.

The switch will still allow you to type it in using the old syntax and will even commit without any complaint:

```
{master:0}[edit]
EX2300# set system services dhcp pool 172.16.30.0/24 address-range low 172.16.30.2
{master:0}[edit]
EX2300# set system services dhcp pool 172.16.30.0/24 address-range high 172.16.30.10
```

```
{master:0}[edit]
EX2300# set system services dhcp pool 172.16.30.0/24 domain-name adroitnetworking.com
{master:0}[edit]
EX2300# set system services dhcp pool 172.16.30.0/24 router 172.16.30.1
{master:0}[edit]
EX2300# commit
configuration check succeeds
commit complete
```

However, it will not provide your client with a DHCP address and when you go looking for the reason you will find it right in the configuration.

```
EX2300# show system services dhcp
##
## Warning: configuration block ignored: unsupported platform (ex2300-c-12p)
##
## Warning: Incompatible with 'system services dhcp-local-server group'
##
pool 172.16.30.0/24 {
    address-range low 172.16.30.2 high 172.16.30.10;
    domain-name five-nines.com;
    router {
        172.16.30.1;
    }
}
```

The old configuration will not work on the newer ELS software. You can still configure a local DHCP server on your switch you just have to do it using the access hierarchy.

The new configuration method is performed by using the access hierarchy. Let's configure VLAN V16 with an Integrated Routing and Bridging (IRB) interface of irb.16 on the 172.16.30.0 network and provide DHCP services on that VLAN. To configure the VLAN:

```
set vlans V16 vlan-id 16
set vlans V16 l3-interface irb.16
```

To configure the irb interface:

```
set interfaces irb unit 16 family inet address 172.16.30.1/24
```

To configure the dhcp-local-server group:

```
set system services dhcp-local-server group V16 interface irb.16
```

Now, add the vlan to you access interfaces:

```
set interfaces ge-0/0/6 unit 0 family ethernet-switching vlan members V16
set interfaces ge-0/0/8 unit 0 family ethernet-switching vlan members V16
```

Use the access hierarchy to define your DHCP attributes:

```
set access address-assignment pool V16 family inet network 172.16.30.0/24
set access address-assignment pool V16 family inet range V16-RANGE low 172.16.30.10
set access address-assignment pool V16 family inet range V16-RANGE high 172.16.30.40
```

```
set access address-assignment pool V16 family inet dhcp-attributes maximum-lease-time 36000
set access address-assignment pool V16 family inet dhcp-attributes server-identifier 172.16.30.1
set access address-assignment pool V16 family inet dhcp-attributes name-server 8.8.8.8
set access address-assignment pool V16 family inet dhcp-attributes router 172.16.30.1
```

If you are just standing the service up for the first time it's also a good idea to add `traceoptions` for troubleshooting.

```
set system processes dhcp-service traceoptions file dhcp_logfile
set system processes dhcp-service traceoptions file size 2m
set system processes dhcp-service traceoptions level all
set system processes dhcp-service traceoptions flag all
```

TIP Traceoptions can also be placed into specific groups and then you can activate them and deactivate them using the `# set apply-groups` command when you need to troubleshoot something. As a best practice, traceoptions should not be left on and constantly running.

Then once you commit the new configuration and connect a DHCP client to the interface you should be able to see the bindings:

```
EX2300# run show dhcp server binding
IP address        Session Id  Hardware address   Expires     State      Interface
172.16.30.10      1           d4:ae:52:ba:65:97  34370       BOUND      irb.16
```

As well as the DHCP statistics:

```
EX2300# run show dhcp server statistics
Packets dropped:
    Total                     0
Messages received:
    BOOTREQUEST               2
    DHCPDECLINE               0
    DHCPDISCOVER              1
    DHCPINFORM                0
    DHCPRELEASE               0
    DHCPREQUEST               1
    DHCPLEASEQUERY            0
    DHCPBULKLEASEQUERY        0
Messages sent:
    BOOTREPLY                 2
    DHCPOFFER                 1
    DHCPACK                   1
    DHCPNAK                   0
    DHCPFORCERENEW            0
    DHCPLEASEUNASSIGNED       0
    DHCPLEASEUNKNOWN          0
    DHCPLEASEACTIVE           0
    DHCPLEASEQUERYDONE        0
```

As network engineers, you always have to be agents-of-change. It may not make sense to us why a perfectly good configuration hierarchy has changed, but in the end, there is always a reason. After looking at the documentation you can see that

the software engineers have added a ton of features and flexibility to DHCP provided by the local switch. For more information on the features and configuring the new ELS DHCP server please go to the TechLibrary: http://www.juniper.net/documentation/en_US/junos/topics/concept/services-dhcp-overview.html.

Configuring DHCP Snooping and Dynamic Arp Inspection (DAI)

Dynamic Host Connection Protocol (DHCP) snooping provides additional security by monitoring downstream DHCP client IP addresses provided by a trusted DHCP server, and maintaining a database. The switch then compares the untrusted IP address from the client to the trusted database, and if verified, they are allowed access to the network. All trunks on the switch are trusted, by default, whereas all access ports are untrusted.

DHCP snooping is disabled by default. To enable the DHCP snooping process and begin building and maintaining the database you have to enable it.

You *cannot* enable DHCP snooping on its own – enabling any other DHCP security feature automatically enables it on specific VLANs. In a way, it forces you into increased security by having to enable other DHCP security features. For our network, let's enable `arp-inspection` to start DHCP snooping.

The lab's EX4600-4 has two directly connected VLANs: V100 and V200. Let's configure them both for basic Dynamic ARP Inspection (DAI) and see how the DHCP binding database is built:

```
EX4600-4_5-VC# show vlans
V100 {
    vlan-id 100;
    l3-interface irb.100;
}
V200 {
    vlan-id 200;
    l3-interface irb.200;
}
EX4600-4_5-VC# set vlans V100 forwarding-options dhcp-security arp-inspection
EX4600-4_5-VC# set vlans V200 forwarding-options dhcp-security arp-inspection
EX4600-4_5-VC# commit
fpc0:
configuration check succeeds
fpc1:
commit complete
fpc0:
commit complete
```

MORE? If you need more about DHCP Snooping please refer to these TechLibrary topics: https://www.juniper.net/documentation/en_US/junos/topics/concept/port-security-dhcp-snooping-els.html.

```
EX4600-4_5-VC> show dhcp-security arp inspection statistics
ARP inspection statistics:
Interface        Packets received  ARP inspection pass   ARP inspection failed
ge-0/0/1.0                     7                    5                      2
ge-0/0/2.0                    10                   10                      0
ge-0/0/3.0                    12                   12                      0
```

This is kind of an odd relationship between DHCP snooping and DAI. You enable DHCP snooping by enabling DAI but DAI relies on the DHCP binding database to compare to IP packets that it inspects on the LAN to validate those Address Resolution Protocol (ARP) packets and protect against ARP cache poisoning. As you can see, it's a very symbiotic relationship between DAI and DHCP snooping.

Configuring IP Source Guard

IP Source Guard is used to prevent spoofing of source IP addresses and source MAC addresses. Again, the DHCP snooping database is consulted by IP Source Guard so you can see why enabling any other feature under the `vlan forwarding-options` stanza enables the DHCP snooping binding database.

If IP Source Guard detects an invalid source IP or MAC address, it will discard that packet. IP Source Guard is VLAN specific so you have to explicitly enable it across all VLANs you want to protect.

CAUTION As of this writing the only device that supports IP Source Guard in the lab is the EX4300-6. The EX2300, EX3400, EX4600, and QFX5100 *do not* support the `ip-source-guard` command, but check at www.juniper.net for updates at the time you are reading this.

```
EX4300-6_7-VC# show vlans
V100 {
    vlan-id 100;
    forwarding-options {
        dhcp-security {
            arp-inspection;
            ip-source-guard;
        }
    }
}
```

Okay, the DHCP snooping database is enabled, DAI is enabled, and now IP Source Guard. To check the status of the database, use the following command:

```
EX4300-6_7-VC> show dhcp-security binding ip-source-guard
IP Address    MAC Address              Vlan      Expires        State      Interface
10.0.2.17     00:05:85:3A:83:77        V100      86265      BOUND      ge-0/0/1.0
10.0.2.18     00:05:85:3A:83:79        V100      86265      BOUND      ge-0/0/1.0
10.0.2.19     00:05:85:3A:83:80        V100      86287      BOUND      ge-0/0/2.0
10.0.2.20     00:05:85:3A:83:81        V100      86287      BOUND      ge-0/0/2.0
10.0.2.21     00:05:85:3A:83:83        V100      86287      BOUND      ge-0/0/2.0
```

MORE? For more information on IP Source Guard and DAI please go to: https://www.juniper.net/documentation/en_US/junos/topics/example/port-security-protect-from-spoofing-els.html#jd0e321.

So, we enabled `ip-source-guard` and verified that it is working and we discussed the reliance on the DHCP snooping database and how enabling `ip-source-guard` will enable the DHCP snooping database.

Configuring 802.1X Network Access Control (NAC)

802.1x provides port-based Network Access Control for the Local Area Network (LAN) as well as Wireless-LAN (WLAN). The Extensible Authentication Protocol (EAP) is carried over the LAN and is also known as EAPoL or Extensible Authentication Protocol over LAN.

There are three distinct parts to 802.1x: the Authentication Server (aka RADIUS), the Client (aka Supplicant), and the Authenticator (aka network device). Figure 11.1 depicts the basic components of the 802.1x system for this chapter section.

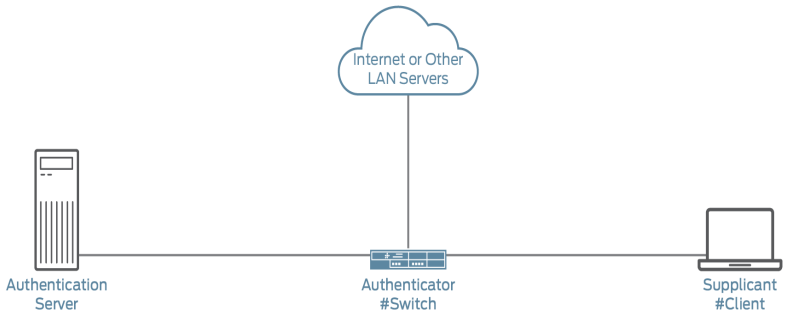


Figure 11.1 *802.1x Lab Diagram.*

As mentioned previously, RADIUS is the go-to server for EAP. For the 802.1x server in this lab let's use a Pulse Policy Secure Virtualized Appliance (thanks to Pulse Secure for providing these and permission to use their screen captures).

NOTE If you are looking for an 802.1x NAC solution check out Pulse Secure at: https://www.pulsesecure.net. It's an awesome platform that is vendor agnostic for mixed environments.

The Pulse Policy Secure (PPS) just like the MAG has an internal port and an external port. The external port is typically used to manage the appliance itself while the internal port is used to provide 802.1x EAPoL authorization to the network. In this deployment, as a Virtual Machine (VM), we're only using the internal interface.

Figure 11.2 *Pulse Secure Login Page.*

The EX Series switch (or any other Junos device) acts as a proxy for the client when it performs an Identity Request to attach to the network using EAPoL. The switch acts as a proxy to relay that request to the RADIUS server for authentication, by extracting the information from the EAPoL, and then relaying it as a RADIUS authentication request (all traffic remains blocked until the RADIUS server verifies the supplicant information or MAC address). The switch (aka Authenticator) does not evaluate the EAPoL request it simply extracts the relevant information and then forwards that request to the Authentication Server. Going back to our simple diagram, let's add the steps for a successful Supplicant connection through the Authenticator and from the Authentication Server, as shown in Figure 11.3.

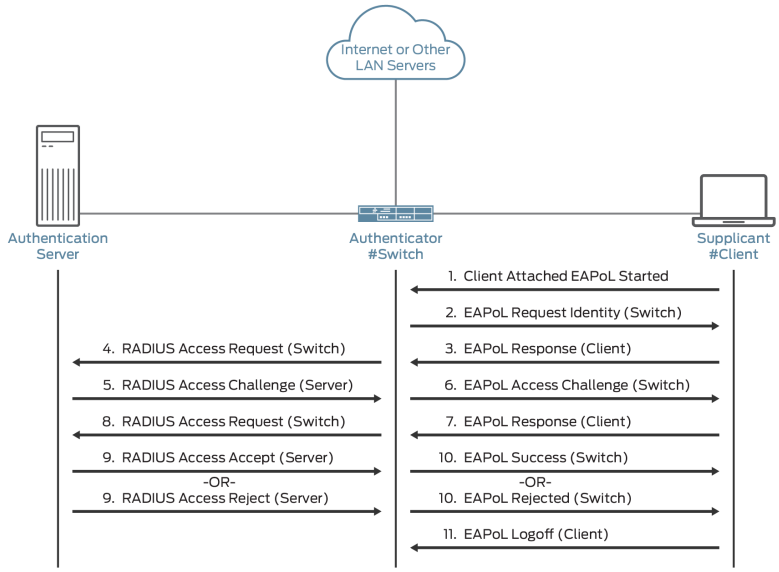


Figure 11.3 *RADIUS Authentication Steps.*

There are three types of supplicant modes supported by EX Series switches: Single 2, Single-Secure 3, and Multiple. Let's discuss each.

The single mode (which is the default on EX Switches) permits access for the first supplicant and then any other supplicants connected to the same port are allowed full access without authenticating to the 802.1X Network Access Controller (NAC). Think of this like a dumb hub with a bunch of devices on it connected to our 802.1x-enabled Junos device. If one of the devices on the hub authenticates, then the other devices will be allowed access through that same port without having to authenticate.

TIP If you have a shared VLAN with a phone and computer and one of the devices *does not* support a supplicant, then you would want to use the Single Supplicant mode.

The Single-Secured mode permits access for a single supplicant and then any other supplicant on that interface attempting to connect is denied access.

The Multiple Supplicant mode permits access for, you guessed it, multiple supplicants. This is used for Voice over IP (VoIP) and data VLANs that share the same interface. You will see this a lot when you have a VoIP phone with a supplicant and then a PC hanging off the data port on the phone to provide access to the network for a computer.

TIP If you have a shared VLAN with a VoIP phone and PC, and they both support supplicants, you want to use the Multiple Supplicant mode as it is more secure and forces both devices to authenticate.

NOTE EX Series switches have a default re-authentication of 3600 seconds (or 1 hour). You can disable re-authentication or modify the settings under the `protocols dot1x authenticator` hierarchy.

Another feature of 802.1X is the ability to assign VLANs to users based on their supplicant within the RADIUS access-accept message (Step 9 from our previous process outlined). In addition, you can also dynamically apply firewall filters upon successful authentication. These features are all made possible through the use of Vendor Specific Attributes (VSAs), as discussed in the RADIUS section, earlier in this chapter.

Before configuring our Pulse Policy Secure (PPS) virtual appliance you need to identify a few things (that whole planning thing again):

1. The IP Address of the PPS
2. The Authentication Key (used between authenticator and authentication server)
3. Users (lab1, lab2, lab3)
4. VLANs (Guest, Remediation, Enterprise, SSL_External)

The Guest VLAN provides basic Internet services for corporate guest's who may just be visiting, and for devices that fail authentication. When the switch receives an access-reject from the RADIUS server it can make the determination of placing the device in the Guest VLAN by checking to see if the Guest VLAN is configured.

Once the Pulse Policy Secure Virtual Machine (VM) is in your hypervisor, you need to launch it and provide the IP address of the VM. For this section we're using a personal lab (so we can connect and control the laptops with supplicants.) The following diagram shows the setup:

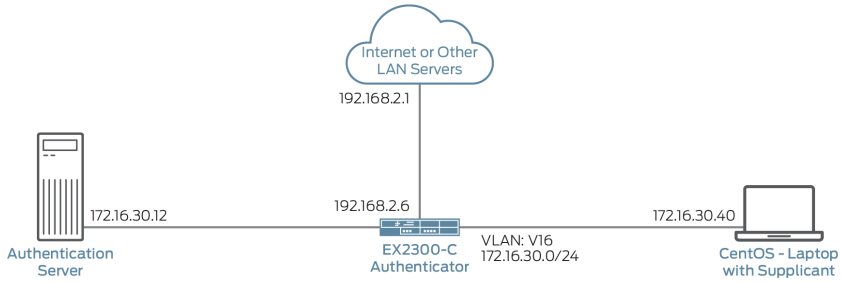


Figure 11.4 *Supplicant Lab*

Once the physical layer and IP addressing are in place you can configure the PPS. Once the VM is launched, it provides a menu (Figure 11.5) to perform the basic configuration to access the web GUI.

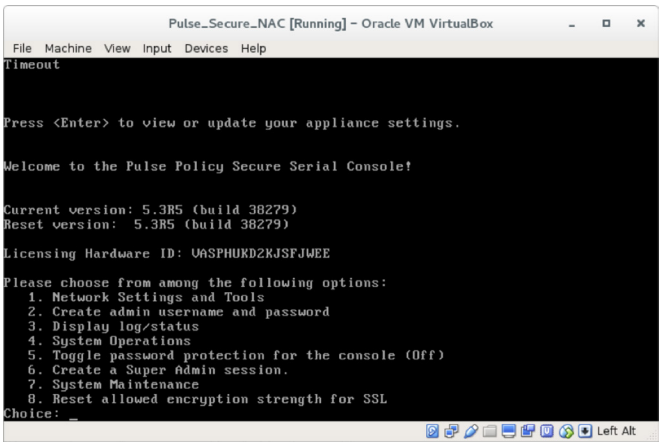


Figure 11.5 *Pulse Secure – Pulse Policy Server (PPS) Menu*

The options shown in Figure 11.5 that we are concerned about are options 1, 2, and 6.

TIP The Pulse Policy Secure Virtual Machine Initial Setup and Configuration guide will provide you with all the required steps to get this far.

Once you select option 1, Network Settings and Tools, you will enter the information for your VM. Figure 11.6 is the result for the PPS setup.

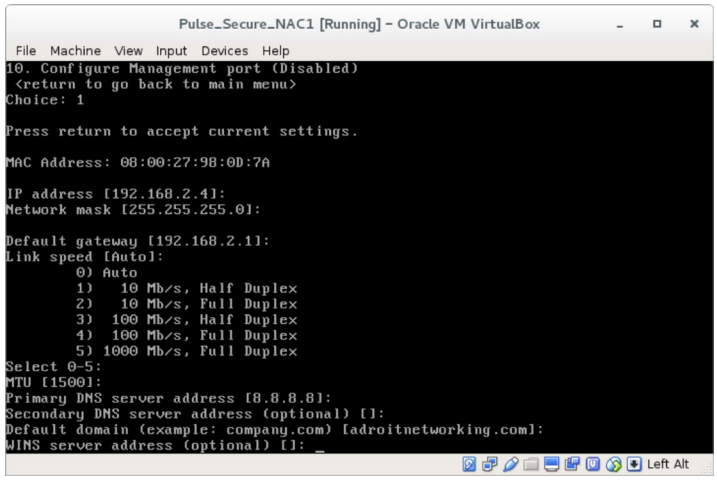


Figure 11.6 *Pulse Policy Secure (PPS) Configuration Menu*

Once you have finished the initial configuration of the VM you can create a super admin session and it will allow you to log into the web GUI to configure the rest of the platform. To reach an Administration session you have to use the /admin URL otherwise it will send you to a user session, e.g., https://192.168.2.4/admin.

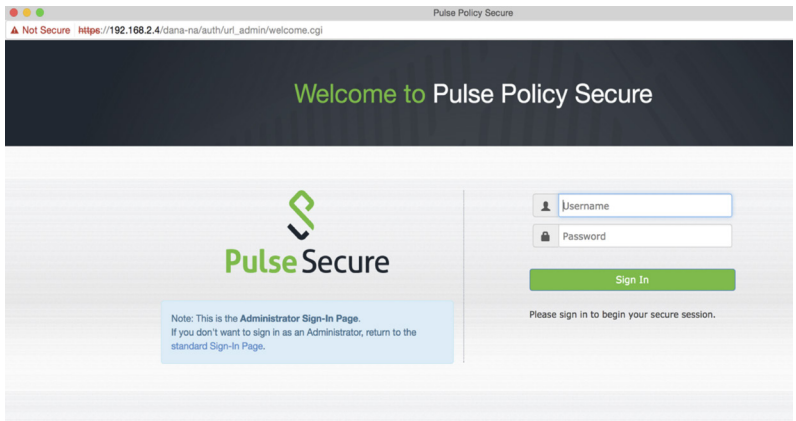


Figure 11.7 *Pulse Secure – PPS Administrator Login Page.*

Note that on the left panel of Figure 11.7 it states you are at the Administration Sign-in Page. When you log into a new device it will take you to the setup page and walk you through setting up the license and the SSL certificate. Once you have completed the initial licensing and setup, it will allow you to configure RADIUS and other features on the VM. Figure 11.8 is a capture of the system status overview showing the license, number of concurrent users, hits, and more.

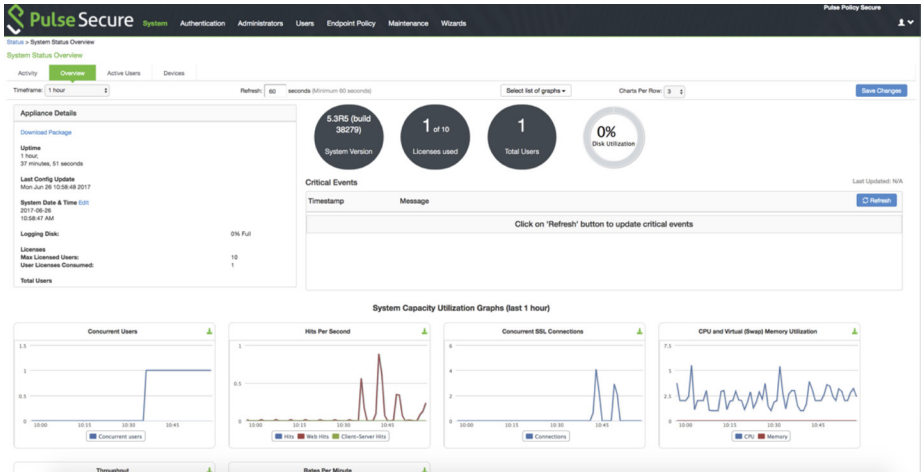


Figure 11.8 *Pulse Secure System Status Overview.*

You have to configure a few things in the PPS VM that include the 802.1x users, MAC Authentication Bypass (MAB), and our devices. Please follow the Pulse Secure deployment guide to configure any of the different areas that are discussed.

NOTE There is no way to do justice to the Pulse Secure product by trying to capture all of the screens and settings for the PPS VM, and besides, it's out of the scope of this book. They have their own technical guides. The goal here is to introduce you to Pulse Secure and the fact that you are using it for an 802.1X NAC solution. Reach out to Pulse Secure via: http://www.pulsesecure.com.

Once you have Pulse Secure set up you can start configuring the Authenticator (aka, EX Series switch). This is where the rubber meets the road as far as configuring the EX Series switch for 802.1X. First, you need to configure the RADIUS server information and this is managed under the `access` hierarchy of the configuration.

```
# set access radius-server 172.16.30.12 port 1812
# set access radius-server 172.16.30.12 secret "$9$fQ3/uORlK8CtK8X7Tz3Ct0BIcre"
# set access radius-server 172.16.30.12 timeout 5
# set access radius-server 172.16.30.12 retry 3
```

Now that the device is pointed to the appropriate RADIUS server for Network Access Control (NAC), you need to set an `access profile`:

```
# set access profile juniper-access-profile authentication-order radius
# set access profile juniper-access-profile radius authentication-server 172.16.30.12
# set access profile juniper-access-profile radius accounting-server 172.16.30.12
# set access profile juniper-access-profile accounting order radius
# set access profile juniper-access-profile accounting accounting-stop-on-failure
# set access profile juniper-access-profile accounting accounting-stop-on-access-deny
```

TIP You can have numerous access profiles if you have more than one RADIUS server. This gives you the flexibility to point different sets of users to different profiles and or servers. More information on setting access profiles is available at the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/radius-server-ex-series-cli.html.

Once you have the `radius-server` and the `profile` configured under the `access` hierarchy you can move on to the dot1x configuration. Our CentOS laptop, as seen in the diagram a few pages ago, is on interface ge-0/0/6 so we need to configure that specific interface for dot1x:

(1)

```
# set protocols dot1x authenticator authentication-profile-name juniper-access-profile
```

(2)

```
# set protocols dot1x authenticator interface ge-0/0/6.0 supplicant single
# set protocols dot1x authenticator interface ge-0/0/6.0 retries 1
# set protocols dot1x authenticator interface ge-0/0/6.0 transmit-period 10
# set protocols dot1x authenticator interface ge-0/0/6.0 reauthentication 3600
# set protocols dot1x authenticator interface ge-0/0/6.0 supplicant-timeout 30
# set protocols dot1x authenticator interface ge-0/0/6.0 server-timeout 60
# set protocols dot1x authenticator interface ge-0/0/6.0 maximum-requests 3
```

(3)

```
# set protocols dot1x authenticator interface ge-0/0/6.0 guest-vlan GUEST
```

(4)

```
# set protocols dot1x authenticator interface ge-0/0/6.0 server-reject-vlan REMEDIATION
```

(5)

```
# set protocols dot1x authenticator interface ge-0/0/6.0 server-fail vlan-name BASIC_INTERNET
```

Let's go through the different options and why we have chosen these specific configurations using the numbered sections.

1. Notice `juniper-access-profile` at the end of this statement. This is how you can control which server you are sending the dot1x queries to per interface.

2. We are only using a single supplicant with this host. Remember there are three (3) types of supplicants: Single, Single-Secure, and Multiple.

3. The guest VLAN needs to be identified. This is not a working VLAN if you look at the context sensitive help you can see that this VLAN is for "unauthenticated or non-responsive hosts".

4.The `server-reject-vlan` is different than the guest VLAN. This VLAN is for clients that actually reach the RADIUS server but are rejected due to some issue or just not authorized. In our case, we are going to send them to a Remediation VLAN that has access to things like virus scan updates, browser updates, security patches, or just needs to be checked out by a technician.

5. If the server fails or is unreachable then we are going to send our user to the BASIC_INTERNET VLAN that only has access to basic Internet functions.

The rest of the configuration should be self-explanatory. Now, as a best practice, when setting up the first dot1x configuration on a device, it's recommend to place a traceoptions configuration to capture the session. This example uses a group (like discussed before) to make it easy to apply or remove it from the configuration:

```
# set groups DOT1X_TRACE protocols dot1x traceoptions file DOT1X
# set groups DOT1X_TRACE protocols dot1x traceoptions file size 1m
# set groups DOT1X_TRACE protocols dot1x traceoptions file files 3
# set groups DOT1X_TRACE protocols dot1x traceoptions flag state
# set groups DOT1X_TRACE protocols dot1x traceoptions flag dot1x-debug
# set groups DOT1X_TRACE protocols dot1x traceoptions flag eapol
```

And now apply the configuration use:

```
# set apply-groups DOT1X_TRACE
# commit and-quit
```

To view the logs all you have to do is `show log DOT1X` or you can use `monitor start DOT1X` and watch as you connect an EAPoL device. That leads us to the next part of the 802.1x configuration, that of the end client. Most Windows, Macs, and Linux devices come with a supplicant already installed. If your device doesn't have a supplicant or you need help configuring the supplicant, please refer to your manufacturer documentation as it is out of the scope of this book. We will however briefly cover the supplicant configuration for the CentOS laptop to connect to our lab.

Under Applications > System Settings > Settings you will find the Network Icon. Click on Network and it will take you to the screen shown in Figure 11.9.

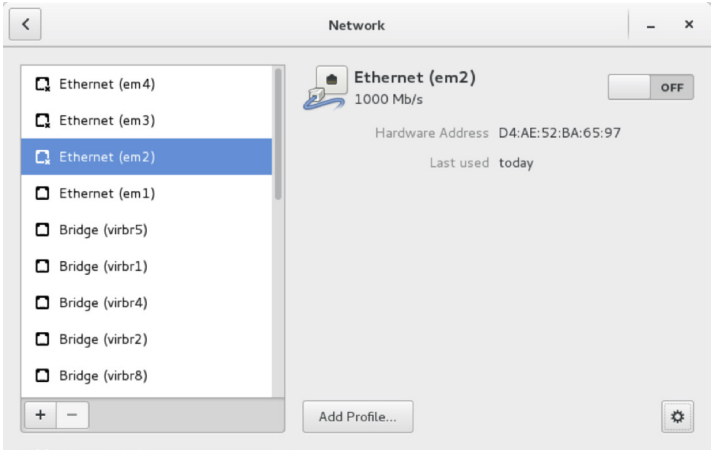


Figure 11.9 *CentOS Network Tool*

We are configuring Ethernet interface em2 in this case. You can see that the current status is OFF. On the bottom right corner there is a settings icon to click to get inside the network configuration for this interface. Once clicked you'll see Figure 11.10.

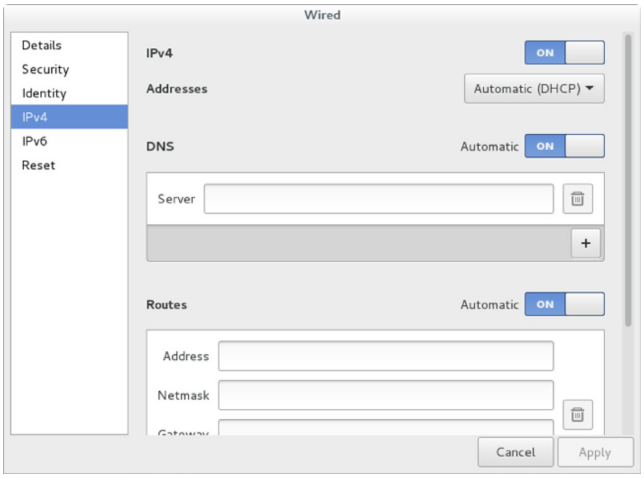


Figure 11.10 *CentOS IPv4 Configuration Interface*

You can see that IPv4 is turned ON as well as DNS and Routes. The address (if properly authenticated) will be learned via DHCP and be placed on the V16 network. The next concern is Security in Figure 11.11.

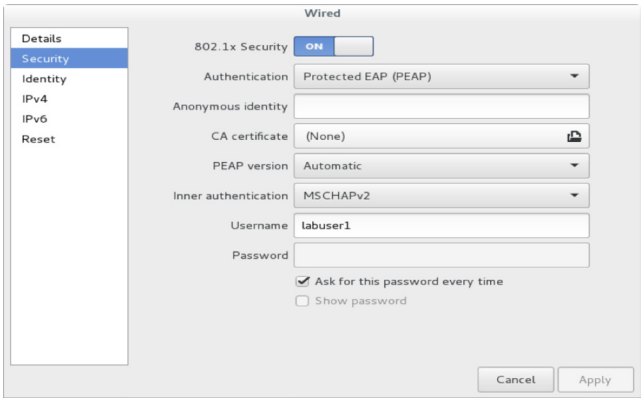


Figure 11.11 *CentOS Security/Supplicant Configuration.*

As you can see here we have chosen Protected EAP (PEAP) with automatic version and MSCHAPv2 as the inner authentication. We have also entered the username *labuser1* but indicated to be prompted for the password every time we access the network.

Once this has been applied you are taken back out to the network screen where you need to turn your interface to the ON position. As soon as you do this you should be prompted for the 802.1x password as shown in Figure 11.12.

Figure 11.12 *CentOS Password prompt for Supplicant*

And once you have been authenticated your network window you will have the DHCP address gateway and DNS updated as in Figure 11.13 (remember we did the DHCP configuration for ELS in an earlier chapter).

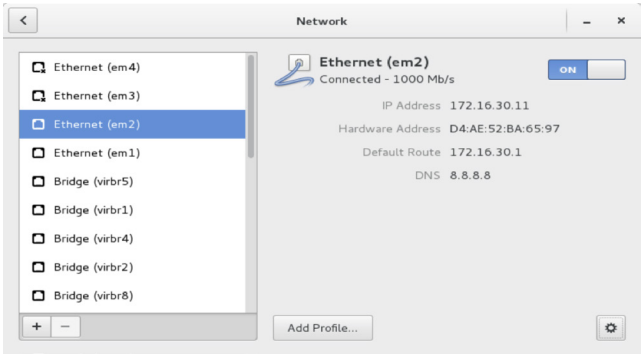


Figure 11.13 *CentOS Network Interface*

That was a pretty simple configuration, right? We utilized our Pulse Policy Secure server to set up our 802.1x users and our RADIUS profile. Then we configured our switch with the RADIUS information and profile. Next we configured the dot1x configuration and told the switch to put us in various VLANs based on the authentication action of `authenticated`, `server-failed` or `server-rejected`. This can give you a lot of flexibility if you need to have devices remediated before they are allowed on the network.

Let's take a look at the actual interface to see what it says about 802.1x:

```
# run show dot1x interface ge-0/0/6.0
802.1X Information:
Interface     Role           State           MAC address          User
ge-0/0/6.0    Authenticator  Authenticated   D4:AE:52:BA:65:97    labuser1
```

And let's not forget our DOT1X log. Let's take a look at the end of that file and see if it says `Authenticated`.

```
# run show log DOT1X | last
Jun 26 12:48:56.589975 Queuing EAPOL frame to be transmitted out on interface ge-0/0/6
Jun 26 12:48:56.590368  EAP Frame Sent with code:  3 !!!
Jun 26 12:48:56.590448  BSM moved to state: SUCCESS !!
Jun 26 12:48:56.590499  ASM Called with Event: BKEND_AUTHSUCCESS, and State: Authenticating
Jun 26 12:48:56.590547  for Port: 561, MAC: d4ae52ba - 6597
Jun 26 12:48:56.590589  Id: 20, SessionNode: 11a4000
Jun 26 12:48:56.590649  TMR: Reauth When Timer Started for port:561, Duration: 3600 !!
Jun 26 12:48:56.590707  TMR: Timer is started
Jun 26 12:48:56.590753  ASM moved to state: AUTHENTICATED from Authenticating!!
Jun 26 12:48:56.590793  ASM moved to state: AUTHENTICATED !!
Jun 26 12:48:56.590832  authenticatedStateCause 3
```

This log is very chatty because we have `dot1x-debug` flag in use so you will get a lot of information especially if you have numerous interfaces authenticating. There are numerous `dot1x` operational commands that are useful in troubleshooting:

```
> show dot1x statistics interface ge-0/0/6
Interface: ge-0/0/6.0
  TxReqId = 268 TxReq = 2318 TxTotal = 2586
  RxStart = 6 RxLogoff = 0 RxRespId = 259 RxResp = 2318
  CoA-Request = 0 CoA-Ack = 0 CoA-Nak = 0
  RxInvalid = 0 RxLenErr = 0 RxTotal = 2583
  LastRxVersion = 1 LastRxSrcMac = d4:ae:52:ba:65:97

> show dot1x accounting-attributes
User: labuser1        MAC Address: d4:ae:52:ba:65:97
Accounting Attribute:
    NAS port (5)                :561
    NAS port ID (87)            :ge-0/0/6.0
    Called station Id (30)      :28-a2-4b-88-21-5d
    Calling Station Id (31)     :d4-ae-52-ba-65-97
    Framed Mtu (12)             :1514
    Session Timeout (27)        :3600
    Acct Authentic (45)         :radius
    NAS Identifier (32)         :EX2300
    Acct Status (40)            :START
    Acct Session Id (44)        :802.1x8152010100062bf7
```

You should have successfully configured dot1x on your EX Series switch and followed the discussion on all the numerous parts of the authentication process including EAPoL, supplicants, authenticator, and authentication server. The next thing we need to do is be able to bypass authentication based on MAC addresses.

Configuring MAC Radius

MAC RADIUS uses the RADIUS server to authenticate based on the MAC address of the host. If both MAC RADIUS and 802.1X authentication are enabled on the interface, the switch first sends the host three EAPoL requests. If there is no response from the host, the switch sends the host's MAC address to the RADIUS server to check whether it is a permitted MAC address. If the MAC address is configured as permitted on the RADIUS server, the RADIUS server sends a message to the switch that the MAC address is a permitted address, and the switch opens LAN access to the nonresponsive host on the interface to which it is connected. For further information on MAC-RADIUS please refer to these excellent resources in Juniper's TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/authentication-mac-radius-ex-series-cli.html.

You can see in Figure 11.14 that the MAC address has been added to the Pulse Policy Secure server and we have saved the change.

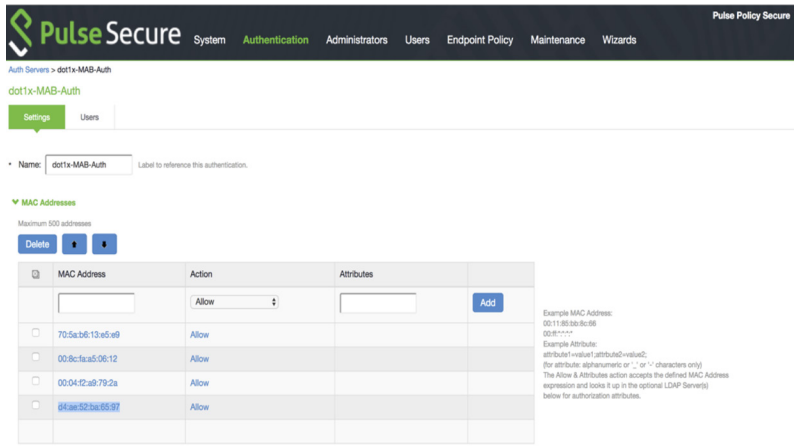


Figure 11.14 *Pulse Secure – Dot1X MAC Authentication Bypass (MAB)*

Now that the MAC is in the database you should be able to configure for a successful connection using only the MAC address. As mentioned above, if you do not use the `restrict` keyword with `mac-radius` the host will send three EAPoL polls before the MAC authentication kicks in. By using the `restrict` keyword you are completely bypassing the supplicant and only using the MAC address, which is faster because it bypasses the EAPoL timeout:

```
# run clear ethernet-switching table interface ge-0/0/6.0
# run clear dot1x interface ge-0/0/6.0
# run show dot1x interface ge-0/0/6.0
802.1X Information:
Interface     Role           State           MAC address          User
ge-0/0/6.0    Authenticator  Connecting
```

In addition, the interface on the host has been reset to clear out any information. 802.1x is also disabled on the host side. Now let's take a look at the `mac-radius` configuration. First, we need our access method:

```
# set access profile mac-radius authentication-order none
# set access profile mac-radius radius-server 172.16.30.12 secret "Juniper123"
```

Let's identify the dot1x profile for `mac-radius`:

```
# set protocols dot1x authenticator authentication-profile-name mac-radius
# set protocols dot1x authenticator interface ge-0/0/6.0 supplicant single
# set protocols dot1x authenticator interface ge-0/0/6.0 mac-radius restrict
```

And once commited, enable the interface on our CentOS host and see if it is authenticating to the RADIUS server using only the MAC address. As soon as the interface is enabled, you'll start getting information from the monitor on the DOT1X traceoptions file:

```
Jun 26 16:04:42.349238 l2ald: received mac learn req for mac: d4:ae:52:ba:65:97 vlan:0 bd_idx:8
Jun 26 16:04:42.349390 l2d query for mac d4:ae:52:ba:65:97 vlan id 16
Jun 26 16:04:42.349449 Mac address d4:ae:52:ba:65:97  on interface ge-
0/0/6.0 NOT found in Static list
Jun 26 16:04:42.349488 pnac_ng_is_native_ifbd_marked_for_deletion:202 native vlan of port 561 NOT-
MARKED for deletion
Jun 26 16:04:42.349539 ASM CONNECTING : Intf ge-0/0/6.0: ReqId Count 0 Reauth Count 0
Jun 26 16:04:42.349590  ASM moved to state: CONNECTING !!
Jun 26 16:04:42.349642 Session d4:ae:52:ba:65:97 Authentication mode: None
Jun 26 16:04:42.349687 Initiate Mac radius. Authentication mode: None
Jun 26 16:04:42.349729 ASM CONNECTING: ONLY-MAC-MODE init on intf ge-0/0/6.0
Jun 26 16:04:42.349771 PnacAuthAsmInitMacRadius Intf ge-0/0/6.0: Launch MACAUTH.
Jun 26 16:04:42.349827  ASM Called with Event: RXRESPID, and State: Connecting
Jun 26 16:04:42.349872  for Port: 561, MAC: d4ae52ba - 6597
Jun 26 16:04:42.349912  Id: 0, SessionNode: 1096000
Jun 26 16:04:42.349951 ASM: Inside PnacAuthAsmRxrespConnecting
Jun 26 16:04:42.349999  ASM moved to state: AUTHENTICATING !!
```

And then the interface is configured on the host. You can also see from the `show dot1x interface` command that we are indeed only using MAC-RADIUS authentication:

```
# run show dot1x interface ge-0/0/6.0 detail
ge-0/0/6.0
  Role: Authenticator
  Administrative state: Auto
  Supplicant mode: Single
  Number of retries: 3
  Quiet period: 60 seconds
  Transmit period: 30 seconds
  Mac Radius: Enabled
```

```
  Mac Radius Restrict: Enabled
  Mac Radius Authentication Protocol: EAP-MD5
  Reauthentication: Enabled
  Reauthentication interval: 3600 seconds
  Supplicant timeout: 30 seconds
  Server timeout: 30 seconds
  Maximum EAPOL requests: 2
  Guest VLAN member: not configured
  Number of connected supplicants: 1
    Supplicant: d4ae52ba6597, D4:AE:52:BA:65:97
      Operational state: Authenticated
      Backend Authentication state: Idle
      Authentication method: Mac Radius
      Authenticated VLAN: V16
      Session Reauth interval: 3600 seconds
      Reauthentication due in 3358 seconds
```

MAC RADIUS is different than MAB. With MAC RADIUS the MAC address is stored on the RADIUS server and the switch polls the radius server to validate the MAC address. Which is an excellent segue to the next section.

Configuring MAC Authentication Bypass (MAB)

MAB is used to allow network access based on the MAC address of the connected host. For instance, if you have a printer that does not have a supplicant you can allow that printer access by identifying the MAC address to the Authentication Server.

For this section, let's utilize the MAC address of the CentOS device but this time only use MAC Authentication Bypass and no supplicant. First, let's identify the MAC address of our device on ge-0/0/6.0:

```
> show ethernet-switching table | match ge-0/0/6.0
    V16                 d4:ae:52:ba:65:97   D         -   ge-0/0/6.0          0         0
```

There are three different configurations on the EX Series Switch that can apply MAB:

1. Standard MAB using a MAC address (allows from any interface)
2. MAB on a specific interface (lock it down)
3. MAB on a specific interface and assigned to a specific VLAN

Here are the corresponding configuration examples:

```
# set protocols dot1x authenticator static d4:ae:52:ba:65:97
# set protocols dot1x authenticator static d4:ae:52:ba:65:97 interface ge-0/0/6.0
# set protocols dot1x authenticator static d4:ae:52:ba:65:97 interface ge-0/0/6.0 vlan-
assignment V16
```

You can see that the three options build on each other until you reach the ability to assign it to a specific VLAN. Let's delete the old protocol dot1x configuration and now commit this last MAB statement.

TIP MAB *cannot* be configured with a single or single-secured supplicant statement but it can be configured with a multiple supplicant.

To apply our CentOS MAC address and bypass any authentication you can remove the entire `dot1x protocol` hierarchy and then add the three lines of code to enable the bypass:

```
# delete protocol dot1x
# set protocols dot1x authenticator interface ge-0/0/6.0 supplicant multiple
# set protocols dot1x authenticator static d4:ae:52:ba:65:97/48 vlan-assignment V16
# set protocols dot1x authenticator static d4:ae:52:ba:65:97/48 interface ge-0/0/6.0
# commit
```

Now turn off the interface on the CentOS platform and clear the dot1x interface on the switch as shown in Figure 11.15.

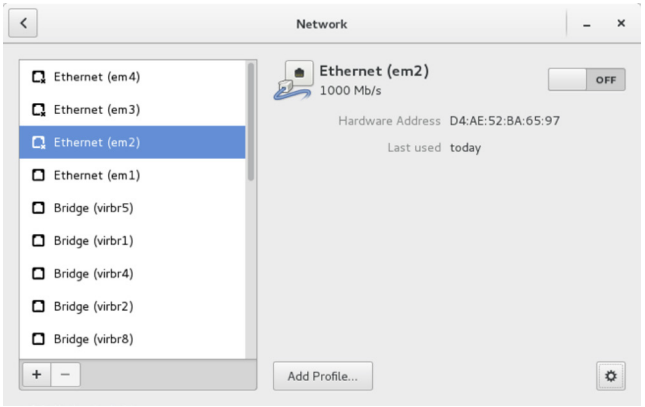


Figure 11.15 CentOS Network Interface

```
# run clear dot1x interface ge-0/0/6

{master:0}[edit]
EX2300# run show dot1x interface ge-0/0/6
802.1X Information:
Interface     Role           State          MAC address          User
ge-0/0/6.0    Authenticator  Connecting
```

And when you turn the interface on, the server it will be assigned to the correct VLAN without actually communicating with the server over RADIUS because it is statically mapped to the interface.

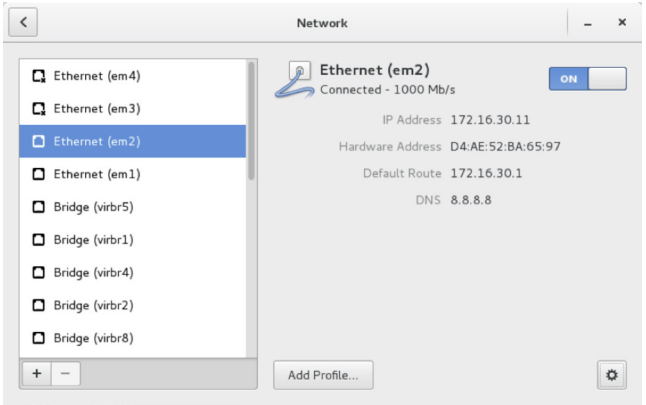


Figure 11.16 *CentOS Network interface.*

Next let's look at the output from the monitor DOT1X traceoptions file:

```
# Jun 26 15:03:35.985393 l2ald: received mac learn req for mac: d4:ae:52:ba:65:97 vlan:0 bd_idx:8
Jun 26 15:03:35.985541 l2d query for mac d4:ae:52:ba:65:97 vlan id 16
Jun 26 15:03:35.985633 check bypass authentication, match configured entry (d4:ae:52:ba:65:97/48)
Jun 26 15:03:35.985677 Match found, Static Mac configured for interface (ge-
0/0/6.0) interface num :561
Jun 26 15:03:35.985710 dot1x configured vlan is (V16)
Jun 26 15:03:35.985758 pnac_ng_get_vlan_id_from_vlan_data:298: vid (16) of vlan (V16)
Jun 26 15:03:35.985795 pnac_bd_tag_lookup 148 4 16
Jun 26 15:03:35.985850 DEBUG dot1x_server_send_mac_ifbd_
response: Sent IFBD ADD to l2ald for d4:ae:52:ba:65:97 vlan:16 bd:8
Jun 26 15:03:35.986269 Add (d4:ae:52:ba:65:97) to mac-bypass list on interface (ge-
0/0/6.0) vlan(V16)
Jun 26 15:03:35.986379 pnac_get_filter_term:1185 Term: dot1x_ge-0/0/6_DOT1X_
sm Filter op:0, Term op:0 Prev term name:
Jun 26 15:03:35.987104 Authentication bypassed for mac address  d4:ae:52:ba:65:97  on interface ge-
0/0/6.0
Jun 26 15:03:35.987204 pnac_bd_tag_lookup 148 4 16
Jun 26 15:03:35.987262 mac learn rsp to l2d ifl:561 mac:d4:ae:52:ba:65:97 vlan:16 bd:8 authstat:1
```

As you can see from the output, as soon as the interface learned the mac address, a match was found and the interface was assigned `vlan V16` and bypassed authentication. You can next see the `dot1x interface` is not using any supplicant and we are not using MAC-RADIUS, but are connected and able to use the interface since all authentication was bypassed based on the static MAC address assignment:

```
# run show dot1x interface ge-0/0/6.0 detail
ge-0/0/6.0
  Role: Authenticator
  Administrative state: Auto
  Supplicant mode: Multiple
  Number of retries: 3
  Quiet period: 60 seconds
  Transmit period: 30 seconds
  Mac Radius: Disabled
```

```
  Mac Radius Restrict: Disabled
  Reauthentication: Enabled
   Reauthentication interval: 3600 seconds
   Supplicant timeout: 30 seconds
   Server timeout: 30 seconds
   Maximum EAPOL requests: 2
   Guest VLAN member: not configured
   Number of connected supplicants: 0

# run ping 172.16.30.11
PING 172.16.30.11 (172.16.30.11): 56 data bytes
64 bytes from 172.16.30.11: icmp_seq=0 ttl=64 time=8.933 ms
^C
--- 172.16.30.11 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/stddev = 8.933/8.933/8.933/0.000 ms
```

To review, this section discussed using supplicants with an 802.1x RADIUS server to allow connectivity based on authentication, as well as MAC RADIUS, where the MAC address sits on the server and is authenticated, and then MAC Authentication Bypass, or MAB, whereby the static assignment of the MAC address bypasses the NAC and allows access based solely on the assigned MAC.

One thing not discussed so far is fallback. Let's say that for some reason you took your laptop to another building and your machine wasn't authorized there? You have the ability to establish fallback access either by VLANs or you can use a captive portal solution, which is next.

Configuring Captive Portal

Anyone who has accessed a guest WiFi outside of work, or even maybe at work or a hotel, has seen a Captive Portal web page. Captive Portal in conjunction with a solid 802.1x policy is a good way to control access.

This section is going to show you how to configure Captive Portal and set up the basic service. First, you need to generate an SSL certificate for your switch.

NOTE You can find numerous references on how to how to build an OpenSSL server via a Google search. If you know a good server guy, they can always generate one for you, too. To make this official you would generate a CSR and send it off to a Certificate Authority (CA).

To generate the self-signed SSL, let's jump on our Centos 7 server that has Open-SSL loaded and generate a Privacy Enhanced Mail (PEM) file – this is the equivalent of a Certificate Signed Request (CSR), where the encoding can be PEM. The following command issued on an Open-SSL server will generate a PEM file that you can use for your HTTPS captive portal:

```
$ openssl req -x509 -nodes -newkey rsa:2048 -keyout captive-portal.pem -out captive-portal.pem
Generating a 2048 bit RSA private key
```

```
.............................+++
.....................................+++
writing new private key to 'captive-portal.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:IL
Locality Name (eg, city) [Default City]:Day One
Organization Name (eg, company) [Default Company Ltd]:Juniper
Organizational Unit Name (eg, section) []:System Engineering
Common Name (eg, your name or your server's hostname) []:EX
Email Address []:lab@juniper.net
$ ls -al captive-portal.pem
-rw-rw-r-- 1 lab lab 3136 Jun 23 13:47 captive-portal.pem
$ scp captive-portal.pem lab@192.168.2.6:/var/tmp/
captive-portal.pem
100% 3136     3.1KB/s   00:00
```

Now that you have generated a certificate and copied it to the switch you need to tell the switch where to find it and to load it into the `security` hierarchy:

```
# set security certificates local captive-portal load-key-file /var/tmp/captive-portal.pem
```

Then you need to apply the rest of the captive portal configuration to the switch. Apply the PEM certificate generated earlier and apply it to the `system services web-management` hierarchy:

```
# set system services web-management https local-certificate captive-portal
```

Next apply HTTP and HTTPS to the interfaces through the `web-management` hierarchy. (You could use a wildcard here if you have a lot of interfaces.)

```
# set system services web-management https interface irb.16
# set system services web-management http interface irb.16
```

Then tell `captive-portal` that it will be using `secure-authentication` via `https`. Supply a custom URL for the `captive-portal` to report to, once authenticated through the captive portal:

```
# set services captive-portal secure-authentication https
# set services captive-portal interface ge-0/0/6.0
# set services captive-portal custom-options post-authentication-url http://192.168.2.5
# commit
configuration check succeeds
commit complete
```

Once the configuration is successfully committed let's check the status:

```
# run show captive-portal interface ge-0/0/6.0
Captive Portal Information:
Interface      State          MAC address          User          Fallen back
ge-0/0/6.0     Connecting                                        No

# run show captive-portal interface ge-0/0/6.0 detail
ge-0/0/6.0
  Supplicant mode: Single
  Number of retries: 3
  Quiet period: 60 seconds
  Configured CP session timeout: 3600 seconds
  Server timeout: 15 seconds
  CP fallen back: No
```

Now all you have left to do is to attach a client to the interface and see if it is working. Unlike MAB or MAC-RADIUS you get the DHCP information right away with captive portal. To test, point a browser at an interface in VLAN16. Once the captive portal sees a request it will post the Terms & Conditions page similar to Figure 11.17.

Figure 11.17 *Terms and Conditions for Captive Portal.*

After accepting the terms, you're taken to an authentication landing page (Figure 11.18).

Figure 11.18 *Captive Portal Login.*

For this example, labuser1 is used for the login. Once authenticated by the RADIUS server you're taken to the URL, or in this configuration, you are redirected to another URL after authentication. This can be used to direct clients to a landing page that has the links to your company's compliance training, downloads for virus updates, or instructions to access the network that requires them to pay for access. On the switch, you can see that labuser1 is authenticated with the captive-portal:

```
EX2300# run show captive-portal interface ge-0/0/6.0
Captive Portal Information:
Interface      State          MAC address          User          Fallen back
ge-0/0/6.0     Connecting     d4:ae:52:ba:65:97    No User       No

EX2300# run show captive-portal interface ge-0/0/6.0
Captive Portal Information:
Interface      State          MAC address          User          Fallen back
ge-0/0/6.0     Authenticated  d4:ae:52:ba:65:97    labuser1      No
```

This example shows the before and after status of the captive portal interface and clearly indicates the state and user as well as the interface and MAC address. We also get supervisory information (Figure 11.19) stating that we have been authenticated and that we are being redirected.

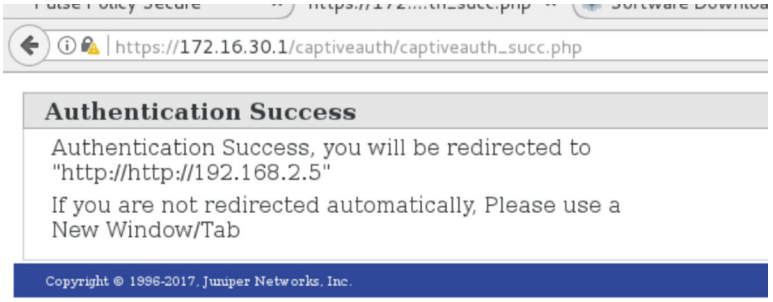


Figure 11.19 *Authentication Response.*

This comes from the redirect statement in our captive-portal configuration.

```
EX2300# set services captive-portal custom-options post-authentication-url http://192.168.2.5
```

Captive portal is an easy way to enforce compliance if you don't have an 802.1X server. It can also be used as a fallback to 802.1X if it becomes unresponsive. For more information on captive portal and different configuration options please refer to the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/authentication-captive-portal-cli.html.

Summary

To review, this chapter showed you dot1X, MAC Authentication Bypass, and Captive Portal. These can also be used in flexible authentication as a fallback mechanism in case one of the authentication methods is not available. To get more detailed information on flexible authentication order go to the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/authentication-order-ex-series-cli.html.

Chapter 12

Configuring EX Series Firewalls and Policers

Configuring Policers and Firewalls

One thing to note is that firewall filters in Junos are not stateful, meaning it does not keep track of network connections, Therefore, if you have traffic that is interrupted while you are working on firewall rules, it will have to start a new session. If you need a stateful firewall, then it would be best to look at the Juniper Networks SRX platforms: https://www.juniper.net/us/en/products-services/security/srx-series/.

Because the firewalls are not stateful, each packet is processed independently, and unlike the SRX Series firewall, that maintains sessions and will open ports back towards the source, you have to explicitly allow traffic in both directions.

NOTE Firewall filters in Junos are hardware based so they are very efficient in processing the packets through the applied filters.

There is no way we can possibly include all options in this *Day One* book or even in a separate book and there is a very good engineering document already written that explains everything in detail – 384 pages of it, in fact, at: https://www.juniper.net/documentation/en_US/junos12.3/information-products/pathway-pages/config-guide-firewall-filter/config-guide-firewall-filter.pdf. Make sure you get the knowledge transfer!

Instead, let's discuss the types of firewalls used in EX Series switches and apply them to the specific interfaces and VLANs on a switch since switches are a little different.

The first thing you need to know for a firewall filter is the family. To understand which family to use you also need to understand where it is going to be applied. There are three distinct families used by the firewall:

```
# set firewall family ?
Possible completions:
> any                   Protocol-independent filter
> ethernet-switching    Protocol family Ethernet Switching for firewall filter
> inet                  Protocol family IPv4 for firewall filter
> inet6                 Protocol family IPv6 for firewall filter
```

There are also three distinct areas of the switch that our firewall filters can be applied:

- Ports
- VLANS
- Routed Interfaces

That is also the order that they are processed, on an ingress interface. If the filter is applied to an egress interface, or outbound, then the order would be routed: interface, VLAN, port.

TIP Firewall filters are also referred to as Access Control Lists (ACLs).

All of the Junos devices have "built-in" policers that protect the routing engine (aka, the Control Plane). You can see some of these using the `show policer` command, like so:

```
EX4300-6_7-VC> show policer ?
Possible completions:
  <[Enter]>            Execute this command
  <policer>            Policer name
  __auto_policer_template_1__
  __auto_policer_template_2__
  __auto_policer_template_3__
  __auto_policer_template_4__
  __auto_policer_template_5__
  __auto_policer_template_6__
  __auto_policer_template_7__
  __auto_policer_template_8__
  __auto_policer_template__
  __cfm_filter_shared_lc__
  __default_arp_policer__
  __dhcpv6__
  __jdhcpd__
  __jdhcpd_l2_snoop_filter__
  |                    Pipe through a command
{master:1}
EX4300-6_7-VC> show policer __default_arp_policer__
Policers:
Name                                    Bytes              Packets
__default_arp_policer__                     0                    0
```

As you can see the built-in policers use underscores to set them off from the rest. Do not attempt to alter these policers or utilize the same names. These policers perform a very specific function.

You can, however, create your own policers (also referred to as rate limiting) to protect the routing engine from excessive SNMP queries or SSH requests or even Denial of Service (DoS) attacks.

To accomplish this, you need to understand the level of traffic you have for each protocol and how policers actually operate within Junos. The token bucket algorithm is used to limit the IP protocol transmit or receive rate on a per interface basis while allowing a maximum burst size on the overall traffic or session. To demonstrate this, we are going to use a bar code that actually reads JUNIPERRULES:

Figure 12.1 *JUNIPERRULES Burst Size Example.*

Now let's assume the thicker lines in the bar code are 5ms bursts of data on a 1Gbps interface and that this is the data represented over 1 second of time. If you do the math you can see that the actual burst limit for 5ms would be 625Kbps.

```
Interface Rate in Bits per second: 1,000,000,000
Burst window: 5ms
Bits: 8
Burst-size = (1,000,000,000bps * .005s)/8b = 625Kbps
```

This tells you where you stand with a 5ms burst now but you can limit the burst rate down to 400Kbps or even lower if desired. The key to this is the time that you allow a burst. The interface bandwidth is fixed (and depends on your type of interface) and so are the bits, therefore your burst time is really the key to the formula: `burst-size = (interface bandwidth x burst time allowed)/8`. There is another formula to calculate the burst-length as well and that is: `burst-size = Interface MTU x 10`.

MORE? There is a lot to cover on calculating burst size and the effects on the network. Please refer to the following documentation for more in-depth information on determining the proper burst size: https://www.juniper.net/documentation/en_US/junos12.3/topics/concept/policer-mx-m120-m320-burstsize-determining.html.

Okay, there's a gigabit ethernet interface and you know you do not want more than a 5ms burst so you can set the burst-length to 625K (62.5% of the overall bandwidth) or use the MTU x 10 option, in which case, you come up with burst size = ((9216 x8) x 10) = ~737K and this is approximately 6 milliseconds of burst time with an MTU of 9216, and with an MTU of 1500 burst size = ((1500 x 8) x 10) = 120K.

TIP You should experiment with burst size and make sure that downstream neighbors can handle the traffic without dropping packets.

Let's demonstrate implementing a policer and a firewall filter across our small lab connection as illustrated in Figure 12.2.

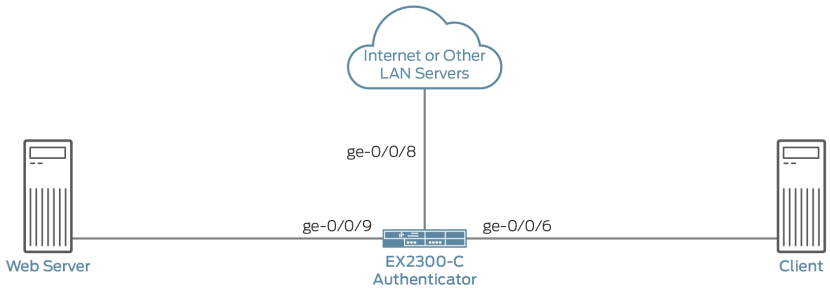


Figure 12.2 *Policer Lab Diagram.*

You need to define a policer that will discard HTTP port 80 traffic if it hits a set `burst-size-limit` and also a `bandwidth-limit`. Let's place the policer inside a firewall filter and then apply it to our interface on ge-0/0/6.0.

Let's set the limits pretty low (the lowest that is accepted, in fact) so we can trigger the discard and see how it is working:

```
# set groups DO_POLICER firewall policer HTTP-DISCARD if-exceeding bandwidth-limit 32k
# set groups DO_POLICER firewall policer HTTP-DISCARD if-exceeding burst-size-limit 1500
# set groups DO_POLICER firewall policer HTTP-DISCARD then discard
```

Next, create a firewall filter to use the policer that we've created:

```
# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP from protocol tcp
# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP from destination-
port http
# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP then policer HTTP-
DISCARD
# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP then count http.
counter
# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP then log
# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term ACCEPT-REST then accept
```

Note that we included the ACCEPT-REST term in the firewall filter to allow all other traffic to be accepted, otherwise it would be dropped. And now we can apply the group DO_POLICER:

```
# set apply-groups DO_POLICER
# commit
```

Now you have a policer and a firewall but you still haven't applied it to the interface ge-0/0/6.0. One thing you should do before applying the filter to the interface is to clear the statistics so you have a clean slate:

```
EX2300# run clear interfaces statistics all
```

And now let's apply the firewall filter to the interface:

```
EX2300# set interfaces ge-0/0/6.0 family inet filter input HTTP-DISCARD-IN
EX2300# commit
configuration check succeeds
commit complete

EX2300# run show firewall filter HTTP-DISCARD-IN
Filter: HTTP-DISCARD-IN
Counters:
Name                                                Bytes              Packets
http.counter                                            0                    0
Policers:
Name                                                Bytes              Packets
HTTP-DISCARD-LIMIT-HTTP                                 0                    0
```

Everything is in place now: the policer, the firewall filter, and the filter applied to the interface. Now let's test from the client connected on port ge-0/0/6 to the web server connected on port ge-0/0/9:

```
$ sudo hping3 -i u10 -S -p 80 -c 30  192.168.2.5
HPING 192.168.2.5 (em2 192.168.2.5): S set, 40 headers + 0 data bytes
len=46 ip=192.168.2.5 ttl=63 DF id=0 sport=80 flags=SA seq=0 win=29200 rtt=0.4 ms
len=46 ip=192.168.2.5 ttl=63 DF id=0 sport=80 flags=SA seq=1 win=29200 rtt=0.5 ms
len=46 ip=192.168.2.5 ttl=63 DF id=0 sport=80 flags=SA seq=4 win=29200 rtt=0.4 ms
...
len=46 ip=192.168.2.5 ttl=63 DF id=0 sport=80 flags=SA seq=28 win=29200 rtt=0.1 ms
len=46 ip=192.168.2.5 ttl=63 DF id=0 sport=80 flags=SA seq=29 win=29200 rtt=0.1 ms
--- 192.168.2.5 hping statistic ---
30 packets transmitted, 23 packets received, 24% packet loss
```

```
round-trip min/avg/max = 0.1/0.4/0.7 ms
```

Run this three times in quick succession and then look at the firewall filter and counter:

```
# run show firewall filter HTTP-DISCARD-IN
Filter: HTTP-DISCARD-IN
Counters:
Name                                                Bytes              Packets
http.counter                                        11520                180
Policers:
Name                                                Bytes              Packets
HTTP-DISCARD-LIMIT-HTTP                             5952               93
```

The counter shows that 11520 bytes actually passed through the interface and 5952 or almost half the packets were caught by the `HTTP-DISCARD-LIMIT` filter. You can also add SSH and other protocols and ports to the firewall filter to create a comprehensive firewall filter.

One last thing to show is that you can view the firewall log and see the ports and protocols being accepted and rejected on a per firewall basis:

```
EX2300# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-
SSH from destination-port ssh
EX2300# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-
SSH then count ssh.counter
EX2300# set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-SSH then reject
```

Now this is *very* important. When you add ACL's, or additional terms to a firewall filter, they will be placed at the *end of the filter*. In this case the three commands above will be placed *below* the `ACCEPT-REST` term. If you *do not change the order of the terms* all SSH traffic will hit the `ACCEPT-REST` term and be accepted and never reach the `LIMIT-SSH` term:

```
# show | display set | match DO_POLICER
…
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term ACCEPT-REST then accept
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-SSH from destination-
port ssh
```

So, you need to move that `LIMIT-SSH` term or insert it above the `ACCEPT-REST` term and the easy way to do that is just use the `insert` command:

```
EX2300# insert groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-
SSH before term ACCEPT-REST
EX2300# commit
```

And now you can see that the order is correct:

```
EX2300# show | display set | match DO_PO
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP from destination-
port http
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP then policer HTTP-
DISCARD
```

```
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP then count http.
counter
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-HTTP then log
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-SSH from destination-
port ssh
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-SSH then count ssh.
counter
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term LIMIT-SSH then discard
set groups DO_POLICER firewall family inet filter HTTP-DISCARD-IN term ACCEPT-REST then accept
set groups DO_POLICER firewall policer HTTP-DISCARD if-exceeding bandwidth-limit 32k
set groups DO_POLICER firewall policer HTTP-DISCARD if-exceeding burst-size-limit 1500
set groups DO_POLICER firewall policer HTTP-DISCARD then discard
set apply-groups DO_POLICER

Now let's send some SSH packets through the interface and see if they are discarded:
# run show firewall filter HTTP-DISCARD-IN
Filter: HTTP-DISCARD-IN
Counters:
Name                                                Bytes              Packets
http.counter                                        11520                  180
ssh.counter                                          3840                   60
Policers:
Name                                                Bytes              Packets
HTTP-DISCARD-LIMIT-HTTP       5504                   86
```

And as you can see the SSH packets were 100% discarded because we took that action and didn't use a policer while our HTTP packets were about 50% successful due to the rate hping3 was hitting the port.

TIP Hping3 is in the yum repository for CentOS as well as the debian repository for Ubuntu. The Kali Tools page has a very in depth package description and utilization examples: https://tools.kali.org/information-gathering/hping3.

This is as far as we're going to explore firewalls and policers because there are already some great books and technical documents that are much more thorough than we can be here.

Monitoring Ternary Content Addressable Memory (TCAM)

One thing you need to be aware of when applying firewalls to an EX Series switch is the Dynamic Ternary Content Addressable Memory or TCAM limitation. In the EX Series, TCAM is used by the different firewall filters such as inet, inet6, vlans, and interfaces, as well as 802.1ag OAM/CFM.

The problem is that the TCAM is limited based on the type of switch you are deploying. Some switches have a fixed TCAM while others are dynamically allocated. Here is a list of Juniper Networks EX Series switches and their firewall rule limits:

Table 12.1 Ternary Content Addressable Memory (TCAM)

EX MODEL	TERMS	DIRECTION	TYPE	STATUS
2200	512	INGRESS+EGRESS	INET/INET6	
2300	256	INGRESS+EGRESS	INET/INET6	New ELS Model
3200	7042	INGRESS+EGRESS	INET/INET6	Check EOL Status
3300	1436	INGRESS+EGRESS	INET/INET6	
3400	256	INGRESS+EGRESS	INET/INET6	New ELS Model
4200	7042	INGRESS+EGRESS	INET/INET6	Check EOL Status
4300	3500	INGRESS	PORT	ELS
4300	3500	INGRESS	VLAN	ELS
4300	7000	INGRESS	INET	ELS
4300	3500	INGRESS	INET6	ELS
4300	512	EGRESS	PORT	ELS
4300	256	EGRESS	VLAN	ELS
4300	512	EGRESS	INET	ELS
4300	512	EGRESS	INET6	ELS
4500	1200	INGRESS+EGRESS	INET/INET6	Check EOL Status
4550	1200	INGRESS+EGRESS	INET/INET6	ELS
4600	2560	INGRESS+EGRESS	INET/INET6	ELS

To demonstrate how to monitor your filter and TCAM limits let's apply some simple input and output firewall filters and apply them to our EX4300 in the lab. Then run a `show` command to see how many filter entries have been used and how many are left for ingress and egress. Here are the two very simple firewall filters applied to the switch:

```
EX2300-16# show firewall | display set
set firewall family inet filter RE-PROTECT term LOG-SSH from protocol tcp
set firewall family inet filter RE-PROTECT term LOG-SSH from destination-port 22
set firewall family inet filter RE-PROTECT term LOG-SSH then count SSH-COUNTER
set firewall family inet filter RE-PROTECT term LOG-SSH then log
set firewall family inet filter RE-PROTECT term LOG-SSH then accept
set firewall family inet filter RE-PROTECT term ACCEPT-ALL then accept
set firewall family inet filter OUTGOING-SSH term SSH-OUT from protocol tcp
set firewall family inet filter OUTGOING-SSH term SSH-OUT from destination-port 22
set firewall family inet filter OUTGOING-SSH term SSH-OUT then count SSH-EGRESS
set firewall family inet filter OUTGOING-SSH term SSH-OUT then accept
set firewall family inet filter OUTGOING-SSH term ALLOW-ALL then accept
```

NOTE You cannot use logging in output filters. Also, you should note the `ACCEPT-ALL` and `ALLOW-ALL` terms at the end of these statements. If you did not accept all other traffic, you would lock yourself out because these are not stateful firewall filters.

NOTE On the EX3300, if you are applying large filters greater than 1000 terms, then you have to apply *all* filters at the same time and delete all filters at the same time using a single commit on each operation.

Now let's apply them to our interfaces. First, apply the RE-PROTECT to the input of the loopback 0 interface and then the OUTGOING-SSH filter to the output of irb.100:

```
EX2300-16# set interfaces lo0.0 family inet filter input RE-PROTECT
EX2300-16# set interfaces irb.100 family inet filter output OUTGOING-SSH
EX2300-16# commit
configuration check succeeds
commit complete
```

Once in place let's check the filter utilization.

```
EX2300-16> show pfe filter hw summary
Slot 0
Group                    Group-ID       Allocated      Used           Free
-------------------------------------------------------------------------------
> Ingress filter groups:
  iRACL group            33             128            8              120
> Egress filter groups:
  eRACL group            54             128            4              124
```

By applying those two simple firewall filters we have eaten up twelve of our 512 total firewall entries. This is why you need to be careful on how you are applying filters, especially if they have a lot of terms.

It should be noted that the term limit is on a *per* switch basis. Therefore, you could have more free ACLs on other switches when you are using them in a Virtual Chassis configuration like so:

```
EX2300-8_9-VC> show pfe filter hw summary
Slot 0
Group                    Group-ID       Allocated      Used           Free
-------------------------------------------------------------------------------
> Ingress filter groups:
  iPACL group            25             128            3              125
> Egress filter groups:
Slot 1
Group                    Group-ID       Allocated      Used           Free
-------------------------------------------------------------------------------
> Ingress filter groups:
> Egress filter groups:
Slot
```

While you are applying filters, it is a good idea to send logs to your user-id, console, or syslog. You can do this with the following commands:

```
EX2300-16# set system syslog user lab firewall emergency
EX2300-16# set system syslog console firewall emergency
EX2300-16# set system syslog file FIREWALL firewall emergency
```

On those devices that support TCAM you can also use:

```
EX4600-4_5-VC# set system syslog user lab pfe emergency
EX4600-4_5-VC# set system syslog console pfe emergency
EX4600-4_5-VC# set system syslog file PFE_TCAM pfe emergency
```

Once those are committed and you encroach the filter limits, you should get notified via a syslog message that looks something like:

```
DFWE DFW: TCAM FULL cannot program filter "RE-PROTECT" (type IRACL_
LO) - TCAM has 200 free entries and the filter requires 128 free entries
```

When applying firewall filters, you have to be conscientious of the number of terms that are available per device.

Configuring Class of Service (CoS)

You can configure class of service on an EX Series switch by hand or an easy way to configure CoS on a Juniper device is to use the predefined EZQOS configuration that is already on the box!

Let's take a look at the EX3400 in the diagram and see what we have in /etc/config/:

```
EX3400-10_11-VC> file list /etc/config
/etc/config:
ex3400-24p-defaults.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-24p-defaults.conf
ex3400-24p-factory.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-24p-factory.conf
ex3400-24t-defaults.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-24t-defaults.conf
ex3400-24t-factory.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-24t-factory.conf
ex3400-48p-defaults.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-48p-defaults.conf
ex3400-48p-factory.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-48p-factory.conf
ex3400-48t-defaults.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-48t-defaults.conf
ex3400-48t-factory.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ex3400-48t-factory.conf
ezqos-voip.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ezqos-voip.conf
ezsetup-tvp.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/ezsetup-tvp.conf
junos-defaults.conf@ -> /packages/mnt/junos-runtime/etc/config/junos-defaults.conf
junos-factory.conf@ -> /packages/mnt/junos-runtime/etc/config/junos-factory.conf
junos-fips-defaults.conf@ -> /packages/mnt/junos-runtime/etc/config/junos-fips-defaults.conf
l2ng-autoimage.conf@ -> /packages/mnt/junos-runtime-ex/etc/config/l2ng-autoimage.conf
pvi-model-factory.conf@ -> /etc/config/ex3400-24t-factory.conf
shmlog/
subs-mgmt-proc-set@ -> /packages/mnt/junos-runtime/etc/config/subs-mgmt-proc-set
{master:0}
lab@EX3400-10_11-VC>
```

Let's use the predefined ezqos-voip.conf file that is located in the /etc/config directory. To load the EZQOS configs perform the following:

```
# load merge /etc/config/ezqos-voip.conf
load complete
```

WARNING Make sure you use the `merge` keyword and *not* the `override` keyword when doing this load. Use the `show | compare` command to check your work and use `commit confirmed` when committing this, just to be safe. You don't want to blow away your entire configuration.

```
# set apply-groups ezqos-voip
# commit
```

Now you have a complete COS profile with very little learning curve. Let's take a look at the actual configuration that was applied.

```
# show | display set | match ezqos
set groups ezqos-voip class-of-service classifiers dscp ezqos-dscp-classifier import default
set groups ezqos-voip class-of-service classifiers dscp ezqos-dscp-classifier forwarding-class ezqos-
voice-fc loss-priority low code-points 101110
set groups ezqos-voip class-of-service classifiers dscp ezqos-dscp-classifier forwarding-class ezqos-
control-fc loss-priority low code-points 110000
set groups ezqos-voip class-of-service classifiers dscp ezqos-dscp-classifier forwarding-class ezqos-
control-fc loss-priority low code-points 011000
set groups ezqos-voip class-of-service classifiers dscp ezqos-dscp-classifier forwarding-class ezqos-
control-fc loss-priority low code-points 011010
set groups ezqos-voip class-of-service classifiers dscp ezqos-dscp-classifier forwarding-class ezqos-
control-fc loss-priority low code-points 111000
set groups ezqos-voip class-of-service classifiers dscp ezqos-dscp-classifier forwarding-class ezqos-
video-fc loss-priority low code-points 100010
set groups ezqos-voip class-of-service forwarding-classes class ezqos-best-effort queue-num 0
set groups ezqos-voip class-of-service forwarding-classes class ezqos-video-fc queue-num 4
set groups ezqos-voip class-of-service forwarding-classes class ezqos-voice-fc queue-num 5
set groups ezqos-voip class-of-service forwarding-classes class ezqos-control-fc queue-num 7
set groups ezqos-voip class-of-service scheduler-maps ezqos-voip-sched-maps forwarding-class ezqos-
voice-fc scheduler ezqos-voice-scheduler
set groups ezqos-voip class-of-service scheduler-maps ezqos-voip-sched-maps forwarding-class ezqos-
control-fc scheduler ezqos-control-scheduler
set groups ezqos-voip class-of-service scheduler-maps ezqos-voip-sched-maps forwarding-class ezqos-
video-fc scheduler ezqos-video-scheduler
set groups ezqos-voip class-of-service scheduler-maps ezqos-voip-sched-maps forwarding-class ezqos-
best-effort scheduler ezqos-data-scheduler
set groups ezqos-voip class-of-service schedulers ezqos-voice-scheduler buffer-size percent 20
set groups ezqos-voip class-of-service schedulers ezqos-voice-scheduler priority strict-high
set groups ezqos-voip class-of-service schedulers ezqos-control-scheduler buffer-size percent 10
set groups ezqos-voip class-of-service schedulers ezqos-control-scheduler priority strict-high
set groups ezqos-voip class-of-service schedulers ezqos-video-scheduler transmit-rate percent 70
set groups ezqos-voip class-of-service schedulers ezqos-video-scheduler buffer-size percent 20
set groups ezqos-voip class-of-service schedulers ezqos-video-scheduler priority low
set groups ezqos-voip class-of-service schedulers ezqos-data-scheduler transmit-rate percent 30
set groups ezqos-voip class-of-service schedulers ezqos-data-scheduler buffer-size percent 50
set groups ezqos-voip class-of-service schedulers ezqos-data-scheduler priority low
```

There are four forwarding classes defined in the ezqos-dscp-classifier:

- ezqos-best-effort – queue 0
- ezqos-video-fc – queue 4
- ezqos-voice-fc – queue 5
- ezqos-control-fc – queue 7

Each of the forwarding classes are also assigned unique Differentiated Service Code Point (DSCP) values:

- ezqos-video-fc – 100010 (34)
- ezqos-voice-fc – 101110 (46)

- ezqos-control-fc – 110000, 011000, 100010 (48-63,24,34)
- ezqos-best-effort – Doesn't get DSCP because it is Best Effort but it will take DSCP 000000 to101111

There are classifiers, predefined forwarding classes, scheduler-maps and schedulers and they are all built around supporting voice and video. Once this group is applied to the configuration using the # `set apply-groups ezqos-voip` command you can assign CoS to your interfaces and firewall filters.

To apply the DSCP to an interface you configure it under the `class-of-service` hierarchy:

```
EX2300# set class-of-service interfaces ge-0/0/6 unit 0 classifiers dscp ezqos-dscp-classifier
```

The `class-of-service` hierarchy is also used to apply the scheduler to an interface:

```
EX2300# set class-of-service interfaces ge-0/0/9 scheduler-map ezqos-voip-sched-maps
```

Now once that is committed you should have a CoS path for the traffic:

```
EX2300# run show class-of-service interface ge-0/0/6
Physical interface: ge-0/0/6, Index: 654
Maximum usable queues: 8, Queues in use: 7
  Scheduler map: <default>, Index: 2
  Congestion-notification: Disabled
  Logical interface: ge-0/0/6.0, Index: 567
Object                  Name                  Type                  Index
Classifier              ezqos-dscp-classifier  dscp                 57624
```

This was just a short primer on CoS for EX Series switches. To dive deeper into the details please refer to the Juniper TechLibrary at: https://www.juniper.net/documentation/en_US/junos/topics/example/cos-ex-series-configuring.html.

Configuring Storm Control

Storm control manages the amount of Broadcast, Unknown Unicast, and Multicast (BUM) traffic on the network. A *storm* occurs when a broadcast message from one device prompts another device in the network to respond with its own broadcast message and then another and another until your entire network bandwidth is utilized for nothing but broadcast messages. The same can happen with Unknown Unicast and Multicast.

Storm control is enabled in ELS capable devices under the `forwarding-options` hierarchy. The EX4300's storm control is configured by default. Other models may not have `storm-control` enabled by default, like the EX9200, so make sure you check your configuration.

The configuration for storm control is pretty straightforward. First, define your storm control profile:

```
# set forwarding-options storm-control-profiles default all
```

The next step is to apply the default storm control profile to all of your interfaces. You can do this with the `wildcard` command and be selective. If for some reason, you didn't want interfaces ge-0/0/9 and ge-0/0/10 to be enabled for storm control you can leave those out.

```
# wildcard range set interfaces ge-0/0/[0-8,11] unit 0 family ethernet-switching storm-
control default
```

Once you commit this you have applied storm control to all of your specified interfaces.

NOTE By default, `storm-control` limits the combined BUM traffic to 80%. That is combined Broadcast, Unknown Unicast and Multicast.

You can set your own limits and customize your profile or on a per interface basis using the two key configuration elements; `Bandwidth Level` and `Bandwidth Percentage`. You can also be very selective with `storm-control`. For more in depth information on managing traffic levels using storm control please seek it here: https://www.juniper.net/documentation/en_US/junos/topics/concept/rate-limiting-storm-control-understanding-els.html.

Storm control is an important mechanism to combat BUM storms in the network. If you think your network is acting sluggish or unresponsive make sure you check that storm control is enabled on all devices in your network. You can also configure storm control to take the action of `action-shutdown` on interfaces that are experiencing a storm and also set recovery actions to turn them back up. The example we just showed you is basic so make sure you follow the TechLibrary links and read up on all the details.

Configuring Port Mirroring

Port mirroring is a valuable troubleshooting tool that allows you to monitor and perform packet captures on interfaces where you need to validate or take a deeper look into the packet with a packet capture tool, such as Wireshark. Juniper allows you to send copied packets to local interfaces for local monitoring or to a VLAN for remote monitoring. You can port mirror on interfaces or VLANs.

Using the `groups` hierarchy can assist you in enabling and disabling the port mirroring configuration.

BEST PRACTICE It is best practice to `disable port-mirroring` configurations when not in use. Using a specific interface is always ideal and limiting the amount of mirrored traffic being captured via firewall filters is also best practice.

The following configuration is much different than the older, non-ELS Junos. Make sure you know the hardware and software that you are configuring by referencing the ELS section at the beginning of this book. This example uses an EX2300 with a server that has Wireshark, as shown in Figure 12.3.

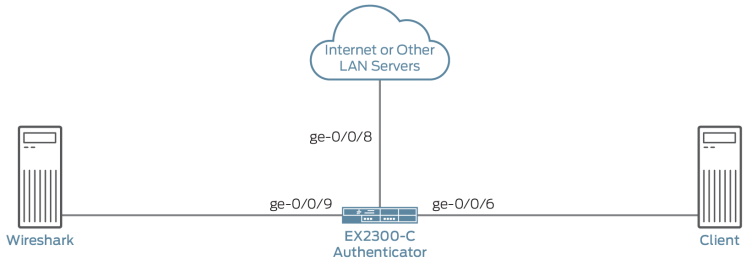


Figure 12.3 *Port Mirror Lab Diagram.*

NOTE The `monitor traffic` command that you were introduced to earlier *only* captures packets destined to the Routing Engine while Port Mirroring allows you to capture *all* transit traffic that is destined for an external host. You can be very selective by using firewall filters to only mirror specific traffic.

The following configuration will capture all ingress traffic from both ge-0/0/6 and ge-0/0/8 interfaces. This should give a good picture of the communication and traffic that is taking place between these two hosts on the same VLAN. The traffic will be captured by Wireshark on a server hanging off of interface ge-0/0/9:

```
# set groups PM forwarding-options analyzer PM-MONITOR input ingress interface ge-0/0/6.0
# set groups PM forwarding-options analyzer PM-MONITOR input ingress interface ge-0/0/8.0
# set groups PM forwarding-options analyzer PM-MONITOR output interface ge-0/0/9.0
# set apply-groups PM
```

Launching Wireshark and filtering on the IP 172.16.30.1 allows a quick filter to capture the traffic between 172.16.30.10 and 172.16.30.1, as seen in Figure 12.4, with 30.10 running an NMAP against 172.16.30.1, all of the ingress traffic is captured on our remote Wireshark interface:

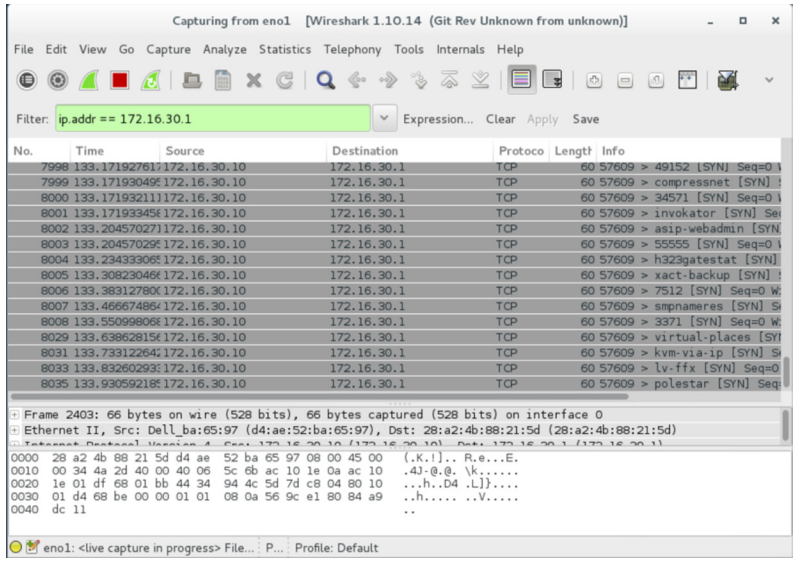


Figure 12.4 *Wireshark 1.10.14 GUI Traffic Capture.*

Summary

This was a really quick introduction in deploying a port mirror to capture packets with an analyzer. For more information on configuring `port-mirroring` on ELS devices please visit the TechLibrary: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/port-mirroring-cli-els.html.

Chapter 13

Configuring Media Access Control Security (MACSec)

What is MACSec

Media Access Control Security (MACSec) is a point-to-point Ethernet technology that provides security between switch-to-switch or host-to-switch connections. When you enable MACSec, a pre-shared key is exchanged between switches and includes a Connectivity Association Name (CKN) as well as a Connectivity Association Key (CAK). Both of these pre-shared keys must match on each switch on the point-to-point link for the link to be protected and then MACSec Key Agreement (MKA) Protocol is enabled.

The MKA is responsible for selecting the key server between the two switches.

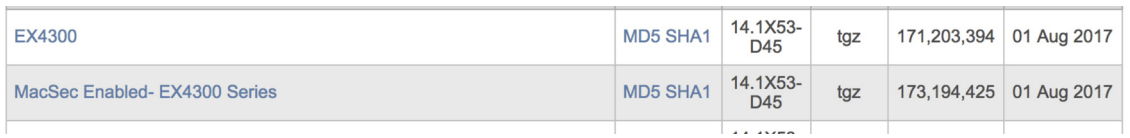
NOTE MACSec is an IEEE 802.1AE Standard: http://standards.ieee.org/about/get/802/802.1.html.

Hardware and Software supporting MACSec

There are several switches that support MACSec. In our *Day One* Lab we have four devices that support MACSec: EX3400, EX4300, EX4600, and QFX5100 but we also need to check the software support for MACsec as well.

TIP To check the hardware and software support for your own use of MACSec use Juniper's pathfinder to verify: https://pathfinder.juniper.net/feature-explorer/feature-info.html?fKey=6117&fn=Media+Access+Control+Security+(MACsec)+for+switch+to+switch+connections.

Each switch will require MACSec capable software to be running on the device before you can configure MACSec. There is a specific MACSec image that is controlled by Juniper, shown in Figure 13.1.



EX4300	MD5 SHA1	14.1X53-D45	tgz	171,203,394	01 Aug 2017
MacSec Enabled- EX4300 Series	MD5 SHA1	14.1X53-D45	tgz	173,194,425	01 Aug 2017

Figure 13.1 *Screen Capture Juniper MacSec Controlled Image.*

The standard image is not capable of supporting MACSec so you have to upgrade to the MACSec capable image before you run through this exercise. The jinstall file for the macsec image shows it as controlled:

```
jinstall-ex-4300-14.1X53-D45.3-controlled-signed.tgz
```

Before the upgrade the software version showed as:

```
EX4300-6_7-VC> show version
fpc0:
--------------------------------------
Hostname: EX4300-6_7-VC
Model: ex4300-24t
Junos: 14.1X53-D35.3
```

You can see that without the controlled software there is no way to configure MACSec:

```
EX4600-4_5-VC# set security mac
                                    ^
syntax error.
EX4600-4_5-VC# set security ?
Possible completions:
> alarms               Configure security alarms
+ apply-groups         Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
> authentication-key-chains  Authentication key chain configuration
> certificates         X.509 certificate configuration
> group-vpn            Group VPN configuration
> ike                  IKE configuration
> ipsec                IPSec configuration
> log                  Configure auditable security logs
> pki                  PKI service configuration
> ssh-known-hosts      SSH known host list
> traceoptions         Trace options for IPSec key management
```

There is no MACSec command present. Once you upgrade the software you can see the difference in the command set:

```
EX4300-6_7-VC> show version detail | match macsec
macsec-actions-dd release 14.1X53-D45.3 built by builder on 2017-07-28 03:34:29 UTC
macsec-actions-dd release 14.1X53-D45.3 built by builder on 2017-07-28 03:34:29 UTC
```

And you can now see that the `macsec` hierarchy is enabled and ready for us to configure:

```
EX4300-6_7-VC# set security ?
Possible completions:
> alarms               Configure security alarms
+ apply-groups         Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
> authentication-key-chains  Authentication key chain configuration
> certificates         X.509 certificate configuration
> group-vpn            Group VPN configuration
> ike                  IKE configuration
> ipsec                IPSec configuration
> log                  Configure auditable security logs
> macsec               MAC Security configuration
> pki                  PKI service configuration
> ssh-known-hosts      SSH known host list
> traceoptions         Trace options for IPSec key management
```

After the upgrade to MACSec-capable Junos, the next step is to acquire and load the MACSec license.

MACSec Feature License

MACSec is not ubiquitous now, but it has a bright future especially within the Federal and Financial verticals. The MACSec license is an additional license to any EFL that may already be present on your system, so you will need to contact your account teams to purchase for your network.

The license for the EX3400, for example, looks like this:

```
Serial No :              NV0216340092
Features :               EX-QFX-MACSEC-
ACC : MACSEC License for 1G Access platforms - EX3300, EX3400, EX4300 and EX4200
Issue Date :             05-AUG-2017
License Key :
Junos942854 aeaqia qmjzld amrrgy ztimbq hezama uqomds
            rwvvr6 ikzuoc heglnu tabzcr urdthx wynwdx
            7zu6ny wlynqk wlyrww t3jmfg rhayaw ky
```

And you apply it just like any other software license:

```
EX3400-10_11-VC> request system license add terminal
[Type ^D at a new line to end input,
 enter blank line between each license key]
Serial No :              NV0216340092
Features :               EX-QFX-MACSEC-
ACC : MACSEC License for 1G Access platforms - EX3300, EX3400, EX4300 and EX4200
Issue Date :             05-AUG-2017
License Key :
Junos942854 aeaqia qmjzld amrrgy ztimbq hezama uqomds
            rwvvr6 ikzuoc heglnu tabzcr urdthx wynwdx
            7zu6ny wlynqk wlyrww t3jmfg rhayaw ky
Junos942854: successfully added
add license complete (no errors)
```

Once loaded you can see that it is licensed:

```
EX3400-10_11-VC> show system license
License usage:
                                Licenses     Licenses    Licenses    Expiry
  Feature name                       used    installed      needed
  ospf3                                 0            1           0    permanent
  ripng                                 0            1           0    permanent
  ospf2                                 1            1           0    permanent
  bfd-liveness-detection                0            1           0    permanent
  unicast-rpf                           0            1           0    permanent
  virtual-router                        1            1           0    permanent
  igmp                                  0            1           0    permanent
  pim-mode                              0            1           0    permanent
  pim-ssm                               0            1           0    permanent
  connectivity-fault-management         0            1           0    permanent
  vrrp                                  0            1           0    permanent
  dot1q-tunneling                       0            1           0    permanent
  svlan                                 0            1           0    permanent
  services-rpm                          0            1           0    permanent
  juniper-msdp                          0            1           0    permanent
  multicast-listener-discovery          0            1           0    permanent
  gr-ifd                                0            1           0    permanent
  macsec                                0            1           0    permanent
Licenses installed:
  License identifier: Junos942358
  License version: 4
  Valid for device: NV0216340092
  Features:
    extended-feature-list - Licensed extended feature list
      permanent
  License identifier: Junos942854
  License version: 4
  Valid for device: NV0216340092
  Features:
    macsec-feature-list - Licensed MACsec feature
      permanent
```

Do this for the rest of the devices and you're ready to start configuring the interfaces for MACSec.

Configure MACSec using Static Connectivity Association Key (SCAK)

First configure SCAK between the EX4300 and the EX4600. You can configure either Connectivity Association Key (CAK) or Secure Association Key (SAK) on point-to-point interfaces but it is best practice and recommended that you enable CAK on switch-to-switch links.

BEST PRACTICES Static CAK security mode ensures security by frequently refreshing to a new random Secure Association Key (SAK) and by only sharing the key between the two devices on the MACSec secured link. For more information, see: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/macsec.html#jd0e239.

All of the MACSec configuration is accomplished within the connectivity association and then two separate channels are created to carry the secure keys. Each one of the channels, one for inbound, and one for outbound, are automatically created and have no configurable parameters.

First set the `connectivity-association`:

```
EX4300-6_7-VC# set security macsec connectivity-association ex4300-to-ex4600
```

Then, set the security mode to `static-cak` as a best practice. `Dynamic` and `static-sak` are the other options:

```
EX4300-6_7-VC# set security macsec connectivity-association ex4300-to-ex4600 security-mode static-
cak
```

Once you have the `security-mode` you need to configure a CKN which is 64 hexidecimal digits:

```
EX4300-6_7-VC# set security macsec connectivity-association ex4300-to-ex4600 pre-shared-
key ckn 080520171913c2c45ddd012aa5bc8ef284aa23ff6729ee2e4acb66e91fe34ba2
```

You also need a pre-shared CAK which is 32 hexidecimal digits:

```
EX4300-6_7-VC# set security macsec connectivity-association ex4300-to-ex4600 pre-shared-
key cak 080520171913c2c45ddd012aa5bc8ef2
```

NOTE The CKN and CAK have to match exactly on both sides of the connection.

The MKA `key-server-priority` provides a way to determine which switch is going to be the main `key-server`. Setting the priority to 0 will insure that switch as master `key-server`. The default priority is 16.

```
EX4300-6_7-VC# set security macsec connectivity-association ex4300-to-ex4600 mka key-server-
priority 16
```

Once the link is up and MACSec is established, all traffic between these connections is encrypted. There is an option to set the interface to clear text if traffic needs to be viewed for troubleshooting (use keyword `no-encryption` after the CA Name).

```
EX4600-4_5-VC# set security macsec connectivity-association EX4300-to-EX4600 security-mode static-
cak
EX4600-4_5-VC# set security macsec connectivity-association EX4300-to-EX4600 pre-shared-
key ckn 080520171913c2c45ddd012aa5bc8ef284aa23ff6729ee2e4acb66e91fe34ba2
EX4600-4_5-VC# set security macsec connectivity-association EX4300-to-EX4600 pre-shared-
key cak 080520171913c2c45ddd012aa5bc8ef2
EX4600-4_5-VC# set security macsec connectivity-association EX4300-to-EX4600 mka key-server-
priority 0
```

When connecting to an EX4300 switch other switches have to enable the keyword `include-sci` an 8-octet long tag that is required for EX4300s. This is significant

overhead on an Ethernet interface so make sure it is only included on interfaces that connect to EX4300s:

```
EX4600-4_5-VC# set security macsec interfaces xe-1/0/0 connectivity-association EX4300-to-EX4600
```

Once the interface has been associated to an interface, and a corresponding switch has the same pre-shared keys, all traffic between those two switches on that ethernet segment is encrypted:

```
EX4300-6_7-VC# set security macsec interfaces xe-1/2/1 connectivity-association ex4300-to-ex4600

EX4300-6_7-VC# commit
```

To check the status of the connection, use the `show security macsec connections` command:

```
EX4600-4_5-VC# run show security macsec connections
    Interface name: xe-1/0/0
        CA name: EX4300-to-EX4600
        Cipher suite: GCM-AES-128   Encryption: on
        Key server offset: 0        Include SCI: yes
        Replay protect: off         Replay window: 0
          Outbound secure channels
            SC Id: 64:64:9B:5F:F9:03/1
            Outgoing packet number: 22
            Secure associations
            AN: 0 Status: inuse Create time: 00:00:26
          Inbound secure channels
            SC Id: 4C:96:14:E4:66:20/1
            Secure associations
            AN: 0 Status: inuse Create time: 00:00:26

EX4300-6_7-VC# run show security macsec connections
    Interface name: xe-1/2/1
        CA name: EX4300-to-EX4600
        Cipher suite: GCM-AES-128   Encryption: on
        Key server offset: 0        Include SCI: yes
        Replay protect: off         Replay window: 0
          Outbound secure channels
            SC Id: 4C:96:14:E4:66:20/1
            Outgoing packet number: 27
            Secure associations
            AN: 0 Status: inuse Create time: 00:01:23
          Inbound secure channels
            SC Id: 64:64:9B:5F:F9:03/1
            Secure associations
            AN: 0 Status: inuse Create time: 00:01:23

EX4600-4_5-VC# run ping 10.1.0.4 count 50 rapid source 10.1.0.5
PING 10.1.0.4 (10.1.0.4): 56 data bytes
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
--- 10.1.0.4 ping statistics ---
50 packets transmitted, 50 packets received, 0% packet loss
round-trip min/avg/max/stddev = 5.019/12.434/89.941/11.138 ms
```

After connectivity, run a quick ping to see that the packets are being protected by MACSec on the interface statistics:

```
EX4600-4_5-VC# run show security macsec statistics interface xe-1/0/0
    Secure Channel transmitted
        Encrypted packets: 434
        Encrypted bytes:   33639
        Protected packets: 0
        Protected bytes:   0
    Secure Association transmitted
        Encrypted packets: 434
        Protected packets: 0
    Secure Channel received
        Accepted packets:  183
        Validated bytes:   0
        Decrypted bytes:   21102
    Secure Association received
        Accepted packets:  183
        Validated bytes:   0
        Decrypted bytes:   0
```

MACSec statistics are also visible in the interface `detail` and `extensive` output:

```
EX4600-4_5-VC# run show interfaces xe-1/0/0 detail
Physical interface: xe-1/0/0, Enabled, Physical link is Up
  Interface index: 651, SNMP ifIndex: 525, Generation: 143
  Link-level type: Ethernet, MTU: 1514, MRU: 0, Speed: 10Gbps, BPDU Error: None, MAC-REWRITE Error:
None, Loopback: Disabled, Source filtering: Disabled, Flow control: Disabled, Media type: Fiber
… Truncated for brevity
  MACSec statistics:
    Output
        Secure Channel Transmitted
        Protected Packets               : 0
        Encrypted Packets               : 525
        Protected Bytes                 : 0
        Encrypted Bytes                 : 39615
     Input
        Secure Channel Received
        Accepted Packets                : 201
        Validated Bytes                 : 0
        Decrypted Bytes                 : 23373
```

As you can see you are able to use the controlled jinstall package to enable MACSec and with a very easy configuration stand up MACSec on our link between the EX4600 and the EX4300 in the lab.

Summary

This is as far as we are going to explore MACsec. We have shown the `best-practice` method and provided the entire process of securing the Ethernet segment between two switches. There are other options for configuration to include dynamic and Static Secure Association Key (SAK). For more information on the other details on MACsec and the other configuration options, as well as allowing specific ports and protocols to be unprotected, please see these important TechLibrary topics: https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/macsec.html#jd0e239.

Chapter 14

Configuring EX Series with Automation

Configuring NETCONF for API Access

Network Configuration Protocol (NETCONF) is used to carry Remote Procedure Calls (RPC) and XML traffic across a secure shell session using either port 22 or port 830. This secure transport mechanism makes it possible to remotely manage an entire network through Application Programmable Interfaces or APIs. This provides an enormous amount of possibilities on how you can mine data from the network, create automated intelligence for your network (think Cyber Security), perform basic operational and maintenance task on the network, and provide automated Change Management (CM). The possibilities are almost endless.

For this section, it's how to configure your devices for NETCONF and then we'll point you at Jonathan Looney and Stacy Smith's book *Automating Junos Networks* (published 2016, by O'Reilly Media). We can't say enough about how amazing that book really is and how valuable it is to a network engineer. It's part of the *Juniper Books* library, so if you don't already have it, you should rush out and get it now: https://www.juniper.net/us/en/training/jnbooks/oreilly-juniper-library/automating-junos-admin/.

MORE? You can also get more information on NETCONF by reading RFC6241 since this is an IETF standardized protocol.

Let's go ahead and configure the Junos devices for NETCONF so you can move on to the fun stuff like PyEZ and Ansible. The configuration is the same across all platforms and this is just the minimal configuration that you need to support the NETCONF protocol.

Place the following on all lab devices:

```
QFX5100-2_3-VC# set system services netconf ssh ?
Possible completions:
  <[Enter]>            Execute this command
+ apply-groups         Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
  connection-limit     Maximum number of allowed connections (1..250)
  port                 Service port number (1..65535)
  rate-limit           Maximum number of connections per minute (1..250)
  |                    Pipe through a command
```

If you want to use the default NETCONF port of 830 you need to specify the port, otherwise if you stop at ssh, it will use port 22 by default. To simplify and reduce the number of ports through the firewallm let's stay with port 22 and stop at ssh:

```
{master:0}[edit]
QFX5100-2_3-VC# set system services netconf ssh
{master:0}[edit]
QFX5100-2_3-VC# commit and-quit
configuration check succeeds
fpc1:
commit complete
commit complete
Exiting configuration mode
{master:0}
QFX5100-2_3-VC>
```

You need to do this for all of the devices you want to be able to manage via NETCONF. This same command has been applied to all devices in the lab supporting this guide.

Once you have NETCONF enabled you can test it by using PyEZ which is our next topic and we'll show you how to verify.

Installing and Using PyEZ

Before diving into the configuration it's important to know Juniper maintains the official PyEZ documentation at the following URL: http://www.juniper.net/documentation/en_US/release-independent/junos-pyez/information-products/pathway-pages/index.html.

And directly from that page you can see that PYEZ is:

"Junos PyEZ is a microframework for Python that enables you to manage and automate devices running the Junos operating system (Junos OS). Junos PyEZ is designed to provide the capabilities that a user would have on the Junos OS command-line interface (CLI) in an environment built for automation tasks. Junos PyEZ does not require extensive knowledge of Junos OS or the Junos XML APIs."

So, there you have it! PyEZ enables automation and does not require extensive knowledge of Junos, XML, or APIs.

The documentation also provides a very thorough walk through of the installation of PyEZ and identifies the dependencies as well, so we are going to point you at that page here: http://www.juniper.net/documentation/en_US/junos-pyez/topics/task/installation/junos-pyez-server-installing.html.

MORE? There's a new *Day One: Junos PyEZ Cookbook* coming. Look for it on the Juniper Books landing page: http://www.juniper.net/books.

Now, once you have completed the install and have all the required dependencies you can test NETCONF and PyEZ at the same time.

For a very simple test of using PyEZ over NETCONF to connect to the Junos devices in the lab and poll the device, use the Python shell (you have to have Python to load PyEZ, if you want to follow along). Ignore the (1-7) numerals in the following these are for reference only and should *not* cut and pasted.

```
 (1)
$ python
Python 2.7.5 (default, Nov  6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(2)
>>> from jnpr.junos import Device
(3)
>>> from pprint import pprint
>>>
(4)
>>> EX4300 = Device(host='172.16.0.6',user='lab',password='jnpr123')
>>>
(5)
>>> EX4300.open()
Device(172.16.0.6)
>>>
(6)
>>> pprint(EX4300.facts)
{'2RE': True,
 'HOME': '/var/home/lab',
 'RE0': {'last_reboot_reason': '0x1:power cycle/failure',
         'mastership_state': 'backup',
         'model': 'EX4300-24T',
         'status': 'OK',
         'up_time': '18 days, 19 minutes, 21 seconds'},
 'RE1': {'last_reboot_reason': '0x1:power cycle/failure',
         'mastership_state': 'master',
         'model': 'EX4300-24T',
         'status': 'OK',
         'up_time': '18 days, 19 minutes, 16 seconds'},
 'RE_hw_mi': False,
 'current_re': ['master',
                'node',
                'fwdd',
```

```
                'member',
                'pfem',
                're0',
                'fpc1',
                'feb0',
                'fpc16'],
 'domain': None,
 'fqdn': 'EX4300-6_7-VC',
 'hostname': 'EX4300-6_7-VC',
 'hostname_info': {'fpc0': 'EX4300-6_7-VC', 'fpc1': 'EX4300-6_7-VC'},
 'ifd_style': 'SWITCH',
 'junos_info': {'fpc0': {'object': junos.version_
info(major=(14, 1), type=X, minor=(53, 'D', 40), build=8),
                        'text': '14.1X53-D40.8'},
                'fpc1': {'object': junos.version_
info(major=(14, 1), type=X, minor=(53, 'D', 40), build=8),
                        'text': '14.1X53-D40.8'}},
 'master': 'RE1',
 'model': 'EX4300-24T',
 'model_info': {'fpc0': 'EX4300-24T', 'fpc1': 'EX4300-24T'},
 'personality': 'SWITCH',
 're_info': {'default': {'0': {'last_reboot_reason': '0x1:power cycle/failure',
                               'mastership_state': 'backup',
                               'model': 'EX4300-24T',
                               'status': 'OK'},
                         '1': {'last_reboot_reason': '0x1:power cycle/failure',
                               'mastership_state': 'master',
                               'model': 'EX4300-24T',
                               'status': 'OK'},
                         'default': {'last_reboot_reason': '0x1:power cycle/failure',
                                     'mastership_state': 'backup',
                                     'model': 'EX4300-24T',
                                     'status': 'OK'}}},
 're_master': {'default': '1'},
 'serialnumber': 'PG3713290002',
 'srx_cluster': None,
 'switch_style': 'VLAN_L2NG',
 'vc_capable': True,
 'vc_fabric': None,
 'vc_master': '1',
 'vc_mode': 'Enabled',
 'version': '14.1X53-D40.8',
 'version_RE0': None,
 'version_RE1': None,
 'version_info': junos.version_info(major=(14, 1), type=X, minor=(53, 'D', 40), build=8),
 'virtual': False}
(7)
>>> EX4300.close()
```

There you go! You received a plethora of valuable information from the device with just the simple facts that were gathered from the initial connection. Now you have verified that you have NETCONF configured on the device and that your Python platform has PyEZ installed and that it is all working!

Let's discuss some of the finer points of what happened, using those numeral indicators:

1. Started the Python Shell by typing python at the $ prompt.
2. Imported jnpr-junos Device (which is one of many PyEZ tables to work with).
3. Imported pprint to format the output (Pretty Print is a python package).
4. Provided the IP, User, and Password for the variable EX4300 (the variable can be anything you want).
5. With the credentials from step 4, initiate the open() function that actually connects to the device using NETCONF. If successful, you will get a reply Device (172.16.0.6) showing that you are connected to the device through that IP. It also gathered the device facts when it connected.
6. Now you can use pprint to print the device facts that were gathered from the initial connection using pprint (EX4300.facts).
7. And finally, close the connection using the EX4300.close() function.

Again, the Automating Junos Administration book is a very good source for all of this, and it's the reason why we are only touching the surface here. Please don't let this be the end of your experience with Python and PyEZ.

Installing and Using Ansible

Ansible Core is an open source product. It takes very little to get it up and running and the Automating Junos Administration book is, once again, an awesome source on walking you through the complete setup and even provides sample working solutions.

What we like most about Ansible is the following:

- Ansible is easy.
- Ansible doesn't require a client like Puppet or Chef (which makes it easy!).
- Ansible works! Once you have Netconf enabled and PyEZ working, you are 90% there.

Jinja2 templates allow you to reference variables.

If you run through the Automating Junos Administration book you will get a true appreciation for Ansible. Especially if you do the Puppet and Chef chapters first! For now, let's do a quick demo on how Ansible works by showing you the yaml file created to pull the set-based configurations from every device in the lab, then run it from the command line, and then show you the results of the Ansible script. Okay, let's do this!

```
$ more ser_grab_set_config.pb.yaml
---
- name: Junos CLI commands
```

```
  hosts: "{{ target }}"
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos
  vars:
    ask_vault_pass: False
    netconf_user:
    netconf_password:
    default_port: 22
    config_dir: CONFIGS
  tasks:
    - name: GET CONFIGURATION FROM {{ target }}
      junos_cli:
        host: "{{ inventory_hostname }}"
        cli: "show configuration | display set"
        logfile: "cli.log"
        dest: "{{ config_dir }}/{{ inventory_hostname }}.setconfig.txt"
        format: text
```

Now this is a very simple yaml file. Many are more complex files that have numerous tasks that go on for pages. Tip: try to keep things simple so your tasks are broken out to individual files and if you need to call them you can simply pull them in if needed. The most important thing about creating this file is keeping the formatting and indentation. The section above is the complete script. No programming knowledge required other than setting the Jinja Variable and that's a low learning curve.

It should be noted that this is using RSA Secure Shell keys so there is no prompts to enter a password when logging into the devices (covered in the previous chapter). The `hosts: or {{ target }}` field utilizes the inventory file that looks like this:

```
$ more ser.inv
[qfx5100]
172.16.0.2
[ex4600]
172.16.0.4
[ex4300]
172.16.0.6
[ex2300]
172.16.0.8
[ex3400]
172.16.0.10
[srx345-12]
172.16.0.12
[srx345-13]
172.16.0.13
[nfx250]
172.16.0.14
[ex2300]
172.16.0.16
[lab-switches]
172.16.0.2
172.16.0.4
172.16.0.6
```

```
172.16.0.8
172.16.0.10
172.16.0.16
```

With the Ansible script/YAML file and the inventory, you just need to run the script and pass in the variables to pull all of the configs simultaneously for your entire lab. Yes - you read that right - this will pull them all at once, not in a serial fashion. This is possible because it forks the processes. Let's show off:

```
$ ansible-playbook -i ser.inv ser_grab_set_config.pb.yaml --extra-vars "target=lab-switches"
PLAY [Junos CLI commands] ***************************************************
TASK [GET CONFIGURATION FROM lab-switches] ********************************
ok: [172.16.0.2]
ok: [172.16.0.4]
ok: [172.16.0.8]
ok: [172.16.0.16]
ok: [172.16.0.10]
ok: [172.16.0.6]
PLAY RECAP ******************************************************************
172.16.0.10                : ok=1    changed=0    unreachable=0    failed=0
172.16.0.16                : ok=1    changed=0    unreachable=0    failed=0
172.16.0.2                 : ok=1    changed=0    unreachable=0    failed=0
172.16.0.4                 : ok=1    changed=0    unreachable=0    failed=0
172.16.0.6                 : ok=1    changed=0    unreachable=0    failed=0
172.16.0.8                 : ok=1    changed=0    unreachable=0    failed=0
```

That script pulled all the configs in under five seconds! And now they are stored in the specified directory "CONFIGS" which was specified under –vars in the yaml file. Let's make sure they are there:

```
$ ls *.txt
172.16.0.10.setconfig.txt
172.16.0.16.setconfig.txt
172.16.0.2.setconfig.txt
172.16.0.4.setconfig.txt
172.16.0.6.setconfig.txt
172.16.0.8.setconfig.txt
```

And let's also take a look inside to make sure that they were indeed captured:

```
$ cat 172.16.0.* | more
set version 15.1X53-D50.2
set groups POC_Lab system host-name EX3400-10_11-VC
set groups POC_Lab system backup-router 172.16.0.1
set groups POC_Lab system authentication-order radius
set groups POC_Lab system authentication-order password
```

We can't stress enough just how easy Ansible is to use. Not only can you mine data from the network but you can also push changes to your network and automate your configuration management for your entire organization with Ansible. It's very powerful and easy to use.

MORE? Check the Juniper Books' web pages for more books on PyEZ and Ansible that are in development as this book goes to press: http://www.juniper.net/books.

Installing and using JAIDEGUI

A cool little python application created by Geoff Rhodes and Nathan Printz is called JAIDEGUI (Junos Aide (JAIDE) and JGUI). We were demonstrating some home-grown python scripts that do the exact same thing, but they don't have a cool GUI, so hats off to Geoff and Nathan for this cool tool!

NetworkAutomation/jaidegui is licensed under the GNU General Public License v2.0 making it available to anyone who wants to download it from: https://github.com/NetworkAutomation/jaidegui.

Download the zip file and then issue `$ python setup.py install` and then you can run the script. Now if you already have Python and pip installed its even easier than that:

```
$ sudo pip install jaidegui
The directory '/Users/sreger/Library/Caches/pip/http' or its parent directory is not owned by the
current user and the cache has been disabled. Please check the permissions and owner of that directory.
If executing pip with sudo, you may want sudo's -H flag.
You are using pip version 7.1.0, however version 9.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
The directory '/Users/sreger/Library/Caches/pip/http' or its parent directory is not owned by the
current user and the cache has been disabled. Please check the permissions and owner of that directory.
If executing pip with sudo, you may want sudo's -H flag.
Requirement already satisfied (use --upgrade to upgrade): jaidegui in /Library/Python/2.7/site-
packages/jaidegui-1.0.0-py2.7.egg
Collecting jaide>=2.0.0 (from jaidegui)
  Downloading jaide-2.0.0-py2-none-any.whl
Requirement already satisfied (use --upgrade to upgrade): Pmw>=1.3.3 in /Library/Python/2.7/site-
packages/Pmw-2.0.1-py2.7.egg (from jaidegui)
Requirement already satisfied (use --upgrade to upgrade): ncclient>=0.4.2 in /Library/Python/2.7/
site-packages (from jaide>=2.0.0->jaidegui)
Requirement already satisfied (use --upgrade to upgrade): ecdsa>=0.11 in /Library/Python/2.7/site-
packages (from jaide>=2.0.0->jaidegui)
Requirement already satisfied (use --upgrade to upgrade): pycrypto!=2.4,>=2.1 in /Library/Python/2.7/
site-packages (from jaide>=2.0.0->jaidegui)
Requirement already satisfied (use --upgrade to upgrade): paramiko<2.0.0,>=1.14.0 in /Library/
Python/2.7/site-packages (from jaide>=2.0.0->jaidegui)
Requirement already satisfied (use --upgrade to upgrade): colorama>0.3.2 in /Library/Python/2.7/
site-packages (from jaide>=2.0.0->jaidegui)
Collecting click<5.0.0,>=4.0.0 (from jaide>=2.0.0->jaidegui)
  Downloading click-4.1-py2.py3-none-any.whl (62kB)
    100% |████████████████████████████████| 65kB 316kB/s
Requirement already satisfied (use --upgrade to upgrade): scp<1.0.0,>=0.8.0 in /Library/Python/2.7/
site-packages (from jaide>=2.0.0->jaidegui)
Requirement already satisfied (use --upgrade to upgrade): setuptools>0.6 in /Library/Python/2.7/
site-packages (from ncclient>=0.4.2->jaide>=2.0.0->jaidegui)
Requirement already satisfied (use --upgrade to upgrade): lxml>=3.3.0 in /Library/Python/2.7/site-
```

```
packages (from ncclient>=0.4.2->jaide>=2.0.0->jaidegui)
Requirement already satisfied (use --upgrade to upgrade): six in /System/Library/Frameworks/Python.
framework/Versions/2.7/Extras/lib/python (from ncclient>=0.4.2->jaide>=2.0.0->jaidegui)
Installing collected packages: click, jaide
Successfully installed click-4.1 jaide-2.0.0
```

Now, launch this tool from the CLI and use it as a CLI tool, or launch the JAIDE GUI. First, let's see the help file for jaide on the command line:

```
$ jaide
Usage: jaide [OPTIONS] COMMAND [ARGS]...
  Manipulate one or more Junos devices.
  Will connect to one or more Junos devices, and manipulate them based on the command you have chosen.
If a comma separated list or a file containing IP/hostnames on each line is given for the IP
  option, the commands will be sent simultaneously to each device.
Options:
  -i, --ip TEXT                The target hostname(s) or IP(s). Can be a comma separated list, or path
to a file listing devices on individual lines.
  -u, --username TEXT
  -p, --password TEXT
  -P, --port INTEGER            The port to connect to. Defaults to SSH (22)
  --quiet / --no-quiet          Boolean flag to show no output, except in certain error scenarios.
Defaults to false (--no-quiet), which shows the output.
  -t, --session-timeout INTEGER RANGE
                               The session timeout value, in seconds, for declaring a lost session.
Default is 300 seconds. This should be increased when no output could be seen for more than 5
                               minutes (ex. requesting a system snapshot).
  -T, --connect-timeout INTEGER RANGE
                               The timeout, in seconds, for declaring a device unreachable during
connection establishment. Default is 5 seconds.
  --version                     Show the version and exit.
  -w, --write [s | single | m | multiple] FILEPATH
                               Write the output to a file instead of echoing it to the terminal. This can be
useful when touching more than one device, because the output can be split into a file per
                               device. In this case, output filename format is IP_FILENAME.
  -h, --help                    Show this message and exit.
Commands:
  commit       Execute a commit against the device.
  compare      Compare commands against running config.
  diff_config  Compare the config between two devices.
  errors       Get any interface errors from the device.
  health       Get alarm and device health information.
  info         Get basic device information.
  operational  Execute operational mode command(s).
  pull         Copy file(s) from device(s) -> local machine.
  push         Copy file(s) from local machine -> device(s).
  shell        Send shell commands to the device(s).
```

The second option is to use it as a GUI. To launch that simply type: $ `jaidegui&` and then you get something similar to Figure 14.1.

If you have Python and PIP installed, you could have an easy open-source scripting tool at your fingertips in just a matter of minutes. There are dependencies, as outlined on Git. Let's run it on the lab network and pull the hardware from all of the devices.

NOTE This is where the CLI option versus the GUI comes in handy. While we do have VNC access to our lab, it is over 800 miles away, so being able to simply run this from the CLI is a plus!

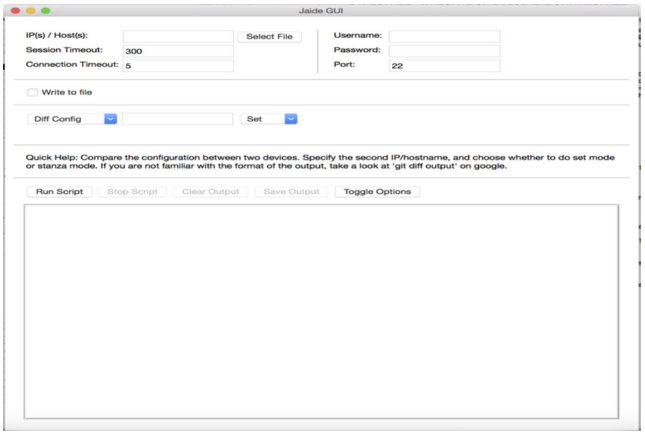


Figure 14.1 *JaideGUI Interface.*

First, build a file called IPLIST and place all the management IP's in the file. You can also do this with comma separated IPs. Then create a file with all the show commands you want to run. Here is a preview of the files:

```
$ more IPLIST
172.16.0.2
172.16.0.4
172.16.0.6
172.16.0.8
172.16.0.10
172.16.0.12
172.16.0.13
172.16.0.16
$ more JAIDECOMMANDS
show chassis hardware | no-more
show virtual-chassis
show configuration | display set | no-more
show interfaces terse
show route
```

And now to run this from the CLI, run the following and watch the feedback on the screen. Using the `-w m` and specifying an extension will also place each switch output in its own file for easy access:

```
$ jaide -i IPLIST -u lab -p jnpr123 -w m jaide.txt operational JAIDECOMMANDS
172.16.0.4 output appended to: 172.16.0.4_jaide.txt
172.16.0.2 output appended to: 172.16.0.2_jaide.txt
172.16.0.8 output appended to: 172.16.0.8_jaide.txt
```

```
172.16.0.16 output appended to: 172.16.0.16_jaide.txt
172.16.0.12 output appended to: 172.16.0.12_jaide.txt
172.16.0.10 output appended to: 172.16.0.10_jaide.txt
172.16.0.6 output appended to: 172.16.0.6_jaide.txt
172.16.0.13 output appended to: 172.16.0.13_jaide.txt
```

This went very quickly and efficiently and ran all the commands without error! Look at the output:

```
$ ls *jaide.txt
172.16.0.10_jaide.txt  172.16.0.13_jaide.txt  172.16.0.2_jaide.txt  172.16.0.6_jaide.txt
172.16.0.12_jaide.txt  172.16.0.16_jaide.txt  172.16.0.4_jaide.txt  172.16.0.8_jaide.txt

$ more 172.16.0.10_jaide.txt
================================================
Results from device: 172.16.0.10

> show chassis hardware | no-more
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                NV0216340092      Virtual Chassis
Routing Engine 0          BUILTIN      BUILTIN           RE-EX3400-24T
FPC 0            REV 10   650-059888   NV0216340092      EX3400-24T
  CPU                     BUILTIN      BUILTIN           FPC CPU
  PIC 0          REV 10   BUILTIN      BUILTIN           24x10/100/1000 Base-T
  PIC 1          REV 10   650-059888   NV0216340092      2x40G QSFP
    Xcvr 0       REV 01   740-038623   MOC15526232233    QSFP+-40G-CU1M
    Xcvr 1       REV 01   740-038623   MOC15306230822    QSFP+-40G-CU1M
  PIC 2          REV 10   650-059888   NV0216340092      4x10G SFP/SFP+
    Xcvr 0       REV 01   740-021309   AWJ0DJT           SFP+-10G-LR
    Xcvr 2       REV 01   740-021309   AWJ0620           SFP+-10G-LR
  Power Supply 0 REV 05   640-060603   1EDV6311841       JPSU-150W-AC-AFO
  Fan Tray 0                                             Fan Module, Airflow Out (AFO)
  Fan Tray 1                                             Fan Module, Airflow Out (AFO)

> show virtual-chassis
Preprovisioned Virtual Chassis
Virtual Chassis ID: 9370.5006.089a
Virtual Chassis Mode: Enabled
                                                 Mstr           Mixed Route Neighbor List
Member ID  Status   Serial No    Model          prio  Role      Mode  Mode ID  Interface
0 (FPC 0)  Prsnt    NV0216340092 ex3400-24t     129   Master*      N  VC
1 (FPC 1)  NotPrsnt NV0216340111

> show configuration | display set | no-more
set version 15.1X53-D50.2
set groups POC_Lab system host-name EX3400-10_11-VC
set groups POC_Lab system backup-router 172.16.0.1
set groups POC_Lab system authentication-order radius
set groups POC_Lab system authentication-order password
… Truncated for brevity
```

And for the GUI it's the same as that shown in Figure 14.2. Super cool and easy app.

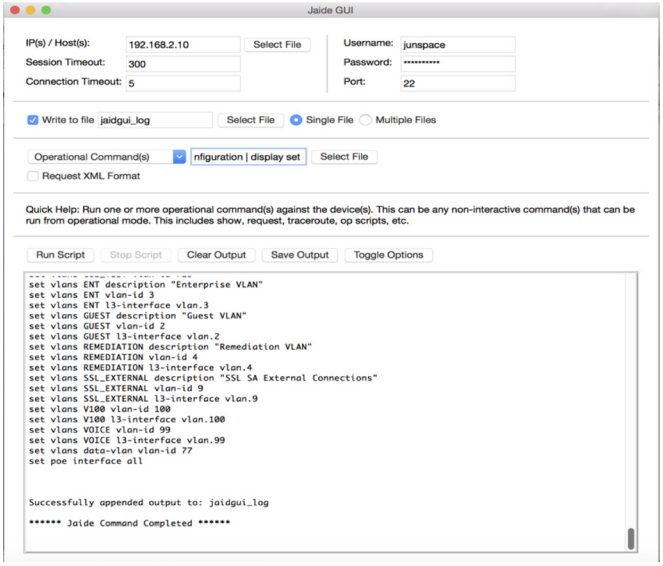


Figure 14.2 *JaideGUI Interface in Operation.*

Summary

Automation can assist with mining data from your network, archiving configs updating your network, and all around Configuration Management (CM) possibilities for your networks. PyEZ allows you to access any part of the Junos configuration, Ansible is an awesome tool for automation and then if you are unable to download Python and Ansible you can always use the executable JAIDEGUI from the Git repository. Whatever you do not forget to buy and read the Automating Junos Administration book written by Jonathan Looney and Stacy Smith, and look for more great automation books in the Juniper Books library at: http://www.juniper.net/books.

Chapter 15

Configuring QFX Series with Vagrant

What is vQFX

The vQFX, or virtualized QFX, is a vagrant packaged virtualized deployment of Juniper Networks QFX10000 series. Why introduce the vQFX in an EX *Day One* book, the answer is simple - One Junos! The fact that all of our products run the same world class Junos OS, and the fact that the QFX is an awesome data center and service provider switch, it has all the same features as the EX Series plus numerous features from the MX Series of routers that currently dominate the service provider environments.

In addition, the vQFX vagrant deployment provides you with a full four node topology with both control and forwarding planes, or you can go down to a simple two-node light deployment which is control plane only.

With either deployment, you will be able to get hands-on time with the Junos CLI and be able to lab up different topologies that you can build real world configurations upon, test features, and run most of the commands included in this *Day One* book. Virtual Chassis features are not available with the vQFX.

The Junos OS has been disaggregated from hardware for a long time now. You can actually download the vSRX, vMX, and Junos SPACE virtual appliances for 60-day evaluations and run an entire network right on your laptop or server. To download the virtualized products, you will need a Juniper authorized account. The vQFX vagrant is open and available from the git repository:

vQFX: https://github.com/Juniper/vqfx10k-vagrant

vSRX: http://www.juniper.net/support/downloads/?p=vsrx#sw

vMX: http://www.juniper.net/support/downloads/?p=vmx#sw

Junos Space: http://www.juniper.net/support/downloads/?p=space#sw

Make sure that you read the CPU and RAM requirements for each of these deployments. Again, Junos is Junos, so once you learn the syntax and how to navigate the Junos CLI you are now able to do so on all of our platforms (unlike some vendors where command sets are very different from one platform to another).

Installing vQFX

The vQFX is deployed using Vagrant, VirtualBox, and Ansible. All of these are freely available.

- Vagrant: https://www.vagrantup.com/
- VirtualBox: https://www.virtualbox.org/
- Ansible: https://www.ansible.com

You want to make sure that PyEZ and Netconify are installed before Ansible, and these are all accomplished from the CLI using Python.

First let's install the PyEZ and Netconify packages along with Ansible using the CLI from your laptop.

NOTE This procedure is being performed on a Mac Book Pro from a terminal window but is the same for CentOS. If you are using a Windows machine the commands may vary slightly.

NOTE If your XCode is not updated on your Macintosh, then you may need to update it before loading PyEZ and the rest of the packages. Make sure you have Python and pip loaded.

```
#PyEZ
1) pip install junos-eznc
#NetConify
1) pip install junos-netconify
#Ansible
1) sudo pip install ansible
2) ansible --version
3) sudo ansible-galaxy install Juniper.junos
4) cd /etc/ansible/
5) vi ansible.cfg
       [defaults]
       log_path=~/.ansible/log/ansible.log
       forks=50
6) sudo usermod -a -G root username
7) sudo chmod g+w ansible.cfg
```

TIP With CentOS, you can run `$ sudo yum install ansible` to download the Ansible package.

Make sure you follow the installation instructions on the git repository to build the vagrant vQFX package: https://github.com/Juniper/vqfx10k-vagrant/blob/master/INSTALL.md.

Download and install VirtualBox along with the extension pack. The latest version is 5.1.24 (as of this book's writing). VirtualBox is a straight-forward install and it will support all of the virtualized products listed here. It also supports Single Root IO Virtualization or SR-IOV through Virtualization Technology for Directed I/O or VT-d. So as long as your host supports the SR-IOV, and VT-d is turned on, the Virtual Machine (VM) in the hypervisor, then you are good to go.

Once you have installed VirtualBox you should be able to launch the hypervisor.

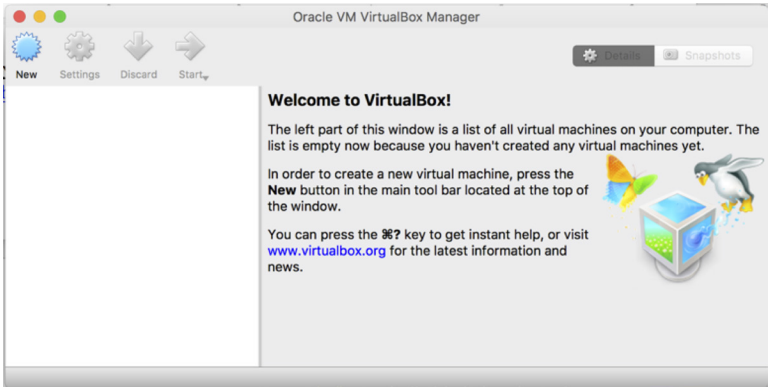


Figure 15.1 *Oracle VirtualBox Hypervisor.*

Make sure you load the extension package as well. If you already have VirtualBox downloaded and operating, you can download the extension pack and it will open with VBox. Once loaded you should be able to locate it under VirtualBox -> Preferences -> Extensions.

So far you have PyEZ, Netconify, Ansible, and VirtualBox and now you need to load Vagrant.

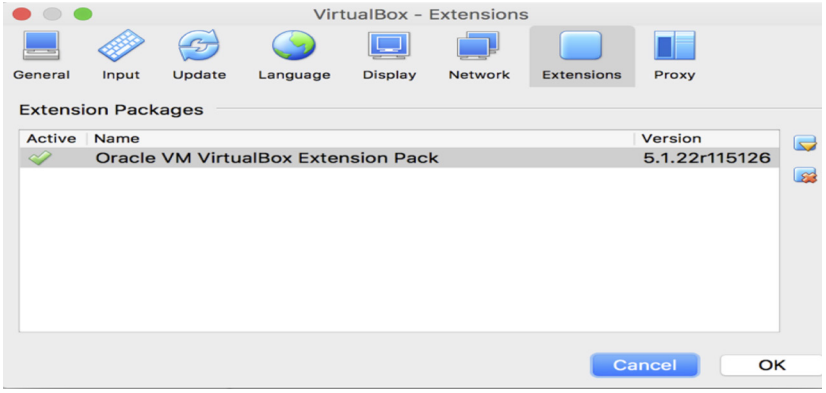


Figure 15.2 *VirtualBox Extensions interface.*

On CentOS 7 you can run the following as root or sudo:

```
# yum –y install https://releases.hashicorp.com/vagrant/1.9.7/vagrant_1.9.7_x86_64.rpm
```

On Mac you can download the vagrant .dmg file and install it from: https://releases.hashicorp.com/vagrant/1.9.7/vagrant_1.9.7_x86_64.dmg

On Windows devices you can download and run the msi: https://releases.hashicorp.com/vagrant/1.9.7/vagrant_1.9.7_x86_64.msi

Very simple installs on all three platforms!

Now that you have all of the pre-installation requirements for vQFX, you can download the actual vQFX git package and install it on your platform. For this section, we're using a MacBook Pro, so for Windows and Linux users, please follow the instructions from the git repository. There are plenty of *How To* guides on the internet that can walk you through your specific setup, too.

```
$ git clone https://github.com/Juniper/vqfx10k-vagrant.git
Cloning into 'vqfx10k-vagrant'...
remote: Counting objects: 387, done.
remote: Total 387 (delta 0), reused 0 (delta 0), pack-reused 387
Receiving objects: 100% (387/387), 68.71 KiB | 0 bytes/s, done.
Resolving deltas: 100% (180/180), done.
```

That went super fast and now there should be a vqfx directory so let's find that.

```
$ ls -al | grep vqfx
drwxr-xr-x   17 lab staff      578 Jul 26 10:59 vqfx10k-vagrant
```

Now everything is in place: PyEz, Netconify, Ansible, Vagrant, and our vQFX packages. The only thing left to do is to launch the vQFX using vagrant and watch the success.

Launching vQFX

There are several pre-built topologies in the vQFX package.

Available examples are:

- full-1qfx
- full-1qfx-1srv
- full-2qfx
- light-1qfx
- light-2qfx
- light-2qfx-2srv
- light-ipfabric-2S-3L

For this exercise, let's launch the full-2qfx. Full topologies include a control plane and a forwarding plane, whereas light topologies only include the control plane. Each of the topologies have specific RAM and CPU requirements, so make sure that you go to: https://github.com/Juniper/vqfx10k-vagrant and review the available topologies and that your host platform meets or exceeds those requirements.

With the downloaded package and verification that the vqfx10k-vagrant directory exists, the only thing left to do is navigate to our desired topology and start the vagrant environment:

```
$ cd vqfx10k-vagrant/
lab-mbp:vqfx10k-vagrant lab$ ls
INSTALL.md   TROUBLESHOOTING.md full-2qfx  light-1qfx   light-ipfabric-2S-3L
LICENSE      full-1qfx    full-2qfx-4srv-evpnvxlan  light-2qfx
README.md    full-1qfx-1srv   full-4qfx  light-2qfx-2srv

$ cd full-2qfx
lab-mbp:full-2qfx lab$ ls
README.md  Vagrantfile ansible.cfg  host_vars  pb.conf.all.commit.yaml pb.conf.save.yamlvqfx.conf.j2
```

Once parked in the full-2qfx directory, launch the topology:

```
$ vagrant up
Bringing machine 'vqfx1-pfe' up with 'virtualbox' provider...
Bringing machine 'vqfx1' up with 'virtualbox' provider...
Bringing machine 'vqfx2-pfe' up with 'virtualbox' provider...
Bringing machine 'vqfx2' up with 'virtualbox' provider...
==> vqfx1-pfe: Box 'juniper/vqfx10k-pfe' could not be found. Attempting to find and install...
    vqfx1-pfe: Box Provider: virtualbox
    vqfx1-pfe: Box Version: >= 0
==> vqfx1-pfe: Loading metadata for box ‹juniper/vqfx10k-pfe›
    vqfx1-pfe: URL: https://vagrantcloud.com/juniper/vqfx10k-pfe
==> vqfx1-pfe: Adding box ‹juniper/vqfx10k-pfe› (v0.1.0) for provider: virtualbox
    vqfx1-pfe: Downloading: https://vagrantcloud.com/juniper/boxes/vqfx10k-pfe/versions/0.1.0/
providers/virtualbox.box
    vqfx1-pfe: Progress: 2% (Rate: 625k/s, Estimated time remaining: 0:16:25)
```

As you can see, it is downloading the Virtual Machines for this topology and it will be a few minutes. One of the things you need to do is actually read the README.md file – it has a lot of information that will help you when accessing the devices, etc. Here is a small exerpt that you will need later on when accessing the device. There is a ton of information in the README.md file so take the time to read it!

```
#### vqfx10k-pfe
Requires:
- 1.5/2GB of memory
- 1 dedicated core
A maximum of 2 interfaces are supported:
- first interface is used by vagrant (eth0)
- second interface is used to connect to the RE VM (eth1)
This VM has 2 account by default:
- login: vagrant, with ssh_key authentication using vagrant insecure_key
- login: root, password: no
And now our PFE is downloaded we see some more movement on the vagrant package.
==> vqfx1-pfe: Successfully added box 'juniper/vqfx10k-pfe' (v0.1.0) for 'virtualbox'!
==> vqfx1-pfe: Importing base box 'juniper/vqfx10k-pfe'...
==> vqfx1-pfe: Matching MAC address for NAT networking...
==> vqfx1-pfe: Checking if box 'juniper/vqfx10k-pfe' is up to date...
==> vqfx1-pfe: Setting the name of the VM: full-2qfx_vqfx1-pfe_1501086890774_88281
==> vqfx1-pfe: Clearing any previously set network interfaces...
==> vqfx1-pfe: Preparing network interfaces based on configuration...
    vqfx1-pfe: Adapter 1: nat
    vqfx1-pfe: Adapter 2: intnet
==> vqfx1-pfe: Forwarding ports...
    vqfx1-pfe: 22 (guest) => 2222 (host) (adapter 1)
==> vqfx1-pfe: Booting VM...
==> vqfx1-pfe: Waiting for machine to boot. This may take a few minutes...
    vqfx1-pfe: SSH address: 127.0.0.1:2222
    vqfx1-pfe: SSH username: vagrant
    vqfx1-pfe: SSH auth method: private key
==> vqfx1-pfe: Machine booted and ready!
==> vqfx1-pfe: Checking for guest additions in VM...
    vqfx1-pfe: No guest additions were detected on the base box for this VM! Guest
    vqfx1-pfe: additions are required for forwarded ports, shared folders, host only
    vqfx1-pfe: networking, and more. If SSH fails on this machine, please install
    vqfx1-pfe: the guest additions and repackage the box to continue.
    vqfx1-pfe:
    vqfx1-pfe: This is not an error message; everything may continue to work properly,
    vqfx1-pfe: in which case you may ignore this message.
==> vqfx1-pfe: Running provisioner: ansible...
    vqfx1-pfe: Running ansible-playbook...
==> vqfx1: Box 'juniper/vqfx10k-re' could not be found. Attempting to find and install...
    vqfx1: Box Provider: virtualbox
    vqfx1: Box Version: >= 0
==> vqfx1: Loading metadata for box 'juniper/vqfx10k-re'
    vqfx1: URL: https://vagrantcloud.com/juniper/vqfx10k-re
==> vqfx1: Adding box 'juniper/vqfx10k-re' (v0.2.0) for provider: virtualbox
    vqfx1: Downloading: https://vagrantcloud.com/juniper/boxes/vqfx10k-re/versions/0.2.0/providers/
virtualbox.box
```

This process takes a while for the very first run but will be very fast when subsequent launches are made because the VM's are already downloaded.

TIP Notice that we downloaded a vqfx10k-pfe and a vqfx10k-re. PFE stands for Packet Forwarding Engine and RE stands for Routing Engine. This shows how to completely separate the Junos OS control and forwarding planes and why it makes us so much more efficient at processing packets than our competitors.

```
==> vqfx1: Successfully added box 'juniper/vqfx10k-re' (v0.2.0) for 'virtualbox'!
==> vqfx1: Importing base box 'juniper/vqfx10k-re'...
==> vqfx1: Matching MAC address for NAT networking...
==> vqfx1: Checking if box 'juniper/vqfx10k-re' is up to date...
==> vqfx1: Setting the name of the VM: full-2qfx_vqfx1_1501087612356_59890
==> vqfx1: Fixed port collision for 22 => 2222. Now on port 2200.
==> vqfx1: Clearing any previously set network interfaces...
==> vqfx1: Preparing network interfaces based on configuration...
    vqfx1: Adapter 1: nat
    vqfx1: Adapter 2: intnet
    vqfx1: Adapter 3: intnet
    vqfx1: Adapter 4: intnet
    vqfx1: Adapter 5: intnet
    vqfx1: Adapter 6: intnet
    vqfx1: Adapter 7: intnet
    vqfx1: Adapter 8: intnet
    vqfx1: Adapter 9: intnet
==> vqfx1: Forwarding ports...
    vqfx1: 22 (guest) => 2200 (host) (adapter 1)
==> vqfx1: Booting VM...
==> vqfx1: Waiting for machine to boot. This may take a few minutes...
    vqfx1: SSH address: 127.0.0.1:2200
    vqfx1: SSH username: vagrant
    vqfx1: SSH auth method: private key
==> vqfx1: Machine booted and ready!
==> vqfx1: Checking for guest additions in VM...
    vqfx1: No guest additions were detected on the base box for this VM! Guest
    vqfx1: additions are required for forwarded ports, shared folders, host only
    vqfx1: networking, and more. If SSH fails on this machine, please install
    vqfx1: the guest additions and repackage the box to continue.
    vqfx1:
    vqfx1: This is not an error message; everything may continue to work properly,
    vqfx1: in which case you may ignore this message.
==> vqfx1: Setting hostname...
==> vqfx1: Running provisioner: ansible...
    vqfx1: Running ansible-playbook...
```

Okay, so we have successfully downloaded the PFE and RE image and it will launch those automatically into VirtualBox and use Ansible to place the configuration required for this specific topology on the switches. There is another series of the output, but its just more of the same for the second vqfx, so it isn't displayed.

At this point you should have successfully launched two full vqfx10ks (full meaning each vqfx is made up of a RE and a PFE) and the underlying physical topology is already built in VBox as well. The next step is to log in and start using the Junos CLI.

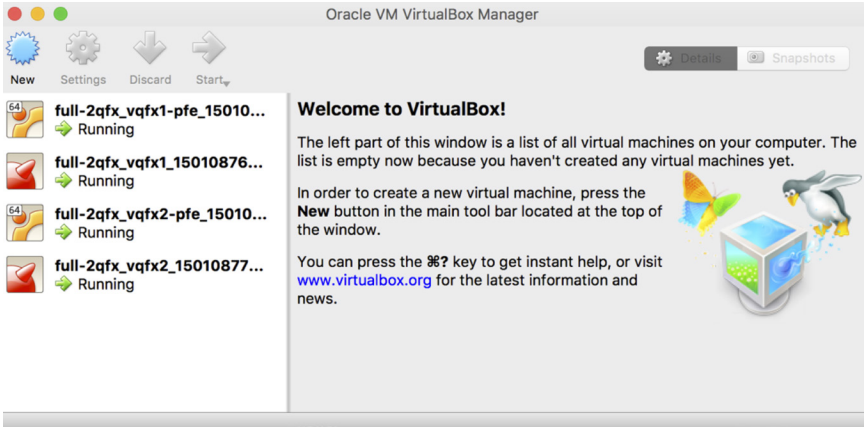


Figure 15.3 *Oracle VirtualBox Hypervisor with Virtual Machines (VM).*

Accessing vQFX

From the screen capture in the last output you can see that the full2-vqfx topology is launched and actively running in VirtualBox. Now let's access the devices from the command line:

```
$ vagrant ssh vqfx1
--- Junos 15.1X53-D63.9 built 2017-04-01 20:45:26 UTC
{master:0}
vagrant@vqfx-re> show chassis hardware
Hardware inventory:
Item             Version  Part number  Serial number     Description
Chassis                                26780205283       QFX3500

vagrant@vqfx-re> show configuration
## Last commit: 2017-04-07 12:42:24 UTC by root
version 15.1X53-D63.9;
system {
    host-name vqfx-re;
    root-authentication {
        encrypted-password "$1$3ttX6Wqv$vYBmNrdOf9f.OQRQxf/SQ1"; ## SECRET-DATA
        ssh-rsa "ssh-rsaAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW6A8YVr+kz
4TjGYe7gHzIw+niNltGEFHzD8+v1I2YJ6oXevct1YeS0o9HZyN1Q9qgCgzUFtdOKLv6IedplqoPkcmF0aYet2P
kEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx4O7PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIhyfHilFR61
RGL+GPXQ2MWZWFYbAGjyiYJnAmCP3NOTd0jMZEnDkbUvxhMmBYSdETk1rRgm+R4LOzFUGaHqHDLKLX+FIPKcF9
6hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure public key"; ## SECRET-DATA
    }
    login {
        user vagrant {
            uid 2000;
            class super-user;
            authentication {
                ssh-rsa "ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6NF8iallvQVp22WDkTkyrtvp9eWW
6A8YVr+kz4TjGYe7gHzIw+niNltGEFHzD8+v1I2YJ6oXevct1YeS0o9HZyN1Q9qgCgzUFtdOKLv6IedplqoPkc
```

```
mF0aYet2PkEDo3MlTBckFXPITAMzF8dJSIFo9D8HfdOV0IAdx4O7PtixWKn5y2hMNG0zQPyUecp4pzC6kivAIh
yfHilFR61RGL+GPXQ2MWZWFYbAGjyiYJnAmCP3NOTd0jMZEnDkbUvxhMmBYSdETk1rRgm+R4LOzFUGaHqHDLKL
X+FIPKcF96hrucXzcWyLbIbEgE98OHlnVYCzRdK8jlqm8tehUc9c9WhQ== vagrant insecure public key
"; ## SECRET-DATA
            }
        }
    }
    services {
        ssh {
            root-login allow;
        }
        netconf {
            ssh;
        }
        rest {
            http {
                port 8080;
            }
            enable-explorer;
        }
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
    extensions {
        providers {
            juniper {
                license-type juniper deployment-scope commercial;
            }
            chef {
…. Truncated for brevity
```

The really cool part is that the physical layer is all prebuilt inside VirtualBox and it's ready to start networking. You can see that there are real interfaces to work with from this output:

```
vagrant@vqfx-re> show interfaces terse
Interface               Admin Link Proto    Local                 Remote
gr-0/0/0                up    up
pfe-0/0/0               up    up
pfe-0/0/0.16383         up    up   inet
pfh-0/0/0               up    up
pfh-0/0/0.16383         up    up   inet
pfh-0/0/0.16384         up    up   inet
xe-0/0/0                up    up
xe-0/0/0.0              up    up   inet
…
```

After logging into each vQFX, and configuring LLDP to the interfaces, you can see the neighbor and the way the two devices are connected:

```
vagrant@vqfx1-re> show lldp neighbors
Local Interface    Parent Interface    Chassis Id          Port info          System Name
xe-0/0/1           -                   02:05:86:71:5e:00   xe-0/0/1           vqfx2-re
xe-0/0/0           -                   02:05:86:71:5e:00   xe-0/0/0           vqfx2-re
xe-0/0/2           -                   02:05:86:71:5e:00   xe-0/0/2           vqfx2-re
xe-0/0/3           -                   02:05:86:71:5e:00   xe-0/0/3           vqfx2-re
xe-0/0/4           -                   02:05:86:71:5e:00   xe-0/0/4           vqfx2-re
```

Summary

You learned how to install the prerequisites for the vQFX10000 vagrant topologies, and how to install and launch the vagrant topology. You've also learned how to discussed how to log into the virtual machines using `$ vagrant ssh vqfx1` and `vqfx2`.

This is a very user friendly way of pulling down and configuring different topologies and learning not only Junos but also a platform where you can use Ansible and PyEZ and even deploy Junos Space to manage these devices.

The entire process takes less than 30 minutes from start to working topology, so it's worth the time to build your own lab or learning environment. By the way, you'll need a laptop with at least 16G RAM to launch a full-4vQFX topology.