

# DAY ONE: FINISHING JUNOS DEPLOYMENTS

Before your next cutover, ask yourself:

*Is there anything I neglected to do?*

Here's a *Day One* book that can help you finalize your deployments with a suite of best practices, sample configurations, and a handy checklist.

By Martin Brown

# DAY ONE: FINISHING JUNOS DEPLOYMENTS

When you are almost done with a deployment and see the straightaway to the finish dead ahead—when you begin racing towards the cutover – that’s exactly when you need to take a deep breath and concentrate on the finishing touches.

*Is there anything important I neglected to do?* Experience has taught you that while it’s probably a good deployment, you still need to consider whether anything is stopping it from becoming a *great* deployment.

*Day One: Finishing Junos Deployments* is a twofold book: first, it serves as a reminder for those all-important finishing touches that should be applied to any Junos OS deployment, and second, it describes how to implement those finishing touches on a network device running Junos OS.

“Martin Brown takes post-Junos deployments to another level in this excellent *Day One* book! We sometimes forget about attention to detail in routine configuration deployments and this book goes to great lengths to ensure that no configuration item, big or small, is overlooked. This book is a time saver, headache reliever, and the pre-deployment checklist is a must!”

*Scott Ware, Network Security Engineer, ITS Security Technologies*

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

- Better understand Network Monitoring
- Use special Junos commands to check that your configuration is working as expected
- Understand how to control access to devices
- Be able to deploy a more secure network device

Juniper Networks Books are singularly focused on network productivity and efficiency. Peruse the complete library at [www.juniper.net/books](http://www.juniper.net/books).

Published by Juniper Networks Books



**JUNIPER**  
NETWORKS

Junos® Fundamentals Series

# Day One: **Finishing Junos® Deployments**

By Martin Brown

<i>Preface .....</i>	<i>vii</i>
<i>Chapter 1: The Necessity of NTP .....</i>	<i>11</i>
<i>Chapter 2: SNMP and the Health of Your Network .....</i>	<i>21</i>
<i>Chapter 3: The Gold Mine That Is Syslog. ....</i>	<i>35</i>
<i>Chapter 4: Authentication, Authorization, and Accounting .....</i>	<i>41</i>
<i>Chapter 5: Securing Device Management. ....</i>	<i>51</i>

© 2015 by Juniper Networks, Inc. All rights reserved. Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

**Published by Juniper Networks Books**

Author: Martin Brown  
Technical Reviewers: Riley Fitzpatrick, Carl P. Francis,  
Pete Robbins, Scott Ware  
Editor in Chief: Patrick Ames  
Copyeditor and Proofer: Nancy Koerbel  
Illustrator: Karen Joice  
J-Net Community Manager: Julie Wider

ISBN: 978-1-936779-14-5 (print)  
Printed in the USA by Vervante Corporation.  
ISBN: 978-1-936779-15-2 (ebook)

Version History: v1, May 2015  
2 3 4 5 6 7 8 9 10

**About the Author:**

Martin Brown is a Network Security Engineer for a major telco based in the UK, and a Juniper Ambassador with knowledge that covers a broad range of network devices. Martin started his career in IT 20 years ago supporting Macintosh computers, became an MCSE in 1999, and has since progressed to networking, supporting most of the major manufacturers including Cisco, F5, Checkpoint, and of course, Juniper.

**Author's Acknowledgments:**

I would like to thank my good friend, Joy Horton, for making me take time out, relax and enjoy my down time once I'm out of the office, and Lee Wild for inspiring me to continue learning and for keeping me on my toes while I'm in the office. I would also like to give a special thank you to Scott Ware for agreeing to be my technical editor and checking that what I've said is accurate, and of course to Julie Wider for administering the Juniper Ambassadors and for helping us to become more engaged in activities outside of our normal day jobs.

This book is available in a variety of formats at:  
<http://www.juniper.net/dayone>.

## Welcome to Day One

This book is part of a growing library of *Day One* books, 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 larger and longer suite of *This Week* books, whose concepts and test bed examples are more similar to a weeklong seminar.

You can obtain 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 Store. Search for Juniper Networks Books.
- 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.
- Purchase the paper edition at either Vervante Corporation ([www.vervante.com](http://www.vervante.com)) for between \$12-\$28, depending on page length.
- Note that Nook, iPad, and various Android apps can also view PDF files.

## Audience

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

The chapters of this book are numbered in a logical sequence designed to establish a natural progressive flow when finalizing a deployment of a network device.

## What You Need to Know Before Reading This Book

Before reading this book, you should be familiar with the basic administrative functions of the Junos operating system, including the ability to work with operational commands and to read, understand, and change Junos configurations. There are several books in the *Day One* library on learning Junos, at [www.juniper.net/dayone](http://www.juniper.net/dayone).

This book makes a few assumptions about you, the reader:

- You have a basic understanding of the Internet Protocol (IP) Version 4.
- You have access to a lab with at least the following components: one SRX-Series firewall, one EX-Series switch (with port mirroring capability), one server, and one workstation.

## By Reading This Book You Will

- Better understand Network Monitoring
- Know and use special Junos commands to check your configuration is working as expected
- Understand how to control access to devices
- Be able to deploy a more secure network device

## Information Experience

This *Day One* book is singularly focused on one aspect of networking technology that you might be able to do in one day. There are other sources at Juniper Networks, from white papers to webinars to online forums such as J-Net ([forums.juniper.net](http://forums.juniper.net)). Look for the following sidebars for access directly to other superb resources for information.

**MORE?** It's highly recommended you go through the technical documentation in order to become fully acquainted with the initial configuration process of Junos devices in order to get a better understanding of the configuration process flow, before you jump in. The technical documentation can be located at [www.juniper.net/documentation](http://www.juniper.net/documentation). Use the Pathfinder tool on the documentation site to explore and find the right information for your needs.

## Preface

As network engineers, our primary focus always tends to be on getting the traffic flowing across a device, or in the case of a firewall, filtering traffic from entering the device. In either case, when we deploy a new network device we like to start assigning IP addresses to interfaces as soon as we can; we want to enable a routing protocol, create VRFs, assign RDs, and set the device as a PE router before finally testing reachability across the LAN. Then, once we know we are able to reach the furthest corners of our WAN, we feel an amazing sense of satisfaction that we have created an integral part of a network that will hopefully remain in place for years to come. In other words, we are almost done with the deployment and we see a straightaway dead ahead and begin racing towards the cutover.

However, many a network engineer will attest that now is exactly the time you need to stop and think: Is there anything important I have neglected to do? Experience has taught you that while this is probably a good deployment, you should consider whether anything is stopping it from becoming a great deployment.

The purpose of this book is twofold: first, it serves as a reminder for those all-important finishing touches that should be applied to any Junos OS deployment, and second, it describes how to implement those finishing touches on a network device running Junos OS.

At the end of this preface there is a checklist that breaks the tasks up into five general categories: NTP, SNMP, system log (syslog), AAA, and Secure Device Management. The remainder of this book then provides a high-level view of the technologies in the checklist, dividing each into its own chapter describing the technology and any pros and cons of particular versions and then detailing how to implement the technology on a device running Junos OS and how to test its functionality.

Figure P.1 shows the logical topology used for the purposes of writing this book, a small LAN consisting of a Juniper EX Switch, connected to a Juniper SRX firewall in addition to an SNMP server, a system log server, and an ACS server. Somewhere in the cloud there is an NTP server.

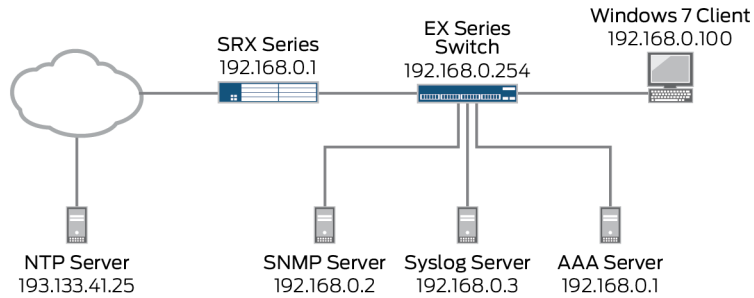


Figure P.1 This Book's Topology

**NOTE** The version of Junos OS software running on the Juniper Network device should not really matter too much as the commands are mostly version neutral, applicable to any version of Junos. That said, for those interested, the switch configuration examples and output in this book were taken from an EX2200 switch running Junos 14.1X53-D15. The SRX, on the other hand, is an SRX100 running Junos 12.1X44-D40. In the case where a command is only available in a more recent release of Junos, this will be noted.

With reference to the servers, in theory, any system log or SNMP monitoring software could be used and there are some very good examples of software available on the market, such as those made by HP or Solarwinds. However for the purposes of testing SNMP, PRTG from Paessler was chosen ([http://www.paessler.com/info/snmp\\_server](http://www.paessler.com/info/snmp_server)). Whilst testing system log, I decided to use the Kiwi Syslog Server from Solarwinds (<http://www.solarwinds.com/products/freetools/free-kiwi-syslog-server.aspx>). Both of these monitoring tools are available for free download from the developer, but because they are free, they obviously have limitations on the number of devices that can be monitored at once. These limitations are removed in the licensed version of the software but for the purposes of a lab environment, or indeed to practice any of the topics in this book, the free versions are satisfactory.

The ACS server, used to test AAA, is a Windows 2008 R2 server with Network Policy Server installed, thus providing the RADIUS functionality. Information on this can be found at [http://technet.microsoft.com/en-us/library/dd365355\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/dd365355(v=ws.10).aspx).



**NOTE** You may notice that all the management servers are utilizing Windows. Some network engineers, however, prefer Linux over Windows and this is a viable option, too. Should you wish to use a Linux server to perform testing with any of the technologies discussed in this book, my fellow Juniper Ambassador Darren O'Connor has written an excellent blog article detailing how to use a Linux server for SNMP, NTP, system log, and RADIUS. The blog can be found here: <https://mellowd.co.uk/ccie/?p=4150>.

Now that you know more about the topology, let's have a quick glance at the checklist.

## The Pre-Deployment Checklist

The next page contains a pre-deployment checklist so that you can see, at a glance, what configuration tasks you have already performed and what tasks remain to be completed. The rest of this book is organized according to this checklist, with Chapter 1 covering NTP, Chapter 2 covering SNMP, and so forth.

**MORE?** The checklist is also available as a single-page PDF on this book's last page as well as on this book's landing page at <http://www.juniper.net/dayone>.

When you finish your next deployment, try using this checklist to help remind you about the finishing touches. Then you can go get a beer and celebrate.

Martin Brown

## Day One: Deploying Junos Finishing Touches

### Pre-Deployment Checklist



NTP			
Device IP added to Syslog Server	<input type="checkbox"/> None	<input type="checkbox"/> MDS	
Redundant NTP	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Delivery	<input type="checkbox"/> Unicast	<input type="checkbox"/> Multicast	<input type="checkbox"/> Broadcast
Junos NTP Server	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
[show ntp associations]	<input type="checkbox"/> Output as expected		
SNMP			
Version	<input type="checkbox"/> 1	<input type="checkbox"/> 2c	<input type="checkbox"/> 3
Community	<input type="checkbox"/> Read Only	<input type="checkbox"/> Read-Write	<input type="checkbox"/> N/A
Authentication	<input type="checkbox"/> None/NA	<input type="checkbox"/> MDS	<input type="checkbox"/> SHA
Privacy	<input type="checkbox"/> None/NA	<input type="checkbox"/> AES	<input type="checkbox"/> 3DES
[show ntp statistics]	<input type="checkbox"/> Output as expected		
Syslog			
Device IP added to Syslog Server	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Source address set in Junos	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Logging level	<input type="checkbox"/> Any	<input type="checkbox"/> "Tuned"	
Facilities	<input type="checkbox"/> Any	<input type="checkbox"/> "Tuned"	
Year/Millisecond specified	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Syslog server receiving logs with correct level and facility	<input type="checkbox"/>		
AAA			
Root password set	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Second superuser created	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Classes linked to VendorID 1	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
AAA Server	<input type="checkbox"/> RADIUS	<input type="checkbox"/> TACACS+	
AAA Accounting enabled	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Test logins work with correct permissions	<input type="checkbox"/>		
Transactions logged with AAA server	<input type="checkbox"/>		
Securing Device Management			
Telnet	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
http	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
SSH Version	<input type="checkbox"/> 1	<input type="checkbox"/> 2	
Firewall Filter	<input type="checkbox"/> Local Subnet	<input type="checkbox"/> http	<input type="checkbox"/> telnet
[Show firewall log]	<input type="checkbox"/> Output as expected		
[show firewall filter <filter name>]	<input type="checkbox"/> Output as expected		

©2015 by Juniper Networks, Inc. All rights reserved. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

For more information, see [www.juniper.net/dayone](http://www.juniper.net/dayone)

Figure P.2 The Pre-Deployment Checklist for Deploying Junos Finishing Touches

**NOTE** A full-sized checklist exists as the very last page of this book.

# Chapter 1

## The Necessity of NTP

Let's suppose you are an implementation engineer whose role is to make changes to network devices, and your manager has given you a change to write and implement. You implement the change during the appropriate change window, which happens to be late in the evening. The next day the support team starts receiving calls about end users who are experiencing issues with the network. The first thing the support team does is check the logs, which show that you made the manager's change and then saved the configuration at 7AM a week ago. The support team assumes the changes you made are unrelated to the network issues and they move on. If the support team is good they'll also check all the approved changes for the previous evening and they will find out that you did make a change, correcting whatever was wrong, but there is an unacceptable delay in problem resolution because the date and time are inaccurate.

Other issues with having incorrect dates and times affect security. As an example, your organization may have an SSL VPN that uses identity certificates for authentication. If the date and time is set in the future, a certificate may expire prematurely and if the date is too far behind, it may never become valid. This is not as farfetched as it may sound, as some network devices do not have the means of storing the time during a power outage, therefore when they reboot, the device will believe it's 1993.

These scenarios and plenty more are the exact reasons why the date and time on network devices needs to be as accurate as possible and why almost all network devices should have the facility to connect to an NTP (Network Time Protocol) server to obtain accurate time information. However, it's your job to add that finishing touch.

## How an NTP Server Finds the Time

The purpose of an NTP server is to act as an accurate and reliable time source, but NTP servers themselves obviously don't work by magic, and they need a way to set their own time before they can tell any other device what time it is.

It is perfectly possible to tell an NTP server to use another NTP server as a time source, and that server in turn can get its time from yet another server. But that gets a little complicated. What you need is an accurate clock attached directly to what is effectively a *root server*, one that can then tell other NTP servers, with a high degree of accuracy, exactly what time it is.

In order to fulfil this need, several LF or Low Frequency radio clocks were built around the globe, each using a different frequency to transmit a time signal. These clocks themselves are attached to what is known as an *atomic clock* that provides accurate time measurement by using the decay of caesium. The only issue with ground-based clocks is that radio waves can be reflected off of buildings as well as the surrounding landscape; this coupled with the receiver needing to decode the information can reduce the accuracy of the time server set by a radio clock to around 1(ms) millisecond. Although 1ms accuracy is not a huge amount that would affect reporting or security within a network, redundancy is and if the radio clock is not transmitting due to maintenance, then the root time source is lost. Happily, another time source has become available, almost as a by-product of the invention of GPS.

GPS satellites need to know exactly what time it is, otherwise their position would be incorrectly reported leading to navigation issues on the ground. GPS satellites send this time signal out to ground based receivers so that they can calculate their position. Some NTP servers can receive this signal, not for navigation purposes but to set the time themselves, so that they can in turn tell it to other devices. As the distance between earth and the satellites is typically 12,000 miles, and it takes time for the signal to reach earth coupled with the need to process the data and forward it on, NTP servers that take their data from GPS satellites may have an accuracy of around 240 microseconds.

Using GPS navigation as an accurate time source usually requires four satellites to be in view. NTP can use the signal from a single satellite as a time source; however the availability of more than one satellite covers the need for redundancy.

Figure 1.1 illustrates how an NTP server can use both a radio clock and a GPS satellite to set its time, which can then be used as a time source for other devices on the network.

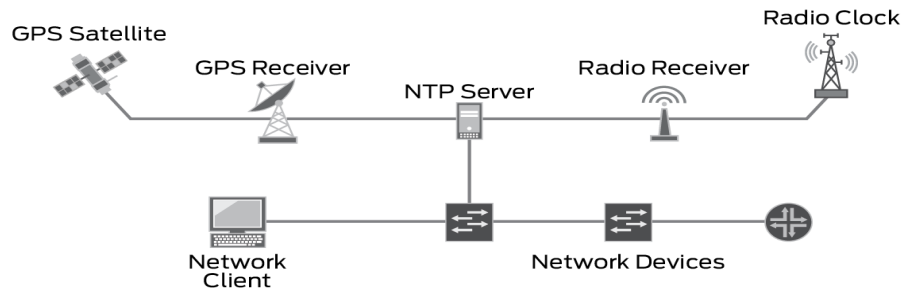


Figure 1.1 A Typical NTP Scenario

**NOTE** Although atomic clocks are extremely accurate, they are unable to take into account that the earth's orbit around the sun varies, therefore atomic clocks in GPS satellites and those connected to radio clocks are changed when needed to take into account the variation in earth's orbit for that year. This time is synchronized with a 150 year-old clock found at the top of the Elizabeth Tower, more commonly known as Big Ben, located in London.

## NTP Reliability

The reliability of an NTP server is measured in stratum, numbered 0 through 16, with 0 being considered reliable and accurate whereas 16 probably shouldn't be trusted. By using human beings as an analogy for a moment, it's easy to see why this is the case.

Let's say your friend owns a wristwatch and this morning he set the time while listening to the radio. Your friend looks at his watch and he can tell that the time is 11AM. He then tells you what the time is. Another friend of yours then asks you the time at which point you duly inform him it's 11AM, at which point he tells a colleague it's 11AM. One minute has elapsed since your friend looked at his watch and it is now 11:01AM, however, your friend's colleague has been told it's still 11AM, therefore he has been told an incorrect time.

If you translate this scenario into stratum, you could say that the time announced on the radio was stratum 0, the wristwatch set by the radio time is stratum 1, your friend reading the time from the watch is stratum 2, you are stratum 3, the friend you told the time to is stratum 4, and your friend's colleague is a stratum 5. If this chain were to continue, at some point the time would be inaccurate enough to make a difference, and for this reason, stratum 16 is the last layer, at which point the device can no longer be used as a valid time source.

Figure 1.2 provides a graphical representation of the how the flow would work between devices. At stratum 0 is the GPS satellite and a LF radio time source. These are directly connected to the atomic clock itself, therefore these are considered extremely accurate. The first NTP server takes its time directly from the time source, this server is at stratum 1. Not all servers are able to do this, however, so the next layer takes the time from the stratum 1 server. This repeats until we get to stratum 16, at which point the server is marked as unreliable. Putting this into perspective, a standalone server with no NTP time source applied is thought to be in the same category as a stratum 16 device.

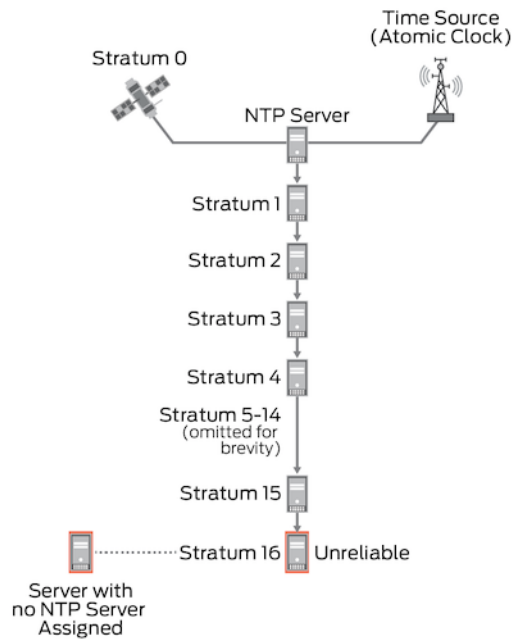


Figure 1.2 NTP Stratum Layers

As mentioned in our scenario, one minute is hardly going to make a difference in human terms, but in network terms, when correlating information as is done in the financial markets, a second, or even a few hundredths of a second, can make a difference.

#### BEST PRACTICE

The time applied to all network devices must be as accurate as possible.

## Configuring Devices Running Junos OS as NTP Clients

All Junos devices are able to set their time via NTP. The only question is: *What time zone should the device use?* In theory, the time zone should be set to the time zone in which the device resides, but what happens during Daylight Saving Time (DST)? The logs would show that 1 AM occurred twice on the same day in the Fall, or would lose an hour in the Spring. Consideration needs to be given to large multinational companies with support teams based around the globe dealing with incidents in other countries. As a result of this situation, many organizations use the Coordinated Universal Time (UTC) as the standard for their devices. UTC isn't a time zone but a standard, and it is the basis for consistent and reliable 24-hour world time, which remains unchanged when DST is observed. So on your Juniper Network device running Junos OS, configure:

```
set system time-zone UTC
```

Now that the time zone is set, the device needs to be told how to reach the server using the command `set system ntp server` and the IP address. After the IP address, the keyword `prefer` can be included; the purpose of which is to tell Junos OS which server to use first in the event multiple servers are listed. The command `set system ntp boot-server` tells Junos OS to use that particular NTP server when the device has been rebooted or has lost power. As soon as network connectivity is restored, the device will immediately synchronize even if the NTP process is disabled for some reason. In addition, if the time difference between the client and the NTP server is greater than 1000 seconds, the NTP server will try to synchronize in steps, but if the `boot-server` is set, Junos OS will synchronize immediately on booting regardless of how far out the device time is from the actual time:

```
set system ntp server 193.133.41.23 prefer
set system ntp boot-server 193.133.41.23
```

After committing the changes, check your Juniper Networks device to ensure it is communicating with the server by using the command `show ntp associations`. In the following instance, the reference ID or `refid` is showing as `.INIT.`, meaning Junos OS is awaiting the initial sync:

```
[edit]
{master:0}[edit]
root@# run show ntp associations
```

remote	refid	st	t	when	poll	reach	delay	offset	jitter
193.133.41.23	.INIT.	16	-	-	64	0	0.000	0.000	4000.00

Unfortunately, the initial sync never came. The configuration was checked and it was discovered that the wrong IP address of the NTP

server was configured. Those familiar with Junos OS know that this can be easily corrected by using the `rename` command, shown in the instance below.

**NOTE** You may notice the command prompt is currently set as `root@#`. This means that the administrator is currently logged in as the user `root`, which is created by default and gives whomever is logged in as this user full access to not only Junos OS, but the underlying operating system, too. Because it is considered bad practice to continue to use this user account, how to create new and create more restrictive user accounts is discussed a little later on in Chapter 3.

```
{master:0}[edit]
root@# rename system ntp server 193.133.41.23 to server 193.133.41.25
root@# rename system ntp boot-server 193.133.41.23 to boot-server 193.133.41.25
```

Once renamed and committed, NTP associations should be checked again:

```
{master:0}[edit]
root@ # commit
configuration check succeeds
commit complete
```

```
{master:0}[edit]
root@# run show ntp associations
```

remote	refid	st	t	when	poll	reach	delay	offset	jitter
193.133.41.25	.STEP.	16	-	107	64	0	0.000	0.000	4000.00

In this instance, the `refid` is showing as `.STEP.` meaning that the time received from the server differs greatly from the time currently set in Junos OS, and instead of just setting the time immediately, Junos will attempt to set the time in steps. This could take too long if the time is too far out, therefore those connected to the console may see the following error appear:

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

Time and ticks drifted too much,                      resetting synchronization...

This means the times were too far apart. Therefore, Junos OS issues a reset on the time synchronization and starts using the time from the NTP server. Checking the NTP association will confirm that the device has synchronized with the NTP server as the column “`st`” referring to the stratum is set as 1, meaning the server is a stratum 1 server and the `refid` column is set as `GPS` meaning that the NTP server set its time directly from a GPS satellite:



```
{master:0}[edit]
root@# run show ntp associations
      remote          refid      st t when poll reach  delay  offset  jitter
=====
193.133.41.25    .GPS.          1 -   1   64    1   17.933  2280208  0.000
```

## NTP Security

Earlier in this chapter it was mentioned that having an incorrect time can cause security issues, therefore it is understandable that some engineers may assume that once NTP is set correctly it will resolve said security issues. Unfortunately, this is not the case.

Time information from an NTP is sent as unencrypted plain text with no authentication. Although this information is normally requested, as opposed to being sent gratuitously, it is possible that an attacker with a bit of knowledge could send a spoofed NTP packet that the client would receive and as such believe that the time showed by the spoofed packet was now the correct time.

To counter this potential problem, NTP does allow for authentication. The NTP servers and clients are issued a trusted key in the form of a number and a password is assigned to this key. The key and password are hashed using MD5, which is then used for authentication.

Under normal circumstances, when a client connects to a server it is the server that asks the client for the login details, but NTP is different. With NTP, it is the client that asks the server for the login details. The reason for this is simple. If a network device that doesn't support authentication exists on the network, then the client can just perform time synchronisation unauthenticated – for the remaining devices on that network that do support it, those clients can use authentication so that they know that the time they have received can be trusted.

## Adding NTP Security to Devices Running Junos OS

Adding the configuration needed to enable NTP authentication consists of telling Junos OS which trusted key to use, which in this case will be 1234, then configuring in Junos OS (1) which NTP servers would use this key, (2) which MD5 hashing should be used, and finally, (3) the password `secretpassword` to be associated with the key:

```
set system ntp trusted-key 1234
set system ntp server 193.133.41.25 key 1234
set system ntp authentication-key 1234 type md5
set system ntp authentication-key 1234 value secretpassword
```

In previous configuration examples, the `show ntp associations` command was used. This command can tell an administrator the state of the server. An alternative command would be to use `show ntp status` that shows the NTP status of the device itself:

```
{master:0}[edit]
root@# run show ntp status
status=0664 leap_none, sync_ntp, 6 events, event_peer/strat_chg,
version=ntpd 4.2.0-a Sun Dec 21 01:10:34 UTC 2014 (1),
processor=arm, system=JUNOS14.1X53-D15.2, leap=00, stratum=2,
precision=-16, rootdelay=2.978, rootdispersion=6.000, peer=47652,
refid=193.133.41.25,
reftime=d8660907.9ce93852 Sun, Jan 18 2015 10:27:19.612, poll=6,
clock=d8660941.6139be63 Sun, Jan 18 2015 10:28:17.379, state=3,
offset=0.000, frequency=0.000, jitter=2.372, stability=0.000
```

A brief look at the output tells you the following:

- The version of Junos OS running on the device.
- The stratum of the device, which in this case it is 2 as the time source is a stratum 1.
- How many leap seconds are in use.
- The `rootdelay`, which details how many seconds would elapse during a round trip to the primary NTP server.
- `Rootdispersion`, which in seconds tells you what the maximum error in relation to the primary reference source.
- `Precision`, which refers to how precise the peer clock is, how precisely the frequency and time can be maintained with that peer, and how many seconds elapse between polls of the NTP server.

As is evident, the output confirms that the device has received valid time updates from the NTP server, meaning NTP authentication is working as expected. If synchronization or authentication with the NTP server had failed, then the stratum would be shown as 16.

## Reducing NTP Traffic Across Low Bandwidth Links

Chances are your network has at least one, or possibly many, remote sites. These remote sites need a time source of some kind, and although it's possible to still have each NTP client connect to an NTP server located in the head office, in some cases, such as the scenario represented in Figure 1.3, some remote sites are connected to the head office via a low bandwidth VPN link, where unnecessary traffic traversing the link needs to be kept to a minimum.

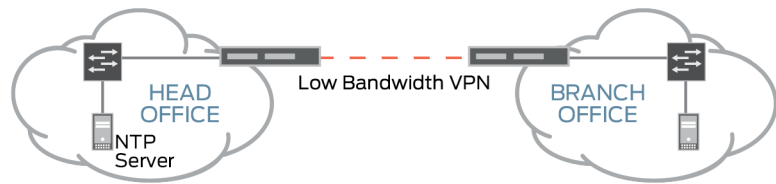


Figure 1.3 Branch Connected to Head Office Via Low Bandwidth VPN Link

One solution would be to have a local NTP server at each site, however this could prove expensive if there are a lot of remote sites, or wasteful if the site list is small. An alternative solution is to use an SRX Series firewall as an NTP server. The SRX itself would take the time from the NTP server located in the head office and the NTP clients in the branch office would use the SRX as an NTP server. Let's try it.

## Configuring Devices Running Junos OS as NTP Servers

All that is necessary to make a device running Junos OS an NTP server is to enable it as an NTP client. In this case the SRX will be used as the server. The same commands as before will be used and security will be enabled:

```
set system ntp trusted-key 1234
set system ntp server 193.133.41.25 key 1234
set system ntp server 193.133.41.25 prefer
set system ntp authentication-key 1234 type md5
set system ntp authentication-key 1234 value secretpassword
```

The EX switch will act as the NTP client. This client will use authentication and the IP address of the server will be the default gateway of VLAN 192, which resides on the SRX:

```
set system ntp trusted-key 1234
set system ntp server 192.168.0.1 key 1234
set system ntp server 192.168.0.1 prefer
set system ntp authentication-key 1234 type md5
set system ntp authentication-key 1234 value secretpassword
```

**WARNING** As mentioned previously, without NTP synchronization, Junos OS will announce itself as being stratum 16 and as discussed, stratum 16 is considered untrusted, therefore any device receiving an update from a stratum 16 time source would immediately discard it. Thus it is not possible to make a device running Junos OS an NTP server without first synchronizing its own clock with an external NTP server.

## Delivering Time Updates Via Multicast

Although many organizations configure devices by specifying IP addresses of individual servers, as a network engineer you have the option to elect to receive updates via multicast. It is also possible to specify that a device running Junos OS acting as an NTP server can advertise NTP updates via multicast or even via broadcast. The benefit of sending NTP updates this way is that if you need to replace an NTP server, or have an NTP server addressed with dynamic addressing, you won't need to go to every device and set the new server address, you just need to ensure the new server is broadcasting or sending multicast packets to the same address as the previous server.

The next command advertises updates via the multicast address 224.0.1.1, which is the address reserved for NTP multicast:

```
set system ntp broadcast 224.0.1.1
```

The following command allows a Juniper Networks device to listen to multicast group 224.0.1.1 for NTP updates:

```
set system ntp multicast-client 224.0.1.1
```

**NOTE** Although the address 224.0.1.1 is officially reserved for NTP multicast, 224.0.0.0/8 addresses typically have a Time To Live (TTL) of 1, meaning the multicast packets won't be sent beyond that local subnet. It is possible to increase this TTL, however as 224.0.0.0/8 addresses were meant to remain in the local subnet; it's better to use a private 239.0.0.0/8 address and then specify a TTL at the end of the command, so that the updates are sent outside your local subnet but as the address range is private, the updates will not be sent outside your network onto the Internet.

## Summary

Time, and the correct time, is absolutely critical to the day-to-day functioning of your network. Your logs will have more meaning, security is stronger, and the communication between your many devices will be accurate.

NTP is a finishing touch in Junos OS. Make sure you have a system of getting the most accurate time you can and a method for sharing it within your network that requires little communication or effort.

# Chapter 2

## SNMP and the Health of Your Network

The Simple Network Management Protocol, or SNMP, allows a network administrator to automatically collect data about the health and availability of a network device under their control. The type of data that it is possible to collect can include real-time information on the state of monitored interfaces, the amount of bandwidth in use, the temperature of the device, system uptime, and so on. In addition to collecting information, it's also possible to configure the device to send a message to the server in the event of interface failure, or when unauthorized personnel attempt to connect to the device. The SNMP server can then send an e-mail or text to your smartphone notifying you that something has occurred – all within seconds of occurrence.

Figure 2.1 illustrates this chapter's SNMP topology of a small LAN consisting of a switch, a router, an SNMP server, and a monitoring client. The network devices are configured to allow the SNMP server to connect to them and in turn the administrator tells the SNMP server which data to collect. The administrator then logs in to a workstation that has the SNMP client software installed, monitors the server, and views the collected information. (Some SNMP servers do not require any software installation and the information can be viewed by connecting to a website stored on the SNMP server.)

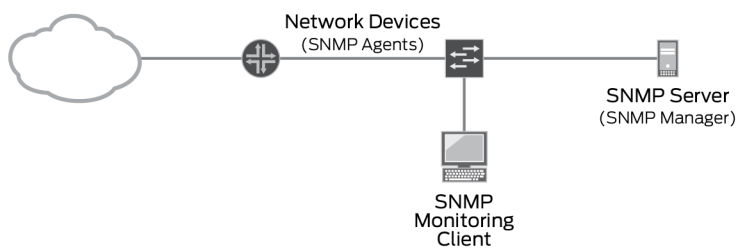


Figure 2.1 SNMP Example Topology

**NOTE** In this example, it is important to notice that in the event the switch fails, the SNMP server loses all connectivity and is never able to notify the administrator of the failed device. For this reason, most organizations use a cluster of SNMP servers connected to redundant switches.

## SNMP Components

SNMP can be broken down into three components: an SNMP manager, which is the SNMP server itself; the SNMP agent, which runs directly on the device to collect and report the relevant data to the manager; and the management information base or MIB, a database stored on the agent that details the objects within the device and how they are configured (for example, at what interface speed and duplex settings). The objects within the MIB are identified by an Object ID more commonly known as an OID.

There are several operations that can be used to obtain information using the SNMP manager, agent, and MIB:

- The first operation is *Get*, where the manager requests information from the MIB stored on the agent.
- The second is *Set*, where an administrator can change how the device is configured by using the Set operation, which can change information in the MIB on the agent.
- The third operation is *Walk*, which can be used to request information from successive objects in the MIB.
- In the fourth operation, *Trap*, the agent will send unsolicited data to the SNMP manager whenever an event has occurred. This event could be anything happening to the device, such as an interface going down or up, losing an OSPF neighborship, or when the device comes back online after losing power.

- The last operation, *Inform*, is the same as a trap, except that an Inform is acknowledged as being received by the manager, whereas a Trap is not.

When configuring the SNMP agent on a device, the administrator has the option to create what is known as a *trap-group* that tells the agent which traps need to be sent to the manager. This group contains a single object or all available objects.

**MORE?** Should you wish to read and understand more about SNMP architecture and SNMP within Junos OS, then the following documentation in the Juniper TechLibrary is worth checking out: [http://www.juniper.net/techpubs/en\\_US/junos14.1/topics/concept/understanding-snmp-junos-nm.html](http://www.juniper.net/techpubs/en_US/junos14.1/topics/concept/understanding-snmp-junos-nm.html).

**NOTE** While it's possible to configure a network device via SNMP, most organizations traditionally use SNMP only as a data collector. This common behavior is explained in the next section.

## Security Not My Problem

One of the issues that occurred with various versions of SNMP was trying to balance security with the complexity of configuration.

Version 1 of SNMP was easy to configure, using a simple password known as *community* for authentication purposes. In addition, it was designed for older devices that couldn't really spare much processing power, therefore the packets were relatively small and they were sent unencrypted as plain text. It was therefore possible for an intruder to run packet capture software on the LAN and capture the password as it was sent across the network. If the community allowed the administrator to modify the device, then the captured password would allow the intruder to control the device. In addition to the security issues, SNMP Version 1 only had 32-bit counters and as bandwidth increased, it was found that these counters could not store bandwidth information. For example, if a 1Gb/s interface was fully utilized, then the 32-bit counter would reach its maximum and reset itself to 0 in 34 seconds.

To resolve the issues with SNMP Version 1, SNMP Version 2 was released, which offered a much better security model as well as better performance and introduced a feature known as *GetBulkRequest*, which allowed the SNMP manager to get data from several objects on the agent at once as opposed to issuing multiple Get requests.

The drawback with SNMP Version 2 was that the security model was

too complex to implement, so a revision SNMP Version 2c was released. It was essentially Version 2 but without the complicated security model, which unfortunately meant authentication was still via a community and all information was sent as plain text. On the upside, Version 2c did introduce 64 bit counters.

Then came SNMP Version 3. It can be configured with an encrypted password, hashed using MD5 or SHA, or can use a plain text password. In addition, the data can also be encrypted using DES, 3DES, or AES using the levels known as *noAuthnoPriv*, *AuthnoPriv*, and *AuthPriv*. NoAuthnoPriv means the password is sent as plain text, as is the data. AuthnoPriv means the password is hashed using SHA or MD5, but the data is sent using plain text. And AuthPriv means the password is hashed and the data is encrypted with DES, 3DES, or 128-bit AES.

It is slightly more complex to configure SNMP Version 3 than the previous versions and getting it working the first time may take some effort and testing. In order to give enhanced security, SNMP Version 3 utilizes two security models: the User-based Security Model (USM), which covers authentication and encryption, and the VACM (View-based Access Control Model), which determines which MIB objects the user has access to and what that user is able to do (for example, change the MIB objects or just read them).

**MORE?** Juniper provides an excellent document detailing these models at: [http://www.juniper.net/techpubs/en\\_US/junos14.1/topics/concept/snmp-v3-overview-junos-nm.html](http://www.juniper.net/techpubs/en_US/junos14.1/topics/concept/snmp-v3-overview-junos-nm.html).

Although SNMP Version 3 is more complex to configure compared with Versions 1 and 2c, it is not impossible to configure and many organizations have successfully deployed this version on their live network. Table 3.1 summarizes the pros and cons of using each version.

Table 2.1 SNMP Pros and Cons

SNMP Version	Pros	Cons
1	<ul style="list-style-type: none"> <li>• Easy to configure</li> <li>• Widely supported by older SNMP servers</li> </ul>	<ul style="list-style-type: none"> <li>• Data and community sent unencrypted</li> <li>• 32 bit counters</li> </ul>
2	<ul style="list-style-type: none"> <li>• GetBulkRequests introduced</li> <li>• Enhanced performance</li> </ul>	<ul style="list-style-type: none"> <li>• Security model too complex</li> <li>• Not widely implemented</li> </ul>
2c	<ul style="list-style-type: none"> <li>• Easy to configure</li> <li>• 64 bit counters introduced</li> </ul>	<ul style="list-style-type: none"> <li>• Data and community sent unencrypted</li> </ul>
3	<ul style="list-style-type: none"> <li>• Encrypted authentication and data</li> <li>• Granular security controls</li> </ul>	<ul style="list-style-type: none"> <li>• Slightly more complex to configure</li> </ul>



## Configuring SNMP Version 2c on Devices Running Junos OS

Configuring SNMP on devices running Junos OS is done in two parts: configuring the agent to send traps to the manager, and then configuring the agent to allow the manager to obtain information from the MIB. The first task, adding the configuration required to send SNMP traps, is reasonably easy. The first thing is to create a trap group, which in this scenario will be called *SNMP*. Create the trap group when you specify the version to be used, and then the trap destination should be set to the IP address of the SNMP server:

```
set snmp trap-group SNMP version v2
set snmp trap-group SNMP targets 192.168.0.2
```

Next, configure Junos OS with the IP address to use as the source of the SNMP packets. You can use a loopback interface as the source, with the advantage of providing an alternative path to the SNMP server, but in this book's test case, let's specify the IP address of the Layer 3 interface more commonly known as a *Switch Virtual Interface* or SVI of VLAN 192:

```
set snmp trap-options source-address 192.168.0.1
```

Finally, you need to tell Junos OS which events to send to the SNMP server. Here are a few of the options, some of which have sub-tiers and some of which aren't of use in our environment:

```
[edit snmp]
root@# set trap-group SNMP categories ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
  authentication        Authentication failures
  chassis               Chassis or environment notifications
  chassis-cluster       Clustering notifications
  configuration          Configuration notifications
  link                  Link up-down transitions
> otn-alarms            OTN alarm trap subcategories
  remote-operations     Remote operations
  rmon-alarm            RMON rising and falling alarms
  routing               Routing protocol notifications
  services              Services notifications
> sonet-alarms          SONET alarm trap subcategories
  startup               System warm and cold starts
  vrrp-events           VRRP notifications
```

Let's assume that our scenario wants the server to receive events related to authentication failure, chassis information (such as power supply health, interface information), operations performed remotely, routing failures, when the device is reloaded, any configuration changes made, and any service failures. The following commands do just that:

```

set snmp trap-group SNMP categories authentication
set snmp trap-group SNMP categories chassis
set snmp trap-group SNMP categories link
set snmp trap-group SNMP categories remote-operations
set snmp trap-group SNMP categories routing
set snmp trap-group SNMP categories startup
set snmp trap-group SNMP categories configuration
set snmp trap-group SNMP categories services

```

Now that the traps have been set, the configuration for the SNMP agent can be added. The first command in the following list sets the device to accept SNMP requests only from that interface, or the SVI of VLAN 192. The next command sets the community name as *JuniperRox* with the permissions being set as *read-only* as opposed to *read-write* and the final command specifies that Junos OS will only accept requests from a particular subnet or IP address, and in this case, access is restricted to the IP address of the SNMP:

```

set snmp interface vlan.192
set snmp community JuniperRox authorization read-only
set snmp community JuniperRox clients 192.168.0.2/32

```

Let's commit the configuration and then move on to verifying our SNMP setup.

## Verifying SNMP Configuration

By running the `show snmp statistics` command you can tell whether the SNMP server is communicating with the device. The SNMP server uses an SNMP Get Request to request data, and in the following output you can see there were 1550 Get requests and that there were 1885 Get responses indicating successful communication:

```

[edit snmp]
root@# run show snmp statistics
SNMP statistics:
  Input:
    Packets: 1885, Bad versions: 0, Bad community names: 0,
    Bad community uses: 0, ASN parse errors: 0,
    Too big: 0, No such names: 0, Bad values: 0,
    Read onlys: 0, General errors: 0,
    Total request varbinds: 6479, Total set varbinds: 0,
    Get requests: 1550, Get nexts: 335, Set requests: 0,
    Get responses: 0, Traps: 0,
    Silent drops: 0, Proxy drops: 0, Commit pending drops: 0,
    Throttle drops: 0, Duplicate request drops: 0
  V3 Input:
    Unknown security models: 0, Invalid messages: 0
    Unknown pdu handlers: 0, Unavailable contexts: 0
    Unknown contexts: 0, Unsupported security levels: 0
    Not in time windows: 0, Unknown user names: 0

```

Unknown engine ids: 0, Wrong digests: 0, Decryption errors: 0  
Output:  
Packets: 1897, Too bigs: 0, No such names: 0,  
Bad values: 0, General errors: 0,  
Get requests: 0, Get nexts: 0, Set requests: 0,  
Get responses: 1885, Traps: 12

A PRTG traffic monitor can act as an SNMP trap receiver, and it can be useful when testing your configuration. For example, Figure 2.2 lists a few of the SNMP traps sent to the PRTG from the SRX switch.

Devices   Local probe   1st group   SRX   SNMP Trap Receiver				
TRAP MESSAGES				
Filter	Source	Agent	Enterprise	Bindings
Items: 50    Date Range: 2014-10-09 23:08				
17/01/2015 22:51:16	192.168.0.1			SNMPv2-MIB::snmpTrapOID.0 = SNMPv2-SMI-v1:enterprises.2636.4.5.0.1 SNMPv2-SMI-v1:enterprises.2636.3.18.1.7.1.2.21 = 3327050 SNMPv2-SMI-v1:enterprises.2636.3.18.1.7.1.3.21 = 43 SNMPv2-SMI-v1:enterprises.2636.3.18.1.7.1.4.21 = 2 SNMPv2-SMI-v1:enterprises.2636.3.18.1.7.1.5.21 = root SNMPv2-SMI-v1:enterprises.2636.3.18.1.7.1.6.21 = SNMPv2-MIB::snmpTrapEnterprise.0 = SNMPv2-SMI-v1:enterprises.2636.1.1.1.2.41
17/01/2015 22:55:30	192.168.0.1			SNMPv2-MIB::snmpTrapOID.0 = RFC1253-MIB::ospf.16.2.16 RFC1253-MIB::ospfRouterId.0 = 192.168.0.1 RFC1253-MIB::ospfRtrAddr.193.133.41.29.0 = 193.133.41.29 RFC1253-MIB::ospfAddresslessIf.193.133.41.29.0 = 0 RFC1253-MIB::ospfRtrState.193.133.41.29.0 = down (I) SNMPv2-MIB::snmpTrapEnterprise.0 = SNMPv2-SMI-v1:enterprises.2636.1.1.1.2.41
17/01/2015 22:55:30	192.168.0.1			SNMPv2-MIB::snmpTrapOID.0 = RFC1253-MIB::ospf.16.2.2 RFC1253-MIB::ospfRouterId.0 = 192.168.0.1 RFC1253-MIB::ospfNbrAddr.193.133.41.1.0 = 193.133.41.1 RFC1253-MIB::ospfNbrAddresslessIndex.193.133.41.1.0 = 0 RFC1253-MIB::ospfNbrRtrId.193.133.41.1.0 = 193.133.41.1 RFC1253-MIB::ospfNbrState.193.133.41.1.0 = down (I) SNMPv2-MIB::snmpTrapEnterprise.0 = SNMPv2-SMI-v1:enterprises.2636.1.1.1.2.41
17/01/2015 22:55:30	192.168.0.1			SNMPv2-MIB::snmpTrapOID.0 = RFC1253-MIB::ospf.16.2.2 RFC1253-MIB::ospfRouterId.0 = 192.168.0.1 RFC1253-MIB::ospfNbrAddr.193.133.41.30.0 = 193.133.41.30 RFC1253-MIB::ospfNbrAddresslessIndex.193.133.41.30.0 = 0 RFC1253-MIB::ospfNbrRtrId.193.133.41.30.0 = 5.5.0.5 RFC1253-MIB::ospfNbrState.193.133.41.30.0 = down (I) SNMPv2-MIB::snmpTrapEnterprise.0 = SNMPv2-SMI-v1:enterprises.2636.1.1.1.2.41
17/01/2015 22:55:31	192.168.0.1			SNMPv2-MIB::snmpTrapOID.0 = SNMPv2-MIB::snmpTraps.4 RFC1213-MIB::ifIndex.508 = 508 RFC1213-MIB::ifAdminStatus.508 = up (I) RFC1213-MIB::ifOperStatus.508 = up (I) IF-MIB::ifName.508 = fe-0/0/0
17/01/2015 22:55:31	192.168.0.1			SNMPv2-MIB::snmpTrapOID.0 = SNMPv2-MIB::snmpTraps.3 RFC1213-MIB::ifIndex.508 = 508 RFC1213-MIB::ifAdminStatus.508 = up (I) RFC1213-MIB::ifOperStatus.508 = down (I) IF-MIB::ifName.508 = fe-0/0/0

Figure 2.2      Received SNMP Traps

As you can see, these trap messages appear to be just a string of numbers and characters. The numbers are the OID from the MIB and this is exactly what SNMP agents send to the SNMP manager. In order for the SNMP server to translate these numbers into meaningful messages, an administrator will need to download MIB translation files and add these to the SNMP server.

NOTE      For Juniper devices, MIB translation files can be downloaded from: [http://www.juniper.net/techpubs/software/index\\_mibs.html](http://www.juniper.net/techpubs/software/index_mibs.html).

As an SNMP server, a PRTG monitor can request MIB information from the device. An example of this is shown in Figure 2.3 where the

SNMP manager has requested information related to the system uptime and the bandwidth usage, and the result of the information sent as a Get response by the agent is shown in meter form.

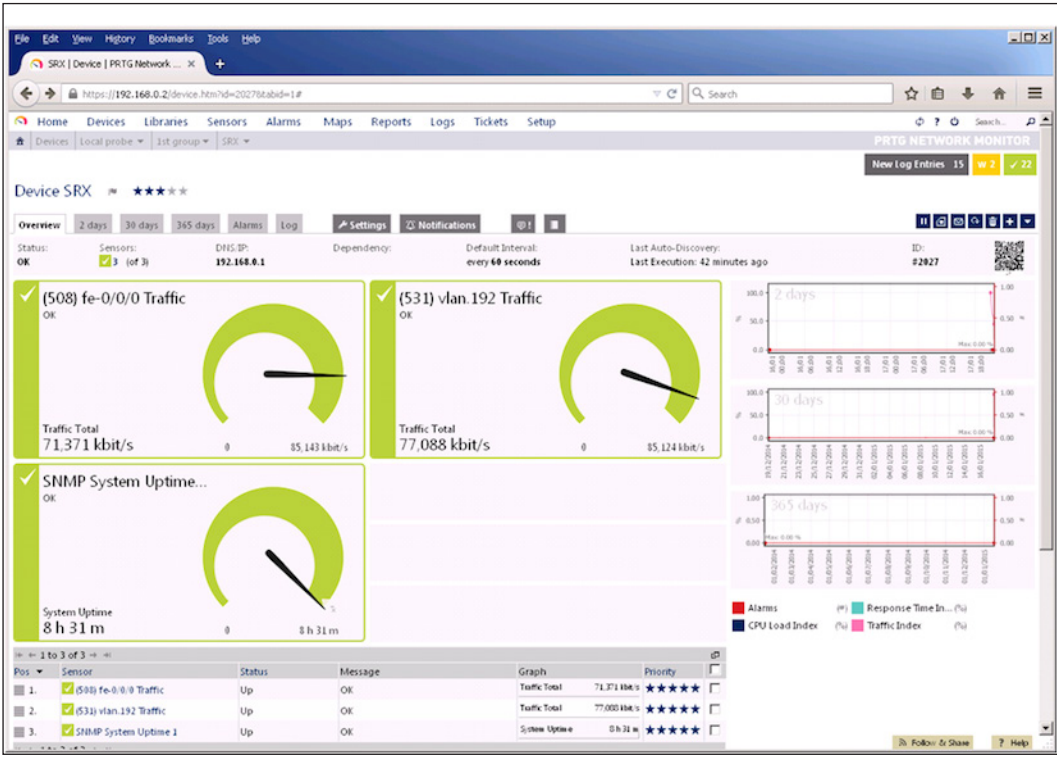


Figure 2.3 SNMP Agent Data Received by the SNMP Receiver

### Configuring SNMP Version 3 on Devices Running Junos OS

Although it's possible to run SNMP Version 2c and Version 3 together in a live environment, you may wish to remove Version 2c prior to configuring Version 3 in the lab in order to simplify your testing. Simply run the command:

```
delete snmp
```

In our scenario, however, the SRX runs Version 2c and the EX switch will be configured to run Version 3 instead. The first thing to do is add the SNMP agent to the device. If you recall, adding Version 2c was relatively straightforward, however, because of the security enhancements Version 3 offers, configuring it is a little harder.

**NOTE** The minimum configuration required to add SNMP Version 3 to a device running Junos OS can be found in the Juniper Tech Library, located here: [http://www.juniper.net/techpubs/en\\_US/junos14.1/topics/task/configuration/snmpv3-minimum-config-junos-nm.html](http://www.juniper.net/techpubs/en_US/junos14.1/topics/task/configuration/snmpv3-minimum-config-junos-nm.html).

To keep things simple, let's use the username *admin* with the password of *secretpassword* (in a live environment, of course, it is best practice to create a complex username and password). The following command creates this user and tells Junos OS to use SHA for authentication, and in addition, specifies that the security model be a USM, or User-based Security Model, as opposed to using a community:

```
set snmp v3 usm local-engine user admin authentication-sha authentication-  
key secretpassword
```

Next, you need to create a view in order to configure the View-based Access Control Model. First, create an access group, which in this case is named `SNMPV3TESTGROUP`, then specify which context should be used. Contexts allow a logical partition of SNMP access, useful if a device is used by multiple customers in a multi-tenancy building, however this is a small LAN, so the default context will be used. The desired security model comes after the context, and you can use SNMP Version 1 or Version 2 security, or you can use the user-based security, or by using the keyword *any*, you can allow any model. Next the security-level is specified, and with it you can specify that this view should use `noAuthnoPriv`, `AuthnoPriv` or `AuthPriv`. This penultimate option sets what that view is able to do such as read, write (where you can change the device configuration), or notify (meaning send trap messages). In this case, all three options are used, therefore the command is run three times with the different options. Finally, the view is given a name, which in this example is `SNMPV3TESTVIEW`:

```
set snmp v3 vacm access group SNMPV3TESTGROUP default-context-prefix security-  
model any security-level authentication read-view SNMPV3TESTVIEW  
set snmp v3 vacm access group SNMPV3TESTGROUP default-context-prefix security-  
model any security-level authentication write-view SNMPV3TESTVIEW  
set snmp v3 vacm access group SNMPV3TESTGROUP default-context-prefix security-  
model any security-level authentication notify-view SNMPV3TESTVIEW
```

The next part of the Version 3 configuration is to map the user created earlier with the group:

```
set snmp v3 vacm security-to-group security-model usm security-  
name admin group SNMPV3TESTGROUP
```

```
set snmp v3 snmp-community SNMPV3TESTCOMMUNITY security-name admin
```

Finally, you need to specify which components of the device, or “objects” a view has access to. The following command allows access to every object. However, this can be controlled by specifying different objects (the details of which are covered in the upcoming discussion on creating SNMP Version 3 traps):

```
set snmp view SNMPV3TESTVIEW oid iso include
```

This concludes how to configure the agent. Now the next task is to create the traps to notify the SNMP server of events. To begin, the target address should be specified. When creating the target address, you need to name the server, and for this example *SNMPV3SERVER* is used. Next, give the server a tag. This helps ease of later administration because you can give multiple servers the same tag then tell Junos OS that traps should be sent to any servers with this tag. The next section defines the parameters Junos OS uses to connect to the SNMP server, and labels them. However in this step let’s just tell Junos OS which label to use for this server:

```
set snmp v3 target-address SNMPV3SERVER address 192.168.0.2
set snmp v3 target-address SNMPV3SERVER tag-list SNMPV3RECEIVER
set snmp v3 target-address SNMPV3SERVER target-parameters SNMPV3TARGETPARAMETERS
```

As mentioned previously, here is where you specify the parameters Junos OS uses to connect to the server. In this case, the version will be Version 3. However, if an SNMP server does not support Version 3 messages, then Version 2c or Version 1 can be specified. In these parameters, the security model and level are the same as described when the views were discussed, and the second-to-last command tells Junos OS to connect to the server using the username *admin*. The last command tells Junos OS which filter should be used when deciding which traps are to be sent to the server (this filter is created in a subsequent section):

```
set snmp v3 target-parameters SNMPV3TARGETPARAMETERS parameters message-processing-model v3
set snmp v3 target-parameters SNMPV3TARGETPARAMETERS parameters security-model usm
set snmp v3 target-parameters SNMPV3TARGETPARAMETERS parameters security-level authentication
set snmp v3 target-parameters SNMPV3TARGETPARAMETERS parameters security-name admin
set snmp v3 target-parameters SNMPV3TARGETPARAMETERS notify-filter SNMPV3TRAPFILTER
```

The next step is to create a notify group, a group to tell Junos OS to send the event information to the SNMP server as a trap. The other option would be to send the event as an *inform*, depending on what your SNMP server supports, with informs intended to replace traps. The notify group also tells Junos OS where this event should be sent and this is done by using the Tag mentioned when creating the target:

```
set snmp v3 notify SNMPV3TRAPS type trap
set snmp v3 notify SNMPV3TRAPS tag SNMPV3RECEIVER
```

When you configured traps using Version 2c, you could use the context sensitive help to tell you what traps were available. With Version 3, unfortunately, you no longer have this option and you need to specify the name of the object from which you wish to receive traps. Happily, Juniper has made a list of these objects, also known as *MIBs* available here: <http://contentapps.juniper.net/mib-explorer/navigate.jsp>. These objects are in a basic hierarchy. At the top of the hierarchy is the object *iso* and beneath this is a tree structure. If you specify part of the tree, the device running Junos OS will send traps from that component plus everything below it. Figure 2.4 shows a screenshot of the MIB explorer from Juniper Networks website. If you configure Junos OS to send traps from the MIB *internet hierarchy*, it would include *mgmt*, *mib-2*, and so on, but would ignore the *org* and *dod* hierarchies.

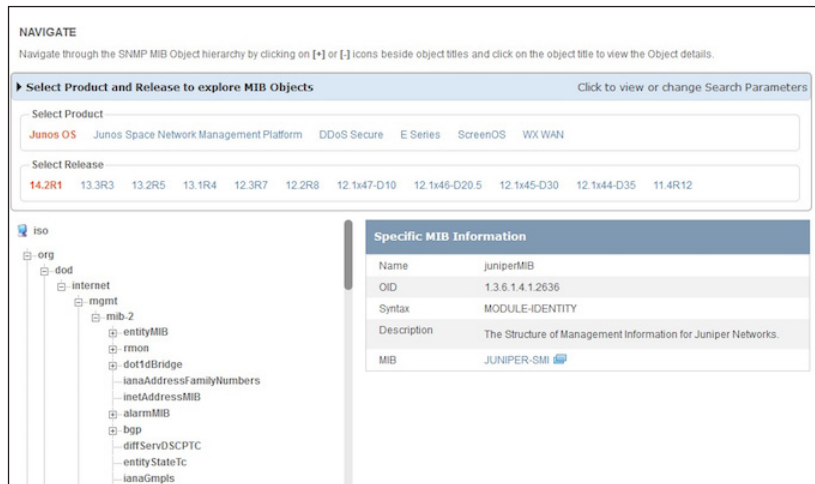


Figure 2.4 Object or MIB Explorer

In theory you could just configure Junos OS to send traps from *iso*, which would include traps from every component, but this may send too much information so it makes sense to specify a component further down the tree. In this example, you configure Junos OS to send traps from *mgmt*, which will also include interfaces, traps from the OSPF process, and so on:

```
set snmp v3 notify-filter SNMPV3TRAPFILTER oid mgmt include
```

You can exclude some components by specifying the *exclude* keyword. This command will exclude the object *bgp*:

```
set snmp v3 notify-filter SNMPV3TRAPFILTER oid bgp exclude
```

Once this configuration has been committed you should verify how SNMP has been configured by running the `show snmp v3` command. If you look under the Filter section, which is in bold in the below output, you can see what objects have been included and excluded. Notice that the object names of *mgmt* and *bgp* have been replaced with their numerical equivalent:

```
{master:0}[edit]
root@# run show snmp v3
```

```
Local engine ID: 80 00 0a 4c 01 c0 a8 00 fe
Engine boots:      2
Engine time:       2189 seconds
Max msg size:      65507 bytes
```

Engine ID: local

User	Auth/Priv	Storage	Status
admin	sha/none	nonvolatile	active
Group name	Security model	Security name	Storage type
SNMPV3TESTGROUP	usm	admin	nonvolatile active

Access control:

Group	Context prefix	Security model/level	Read view	Write view	Notify view
SNMPV3TESTGROUP		any/authent	SNMPV3TEST	SNMPV3TES	SNMPV3TEST

SNMP Target:

Address name	Address	Port	Parameters name	Storage type	Status
SNMPV3SERVE	192.168.0.2	162	SNMPV3TARGE	nonvolatile	active

Parameters name	Security name	Security model/level	Notify filter	Storage type	Status
SNMPV3TARGETPA	admin	usm/authent	SNMPV3T	nonvolatile	active

SNMP Notify:

Notify name	Tag	Type	Storage type	Status
SNMPV3TRAPS	SNMPV3RECEIVER	trap	nonvolatile	active

Filter name	Subtree	Filter type	Storage type	Status
SNMPV3TRAPFILTER	1.3.6.1.2	include	nonvolatile	active
SNMPV3TRAPFILTER	1.3.6.1.2.1.15	exclude	nonvolatile	active

Community index	Security name	Context name	Tag	Storage type	Status
SNMPV3TESTCOMMUNITY	admin			nonvolatile	active



All that remains now is ensuring that the traps are being received by the server, and that the server can connect to the agent. Use the same `show snmp statistics` command you used for Version 2c to verify that Version 3 is sending and receiving SNMP data:

```
{master:0}[edit]
root@# run show snmp statistics
SNMP statistics:
  Input:
    Packets: 1178, Bad versions: 0, Bad community names: 0,
    Bad community uses: 0, ASN parse errors: 0,
    Too bigs: 0, No such names: 0, Bad values: 0,
    Read onlys: 0, General errors: 0,
    Total request varbinds: 2573, Total set varbinds: 0,
    Get requests: 477, Get nexts: 112, Set requests: 0,
    Get responses: 0, Traps: 0,
    Silent drops: 0, Proxy drops: 0, Commit pending drops: 0,
    Throttle drops: 0, Duplicate request drops: 0
  V3 Input:
    Unknown security models: 0, Invalid messages: 0
    Unknown pdu handlers: 0, Unavailable contexts: 0
    Unknown contexts: 0, Unsupported security levels: 0
    Not in time windows: 0, Unknown user names: 0
    Unknown engine ids: 589, Wrong digests: 0, Decryption errors: 0
  Output:
    Packets: 629, Too bigs: 0, No such names: 0,
    Bad values: 0, General errors: 0,
    Get requests: 0, Get nexts: 0, Set requests: 0,
    Get responses: 589, Traps: 40
```

By studying this output, it may be noted that there are a number of unknown “engine id” packets because the SNMP server is asking for data for an object that is not part of Junos OS. In this situation, the configuration of the SNMP sensors needs to be checked and the offending sensor removed.

As a final verification, let’s check the SNMP server, and you can see the reception of live data from the EX switch, indicating that SNMP Version 3 is working as expected.

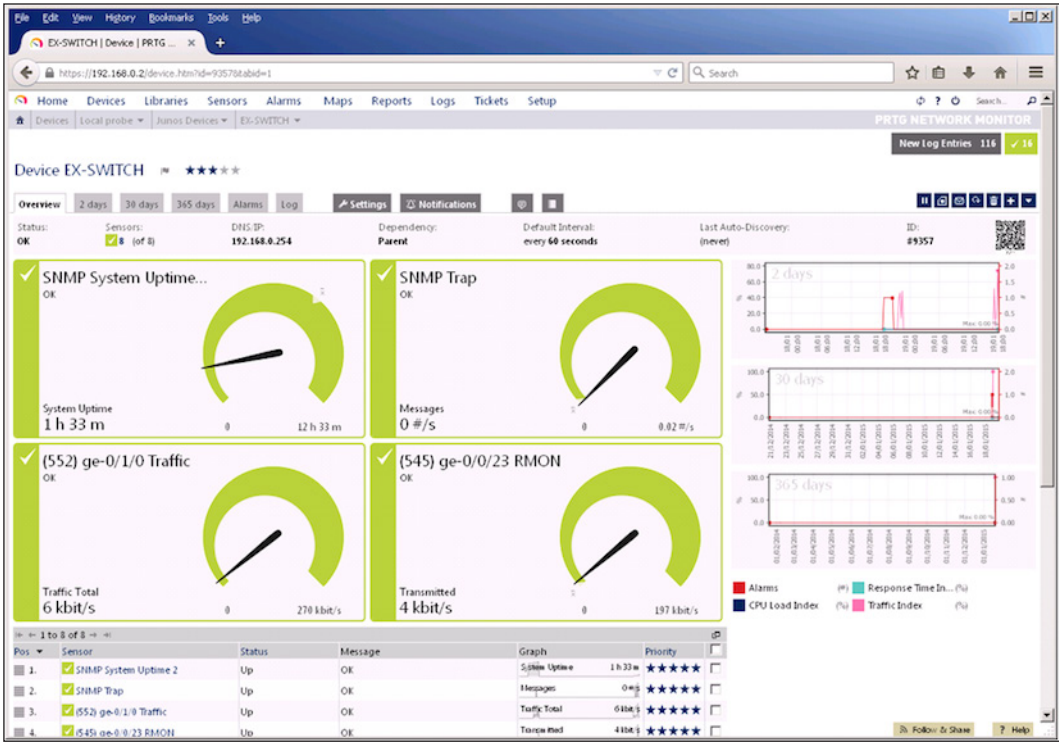


Figure 2.5 Verifying EX Switch

## Summary

SNMP is a finishing touch in the configuration of devices running Junos OS. This chapter has shown you how to configure a sample network. Once you have your network configured, make sure every device you add is tied in to the SNMP configuration. If you need to upgrade to SNMP Version 3, it should be done in the lab first, then taken into your production environment.

# Chapter 3

## The Gold Mine That Is Syslog

Think back to the scenario on NTP in Chapter 1 where we created a hypothetical situation in which an implemented change had caused a network issue, and we then discussed how the support team might resolve the issue by looking at the various logs that were available on each device. The problem is that connecting to each device takes time, and in today's modern networks trying to achieve 99.999% availability, issues need to be resolved as quickly as possible. What is needed is a way of collating the log files from all the network devices and storing them at a central location so that the information can be correlated and searched to help determine where the problem initially started. What is needed is a System Log server, more commonly known as a *Syslog* server.

Syslog's primary purpose is to collect log information. However, as you shall see later, if every device blindly sent every piece of information it logged the storage required would be quite substantial and an excessive amount of network traffic would be generated. More important, perhaps, is the concern that if an engineer is required to trawl through a large amount of data during a crisis, there is a chance that the important information could be obfuscated.

The log is stamped with not just the date and time, but also with a severity level and a facility whenever a network device sends data to a Syslog server. The severity level relates to the urgency of the message in the log (don't confuse this urgency with Quality of Service and it doesn't mean that the message will be sent across the network at a higher priority). Severity simply gives the administrator a way of indicating which message is more important and which is just general information. The levels are numbered from 0 to 7, with appropriate levels of severity as listed in Table 3.1.

Table3.1 Syslog Severity Levels

Code	Severity
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice
6	Informational
7	Debug

The facility relates to what component or service on the network device generated the message and it is possible to specify that you only want to see logs from, say, the firewall facility, which would show you logs from firewall filtering, or from the change log which would only show logs from when the configuration changes. It's also possible to configure a device to send logs from multiple facilities or to specify all by using the *any* command.

Let's investigate this finishing touch in Junos OS.

## Local Log Files in Junos OS

As with most good networking equipment, devices running Junos OS store local log files, recording errors and information that may be of interest to an administrator. By default, Junos OS will log messages that are at the critical severity level to a file named *messages*. If you study the default configuration for logging, a copy of which is below, you may notice something interesting:

```
system {
  syslog {
    user * {
      any emergency;
    }
    file messages {
      any critical;
      authorization info;
    }
    file interactive-commands {
      interactive-commands error;
    }
  }
}
```

What is interesting is the configuration for the local log files is located at the [edit system syslog] hierarchy level. The reason for this is because Junos OS is its own Syslog server. This means Junos OS runs a Syslog daemon and when an event that needs to be logged occurs, it sends a syslog message to itself, storing it in the appropriate file.

Should you wish to change the level of logging from the default, you can enter the `set system syslog file messages any... command` followed by the level you wish to log from (use the levels listed in Table 3.1). And to view these logs, enter the `show log messages` command.

**MORE?** As this chapter is really about logging to an external server, it doesn't cover local logs other than what has already been discussed. For more information about local logging use this kb at the Juniper knowledge base:

<http://kb.juniper.net/InfoCenter/index?page=content&id=KB16502>.

## Configuring Syslog on Devices Running Junos OS

Some Syslog servers require you to tell the server which devices are allowed to log to it. This is actually a useful feature that helps to stop an attacker flooding your Syslog server with false messages. If your device has a backup path to the Syslog server, then it may be an idea to add a loopback address, which will always be up, regardless of which physical interface fails. The following command tells Syslog which address to use as the source, which in this case is the SVI of VLAN 192 on the EX switch:

```
set system syslog source-address 192.168.0.254
```

Now that the source has been set, the following command tells Junos OS the IP address of the Syslog server. The third word in the command, `host`, tells Junos OS that this is a server to send logs to. The fourth command option is the IP address of the server and the fifth is the facility; here a list of all possible facilities is shown using the help prompt:

```
{master:0}[edit]
root@EX-SWITCH# set system syslog host 192.168.0.3 ?
Possible completions:
  allow-duplicates  Do not suppress the repeated message
  any              All facilities
+ apply-groups     Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
  authorization    Authorization system
  change-log       Configuration change log
  conflict-log     Configuration conflict log
  daemon          Various system processes
  dfc             Dynamic flow capture
```

exclude-hostname	Exclude hostname field in messages
explicit-priority	Include priority and facility in messages
external	Local external applications
facility-override	Alternate facility for logging to remote host
firewall	Firewall filtering system
ftp	FTP process
interactive-commands	Commands executed by the UI
kernel	Kernel
log-prefix	Prefix for all logging to this host
match	Regular expression for lines to be logged
ntp	NTP process
pfe	Packet Forwarding Engine
port	Port number
security	Security related
source-address	Use specified address as source address
> structured-data	Log system message in structured format
user	User processes

In our scenario, the *any* facility will be specified, meaning all facilities will be logged. The last part of the command relates to the severity level. Setting *any* (again) as the severity level means send all severities, from debug to emergency:

```
set system syslog host 192.168.0.3 any any
```

Should a network engineer so desire, it is perfectly possible to configure Junos OS to send logs from one facility to one server and logs from a different facility to another server, or to specify all facilities with a severity of one level and then specify to the same server a more specific facility with a severity of a lower level. Here's an example of these commands:

```
set system syslog host 192.168.0.3 any critical
set system syslog host 192.168.0.3 authorization info
set system syslog host 192.168.0.4 firewall notice
```

**NOTE** Since Junos OS Release 9.6, it has been possible to specify two remote Syslog servers in addition to the local logs. This is useful for purposes of redundancy or if an engineer wishes to reserve a Syslog server for sending debug messages.

As an optional command, you can also adjust how the time stamp is sent to the Syslog server. It's possible to tell Junos OS to include milliseconds or the year in the time stamp:

```
set syslog time-format millisecond
set system syslog time-format year
```

## Verifying Syslog Operation

The best way to verify if the configuration is correct is to open the monitor window and perform some administrative functions on the network device. If a commit is performed on the device, this will generate some messages, a sampling of which is shown in Figure 3.1.

Date	Time	Priority	Hostname	Message
01-21-2015	18:54:48	Crit,Info	192.168.0.1	Jan 21 18:55:00 SRX cron[1402]: [root] CMD ( /usr/libexec/atrun)
01-21-2015	18:54:47	Crit,Info	192.168.0.254	Jan 21 18:55:00 EX-SWITCH /usr/sbin/cron[1255]: [root] CMD ( /usr/libexec/atrun)
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: signaling 'Alarm control process', pid 106 enabled
01-21-2015	18:53:17	Local? Warning	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_COMPLETED: commit complete
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: commit complete
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: child notifying daemons of new configuration...
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: rotating daemons of new configuration...
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: signaling 'Simple Network Management P...
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: Rotate backup config...
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: activating '/usr/run/db/janiper.data'
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: db_groups_info_clear done
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: db_groups_info_clear start
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: db_check_constraint_ids_clear done
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: db_check_constraint_ids_clear start
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: copying configuration to juniper.save
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: finish ftp activate
01-21-2015	18:53:17	Local? Info	192.168.0.254	Jan 21 18:53:30 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: not executing ui_commit in rc.ui
01-21-2015	18:53:16	Local? Info	192.168.0.254	Jan 21 18:53:29 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: executing foreign_commands
01-21-2015	18:53:16	Local? Info	192.168.0.254	Jan 21 18:53:29 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: activating '/usr/etc/certs'
01-21-2015	18:53:16	Local? Info	192.168.0.254	Jan 21 18:53:29 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: activating '/usr/etc/asn'
01-21-2015	18:53:16	Local? Info	192.168.0.254	Jan 21 18:53:29 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: activating '/usr/etc/pam.conf'
01-21-2015	18:53:16	Local? Info	192.168.0.254	Jan 21 18:53:29 EX-SWITCH mgd[1232]: UI_CHILD_STATUS: Cleanup child '/usr/sbin/ftp', PID 1245, status 0
01-21-2015	18:53:15	Local? Info	192.168.0.254	Jan 21 18:53:28 EX-SWITCH mgd[1232]: UI_CHILD_START: Starting child '/usr/sbin/ftp'
01-21-2015	18:53:15	Local? Info	192.168.0.254	Jan 21 18:53:28 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: start ftp activate
01-21-2015	18:53:15	Local? Info	192.168.0.254	Jan 21 18:53:28 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: commit wrapup...
01-21-2015	18:53:15	Local? Info	192.168.0.254	Jan 21 18:53:28 EX-SWITCH mgd[1232]: UI_CHILD_STATUS: Cleanup child '/usr/sbin/ftp', PID 1245, status 0
01-21-2015	18:53:14	Local? Info	192.168.0.254	Jan 21 18:53:27 EX-SWITCH mgd[1232]: UI_CHILD_START: Starting child '/usr/sbin/ftp'
01-21-2015	18:53:14	Local? Info	192.168.0.254	Jan 21 18:53:27 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: daemons checking new configuration
01-21-2015	18:53:14	Local? Info	192.168.0.254	Jan 21 18:53:27 EX-SWITCH mgd[1232]: UI_CHILD_STATUS: Cleanup child '/usr/sbin/ftp', PID 1236, status 0
01-21-2015	18:53:13	Local? Info	192.168.0.254	Jan 21 18:53:26 EX-SWITCH mgd[1232]: UI_CHILD_START: Starting child '/usr/sbin/ftp'
01-21-2015	18:53:13	Local? Info	192.168.0.254	Jan 21 18:53:26 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: dropping unchanged foreign files
01-21-2015	18:53:13	Local? Info	192.168.0.254	Jan 21 18:53:26 EX-SWITCH mgd[1232]: UI_COMMIT_PROGRESS: Commit operation in progress: complete foreign files

Figure 3.1 Syslog Messages After a Commit

It is quite evident that the output is verbose, probably too much for a simple commit, therefore it would be best practice to tune the severity level of the logging so that you still get a few messages, but certainly fewer than you would get with the level set as any.

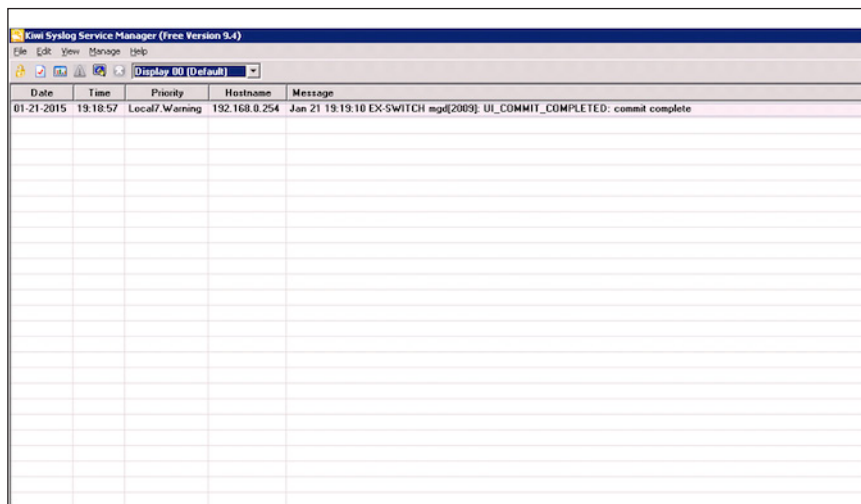
If you study the output closely, specifically the Priority column, you can see that most of the messages had a severity of *informational*. If further scrutiny is paid to the output, one may notice that there is one message with a severity of *warning*, which states commit complete. So by looking at this you can configure the severity level to anything above *any* or *informational*, to reduce the output, but still send a message that an administrator committed a configuration.

The level above *informational* is *notice*, so the logical option is to set the logging level to *notice*, as shown in the following command:

```
set system syslog host 192.168.0.3 any notice
```

There is no need to delete the existing setting as this command will replace it.

Now, if the commit is run once more, the output is drastically reduced as is shown in Figure 3.2



The screenshot shows the Kiwi Syslog Service Manager (Free Version 9.4) window. The menu bar includes File, Edit, View, Message, and Help. Below the menu bar is a toolbar with icons for file operations and a dropdown menu currently set to 'Display: 00 (Default)'. The main area is a table with the following columns: Date, Time, Priority, Hostname, and Message. A single log entry is visible in the first row.

Date	Time	Priority	Hostname	Message
01-21-2015	19:18:57	Local7:Warning	192.168.0.254	Jan 21 19:19:10 EX-SWITCH mgd[2009]: UI_COMMIT_COMPLETED: commit complete

Figure 3.2 Syslog Output with Severity Level Set as Notice



# Chapter 4

## Authentication, Authorization, and Accounting

An Authentication, Authorization, and Accounting (AAA or Triple A) server, like its name suggests, performs its three functions by first allowing the server to identify a user attempting to access the system, usually in the form of a username and password. Then the authorization determines whether a user is allowed to log in and what the user is allowed to do if the server has granted them access to the device. Finally, accounting means that the device records when the user logged in, what the user did while logged in, and what time the user disconnected. (For network engineers, accounting can also be used for reporting purposes in the event a change has had an unexpected side effect.)

Primarily, AAA servers are used to authenticate dial-up and VPN connections while keeping track how much to charge a company or individual for the time they were connected to a service. This chapter will explore something a little less visible.

There are two types of AAA server available today: TACACS+, and RADIUS. TACACS+ is a proprietary protocol, developed by Cisco systems and is typically available in a Cisco ACS server. RADIUS is a standards protocol, and as such there are many RADIUS servers from plenty of vendors. Some servers are designed to be simple solutions allowing authentication to a small number of devices and others, such as Juniper Networks SBR Enterprise Edition (<http://www.juniper.net/us/en/products-services/ipc/sbr-enterprise-series/sbr-ee/>) can support 1,500 transactions per second.

Junos OS based devices support both TACACS+ for networks where a Cisco ACS server has been deployed, and RADIUS, for networks where a RADIUS server is used. This book focuses on configuring Junos OS to authenticate using a RADIUS server. To configure the device to authenticate with TACACS+, you would basically replace the keyword `radius` with `tacplus`.

**MORE?** For more information on how to configure Junos OS to authenticate with TACACS+ see Juniper's TechLibrary: [http://www.juniper.net/techpubs/en\\_US/junos14.1/topics/task/configuration/tacacs-authentication-configuring.html](http://www.juniper.net/techpubs/en_US/junos14.1/topics/task/configuration/tacacs-authentication-configuring.html).

## Why is AAA Necessary?

When configuring a device running Junos OS for the first time, or indeed, after applying the `request system zeroize` command, it is necessary to run the following command from the top level of the hierarchy before the new configuration can be committed:

```
set system root-authentication plain-text-password
```

This command helps secure access to the device from unauthorized personnel by providing a root user login.

The only problem is that this command has complete access to both the FreeBSD shell and Junos OS. Therefore, giving the details of the root user login to everyone that requires access to the device is considered risky. In addition to giving the same login to everyone, if a network engineer makes a change to the configuration, all the logs would show that a root user made the change and it would be hard to tell *who* was logged in to the device at that time.

To resolve the situation, you create a separate user account for each user and assign each account with varying degrees of rights, or *login class*. Let's see what classes exist and what they allow:

```
{master:0}[edit]
networkadmin@EX-SWITCH# set system login user fred class ?
Possible completions:
<class>          Login class
operator         permissions [ clear network reset trace view ]
read-only        permissions [ view ]
super-user       permissions [ all ]
unauthorized     permissions [ none ]
```

**NOTE** In addition to the four default classes, it is also possible to create custom classes. Creating custom classes is not covered in this book, but those interested in creating their own classes should read: [http://www.juniper.net/techpubs/en\\_US/junos14.1/topics/example/authentication-login-classes-configuring.html](http://www.juniper.net/techpubs/en_US/junos14.1/topics/example/authentication-login-classes-configuring.html).

Assigning everyone their own login and password works great up until a certain point, but what happens when the network grows? When a new network engineer joins the company or when one leaves, you very quickly realize that someone will be given the unenviable task of

connecting to every device and creating a new account or suspending the account of the engineer in question. To counter this situation, an administrator can deploy an *external* authentication server, known as an AAA, or *Triple A* server.

## Configuring RADIUS Authentication

Before configuring RADIUS authentication, information about the RADIUS server(s) should be configured on the device. As mentioned in the Preface, this scenario uses a Microsoft Windows 2008 R2 server configured as a domain controller with Network Policy Server or NPS installed. Once this is installed, two active directory groups are created: *Juniper Radius Super User* and *Juniper Radius Readonly*.

After that, two Active Directory (AD) users are created as follows:

The first user named *Network Admin* with a login ID *networkadmin*. This user is added to the AD group *Juniper Radius Super User*. The second user named *Network Readonly* with a login ID *networkreadonly*. This user is added to the AD group *Juniper Radius Readonly*.

Figure 4.1 shows these users being associated with their respective groups.

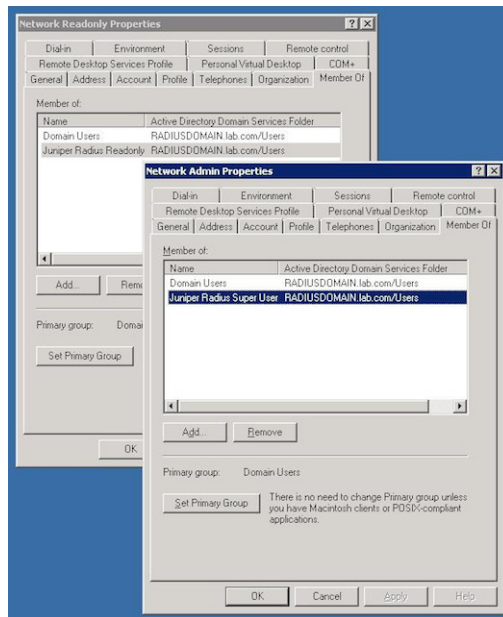


Figure 4.1 Windows Active Directory Users Network Readonly and Network Admin and Their Group Memberships

Once these users are created for the lab work, attention could then be turned to creating the policy within NPS. While this book is not meant to offer instruction on how to configure NPS, it includes a few screen-shots to help you visualize how the users within Windows are linked to the authorization part of AAA within Junos OS.

Figure 4.2 shows the network policies screen in the NPS snap-in. In this screen, the policy *Juniper Radius Super User* has been selected. In the middle pane, a user group is listed as a condition. This means that when a user who is a member of that group attempts to authenticate, the RADIUS server will match that condition and use that policy.

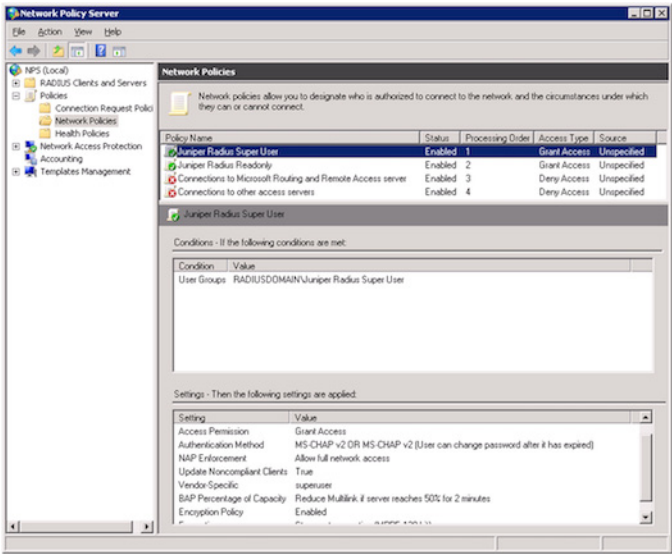


Figure 4.2 Network Policy Juniper Radius Super User Associated With an AD Group of the Same Name

In the last pane of Figure 4.2, the settings that the policy should use are displayed. Of these, the most important settings are *Access Permission* showing that access would be granted, *Authentication Method* showing that MS-CHAP V2 is used as the authentication method, and *Vendor Specific* that is set to *Super User*.

This Vendor Specific component is a custom attribute, added to the RADIUS server by the administrator. These attributes are useful as they allow the RADIUS server to tell the device specific information such as what rights a user will have when they log in, therefore in this case, within NPS, the Vendor Specific setting is applied with a key value of 1. By using this key, the login process is as follows:

1. The user supplies their username and password to Junos OS.

2. Junos OS asks the RADIUS server if this username and password are valid.
3. The RADIUS server confirms the username and password are valid.
4. The RADIUS server informs Junos OS the string value of vendor specific key 1.
5. Junos OS compares the vendor specific key 1 string with the device configuration.
6. Junos OS matches the string with the relevant user group and associates it with the correct login class.

Using our scenario as an example, when members who are part of the Windows AD group Juniper Radius SuperUser login, the RADIUS server tells Junos OS that the user should be assigned the key 1 string of SuperUser.

Figure 4.3, on the other hand, shows how the policy *Juniper Radius Readonly* is configured. In this case, the Vendor Specific or key 1 string is set as *readonly*.

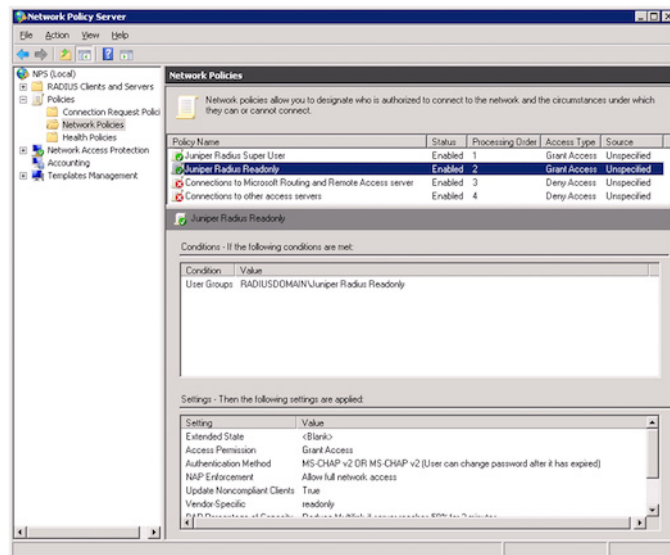


Figure 4.3 Network Policy Juniper Radius Readonly Associated With an AD Group of the Same Name

**NOTE** Not all RADIUS servers have the necessary attributes to support Juniper Networks devices out of the box, however, most RADIUS servers will allow you to add custom attributes. Juniper provides a list of the attributes you can enter into a radius server for this purpose:

[http://www.juniper.net/techpubs/en\\_US/junos14.1/topics/reference/general/radius-vendor-specific-attributes-juniper-networks.html](http://www.juniper.net/techpubs/en_US/junos14.1/topics/reference/general/radius-vendor-specific-attributes-juniper-networks.html).

Once the RADIUS server is set, configure the IP address of the server and the password or secret key to connect to the server. In this case, the server is accessible via 192.168.0.4 and the secret is *thisisapass-word*:

```
set system radius-server 192.168.0.4 secret thisisapassword
```

Next you specify the source address used to connect to the RADIUS server. As with Syslog and SNMP, this command is optional, but is useful if you have multiple paths to the server for redundancy purposes and as such, a loopback interface is used as the source:

```
set system radius-server 192.168.0.4 source-address 192.168.0.254
```

There are in fact two default ports a device uses to connect to a RADIUS server. The original port was UDP 1645, however, this conflicted with another service, so it was later changed to UDP 1812 in RFC 2138. By default, Junos OS uses port 1812 for communicating with the RADIUS server. However, it is possible to specify that Junos OS uses a port of your own choosing when the need arises, by using the port statement after the IP address of the RADIUS server. This can be seen on the output below, and the port can be anywhere between 1 and 65535, though it is recommended to use the default port to avoid conflicts:

```
{master:0}[edit]
networkadmin@EX-SWITCH# set system radius-server 192.168.0.4 port ?
Possible completions:
  <port>          RADIUS server authentication port number (1..65535)
```

**NOTE** RFC 6613 does mention RADIUS over TCP, however, this is only relevant to IPsec and TLS, and as neither of these are used with user authentication on a Junos device, this will not have any bearing in our scenario, so it is ignored for this use case.

When it comes to sending the user password to the RADIUS server, depending on the server software in use, it may support a number of different protocols. Currently, Junos OS supports PAP and MS-CHAP V2 only. If PAP sends the password in clear text, which is a security risk, MS-CHAP V2 is used. This is enabled using the following command:

```
set system radius-options password-protocol mschap-v2
```

Next, the device needs to be told to use the RADIUS server to authenticate and this is done by setting an authentication order. If the keyword password is set before “radius”, then Junos OS looks at the local user

database and if the user is not found, it will then contact the RADIUS server. By specifying that Junos OS can still use local authentication, an administrator can log in as the root user when necessary:

```
set system authentication-order [password radius]
```

Finally, we need to create the groups so that when the RADIUS server authenticates the user and sends key 1, Junos OS knows what authorization that user has. The command is similar to creating a local user, however, no password is specified. When adding the group the user name is the same name as the key 1 string sent from the RADIUS server:

```
set system login user readonly class read-only
set system login user superuser class super-user
```

All that remains to be done is to commit the configuration and test.

**NOTE** Further information on configuring Junos OS for RADIUS authentication can be found in Junipers TechLibrary: [http://www.juniper.net/documentation/en\\_US/junos14.1/topics/task/configuration/radius-authentication-configuring.html](http://www.juniper.net/documentation/en_US/junos14.1/topics/task/configuration/radius-authentication-configuring.html).

## Testing RADIUS Authentication

Although there is no command for checking whether the device has connected to the RADIUS server, the easiest way to check is to just log in as one of the users created in the configuration. Figure 4.4 shows two PuTTY sessions. The session in the foreground shows the user *networkreadonly* has logged in and is in the Junos OS CLI as opposed to the shell. The background session shows that the user *networkadmin* has successfully logged in, and again, is on the CLI.

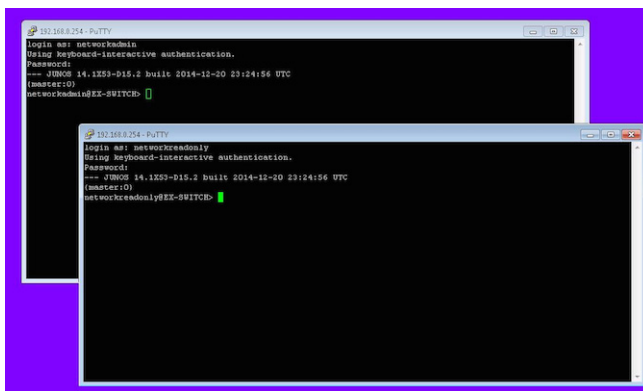


Figure 4.4 Test SSH Sessions

Logging in as a user is one thing, but determining whether the authorization process worked as expected is another. Let's use the session logged in as the username *networkreadonly* and attempt to look at the configuration:

```
{master:0}
networkreadonly@EX-SWITCH> show configuration
## Last commit: 2015-01-22 20:44:31 UTC by networkadmin
version /* ACCESS-DENIED */;
system { /* ACCESS-DENIED */ };
interfaces { /* ACCESS-DENIED */ };
snmp { /* ACCESS-DENIED */ };
routing-options { /* ACCESS-DENIED */ };
protocols { /* ACCESS-DENIED */ };
ethernet-switching-options { /* ACCESS-DENIED */ };
vlans { /* ACCESS-DENIED */ };
```

You can see that the user *networkreadonly* has been denied access, confirming that this user has read-only rights. Now, by running the `show configuration system radius-server` command while logged in as the *networkadmin* user, not only can we see the RADIUS server configuration, but we can also confirm that the *networkadmin* user has super-user rights:

```
{master:0}
networkadmin@EX-SWITCH> show configuration system radius-server
192.168.0.4 {
  secret $9$BZ/1Se7Nbs2a7-oGiHTQFn/9u1REyKWxcyoGi.zFcylévL; ## SECRET-DATA
  source-address 192.168.0.254;
```

The output is as expected and this is enough confirmation for our purposes that Junos OS has been correctly configured for RADIUS authentication. Now, to complete the configuration of AAA, you need to configure the last “A”: RADIUS accounting.

## Configuring RADIUS Accounting On Devices Running Junos OS

When configuring RADIUS accounting, although it's considered a security risk to set two passwords to be the same, Junos OS will allow the secret keyword to be the same as that used for authentication. But just as with authentication, it is wise to set the loopback interface as the source address for networks with redundant paths. In this scenario, the IP address of the SVI of VLAN 192 is used (purely to illustrate the command in use):

```
set system accounting destination radius server 192.168.0.4 secret thisisapassword
set system accounting destination radius server 192.168.0.4 source-
address 192.168.0.254
```



As with RADIUS authentication, there are two default ports the device could use to send RADIUS accounting information. The original port for RADIUS was UDP 1646, but this was changed to UDP 1813 in RFC 2138. By default, Junos OS uses port 1813 for RADIUS accounting but this port can be changed if needed, per the following command:

```
{master:0}[edit]
networkadmin@EX-
SWITCH# set system accounting destination radius server 192.168.0.4 port ?
Possible completions:
<port>          RADIUS server authentication port number (1..65535)
```

Finally, Junos OS needs to be told what information it should send to the RADIUS accounting server. There are three options:

- *login* meaning to send an event when a user logs in
- *change-log* referring to when an administrator changes the configuration
- *interactive-commands* meaning when an administrator issues a command that creates an output

In this chapter's scenario all three options will be used. In a live environment it may be the case that by specifying the *interactive-commands* option, too many messages may be sent to the RADIUS server causing excess traffic and increasing the size of the database and in that scenario only the login and change-log options would be used:

```
set system accounting events login
set system accounting events change-log
set system accounting events interactive-commands
```

These configuration changes can now be committed, after that you need to verify the RADIUS accounting setup.

## Testing RADIUS Accounting

Some RADIUS servers allow you to run reports and some RADIUS servers are just data collectors and an external utility is required to parse the RADIUS logs. In the case of Microsoft Windows NPS, an external application is required.

NPS is able to save accounting data to either an SQL database or a text file. For the purposes of this book, the NPS option "output to a text file" was set as "DTS compliant," although, ODBC compatible could be selected for importing into a database.

Let's assume a user connected to the EX switch, made a configuration change, and committed the configuration. Figure 4.5 shows the

information that was written to the text file. By studying the text file, it is *possible* to see the date and time of the device that was logged into, the IP address of the workstation that connected to the switch, and even finer details such as the network policy that was associated with the user. However you would need to study the output carefully in order to find the information you required and in the case of verbose output it would be difficult to find the information. Therefore, it is recommended (in this book's test case) to use a third party RADIUS log-parsing tool to make things easier to verify RADIUS Accounting. What Figure 4.5 does illustrate, however, is the *amount* of accounting data sent to the RADIUS server, indicating how useful collecting this information could be for any of a variety of purposes.

Figure 4.5 RADIUS Accounting Logs Saved as a DTS Compliant Text File

## Summary

As your network grows, AAA is a critical finishing touch to any device configuration. It allows a failsafe system for your team members to access and control devices with a convenient system that can be customized for your organization. This chapter provided one simple system, and you should experiment in the lab with this and more complex AAA setups that suit your network and its operation.

# Chapter 5

## Securing Device Management

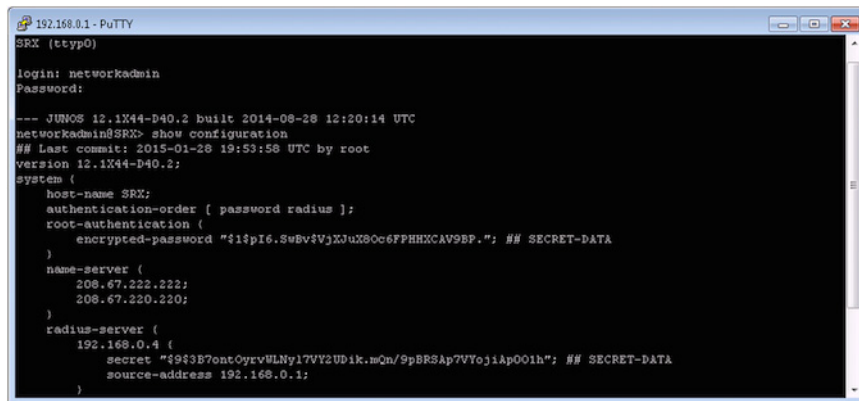
The previous chapters of this book showed you how to initiate Syslog, SNMP, and AAA accounting, and the book has also covered accurate reporting by configuring NTP, and addressed how to enhance security and ease administration by utilizing AAA authentication and authorization. That being the case, an administrator *could* justifiably feel that having implemented these technologies this device is now secure and if it isn't, then the administrator would be notified when an attacker had tried to gain unauthorized access.

The below output shows the default configuration of most of the devices running Junos OS:

```
root@> show configuration system services
ssh;
telnet;
xnm-clear-text;
web-management {
    http {
        interface vlan.0;
    }
    https {
        system-generated-certificate;
        interface vlan.0;
    }
}
```

If you study this configuration, it should be apparent that by default Junos OS allows an administrator to connect to the device using both SSH and Telnet, as well as HTTP and HTTPS, when connecting via the default VLAN. While this allows a network engineer to quickly implement the device utilizing a variety of methods, as you shall soon see, some of these methods may not be desirable in a live production environment.

Figure 5.1 shows a PuTTY session connected to a lab SRX using Telnet. In this illustration the user *networkadmin* has logged in (as created in Chapter 3). Once connected, the `show configuration` command was run and Junos OS displayed the configuration as requested. Note how the session begins with the words SRX (ttyp0).



```

192.168.0.1 - PuTTY
SRX (ttyp0)

login: networkadmin
Password:

--- JUNOS 12.1X44-D40.2 built 2014-08-28 12:20:14 UTC
networkadmin@SRX> show configuration
## Last commit: 2015-01-28 19:53:58 UTC by root
version 12.1X44-D40.2;
system {
    host-name SRX;
    authentication-order { password radius };
    root-authentication {
        encrypted-password "$1$pi6.SuBv4VjXJuX8Oc6FPHMXCAV9BP."; ## SECRET-DATA
    }
    name-server {
        208.67.222.222;
        208.67.220.220;
    }
    radius-server {
        192.168.0.4 {
            secret "$943B7ont0yrvULNy17VY2UD1k.mQn/9pBRSap7VYoj1Ap001h"; ## SECRET-DATA
            source-address 192.168.0.1;
        }
    }
}

```

Figure 5.1 Putty Session to an SRX Via Telnet

At the same time, as the user *networkadmin* was logging into the SRX, on another workstation a very useful application known as Wireshark was being run. Wireshark allows a network engineer to capture network packets and frames and then filter them. Under normal circumstances this would allow the engineer to analyze traffic, checking whether the correct traffic is reaching its destination, or to see what traffic is being sent. Wireshark can also be used by less savory personnel to see what information a company or individual is trying to send. In this case, the application was being run in the guise of the later.

**NOTE** Wireshark is not a requirement for configuring Junos OS, but it can be a useful tool for monitoring traffic flows, which may assist in diagnosing some network issues: <https://www.wireshark.org/>.

Figure 5.2 shows the packets that have been captured: the source and destination addresses show in which direction the traffic flow occurred. A filter has also been placed on the capture to show only those packets related to the Telnet session between the client and the SRX. What is interesting is the amount of packets sent. A Telnet client sends one packet for each character typed, but can send responses from the device in a single packet – therefore, in Figure 5.2 the highlighted packet shows the word *login*.

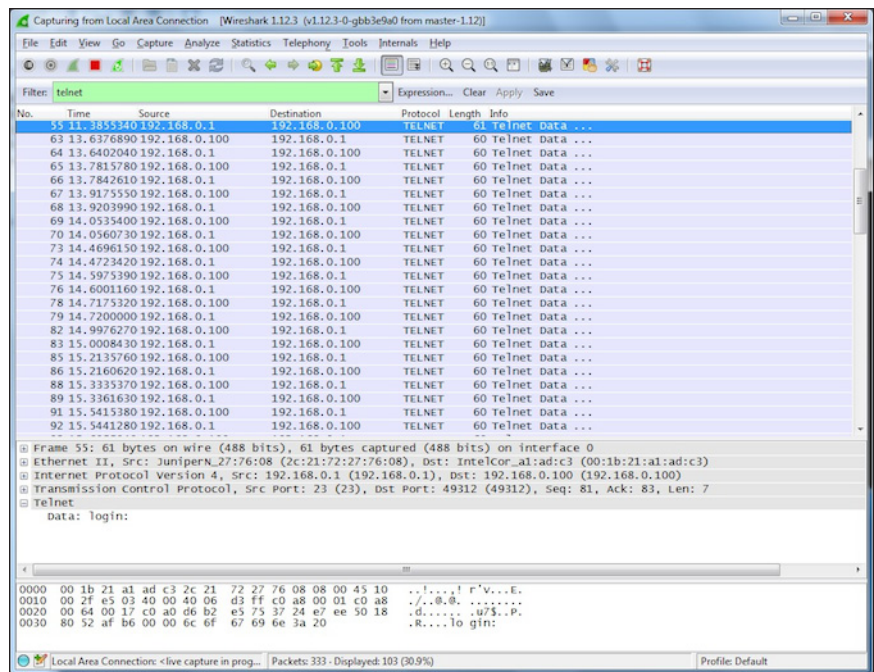


Figure 5.2 Wireshark With Captures Filtered for Telnet Packets

By going through each packet, you can begin to build up a picture of the data that has been received and transmitted. While this could take a bit of time to complete, as the person capturing this information would need to know about the headers and which information can be discarded, Wireshark has a useful tool to allow you to follow the stream of traffic.

If one were to right click on one of the captured packets, a menu appears allowing you to select the option *Follow TCP Stream*. This tells Wireshark to collate the relevant packets, join them, and then display the results in a separate screen, the result of which is shown in Figure 5.3. It's surprising how much information is passed between the device and the Telnet client. The first few lines of information are the Telnet client negotiating with the device, the client tells the device the speed it's expecting to use, the emulation it's using along with random characters you could assume to be a preamble, then at the fourth line some familiar text can be seen.

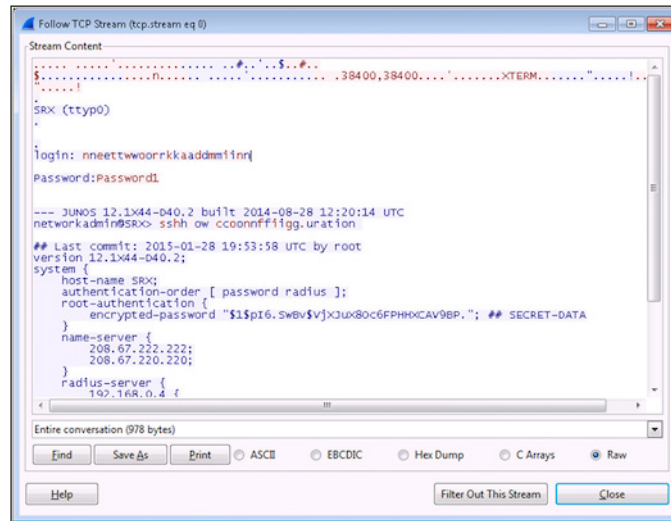


Figure 5.3 Results from Following the TCP Stream

You may recall from Figure 5.1, the text in the Telnet session started with the line *SRX (ttyp0)*. The fourth line in the captured text starts in exactly the same way. This is because Wireshark captured this text as it was sent to your terminal.

Below the *SRX (ttyp0)* message, there are a few periods, which one could assume to be simple carriage returns, then the message:

```
login: nneettwoorrrkaaddmmiinn
```

What is interesting is that each character is sent twice by the Telnet application, and an attacker who captured this would know to remove the duplicates giving them the username. After the username is the line *password:Password1*. This means that the person monitoring our session now has a username and password of a super user, and to make matters worse, as the login details reside on the AAA server; this in turn means the attacker could potentially connect to any network device serviced by that AAA server.

Should this capture be legitimate, then before closing this screen, the capture can be saved to a file for future reference, printed or viewed as HEX, ASCII, and so on by using the buttons at the bottom of the screen.

To add further insult to injury, that unauthorized person also has the potential to connect to the device from almost anywhere and in addition, no logs would be created to alert the administrator that an attacker was trying to gain access, because as far as Junos OS is concerned, these are valid login details.



In the configuration shown at the beginning of this chapter, it was evident that both Telnet *and* SSH were enabled by default, therefore, the same test should be run, but this time the connection will be via SSH or secure shell. Figure 5.4 shows PuTTY opening the SSH session to the SRX. Note that the session doesn't start with SRX (tty0) and instead has the line *Using keyboard-interactive authentication*, and this is a clear indicator that the session is using SSH as opposed to Telnet.

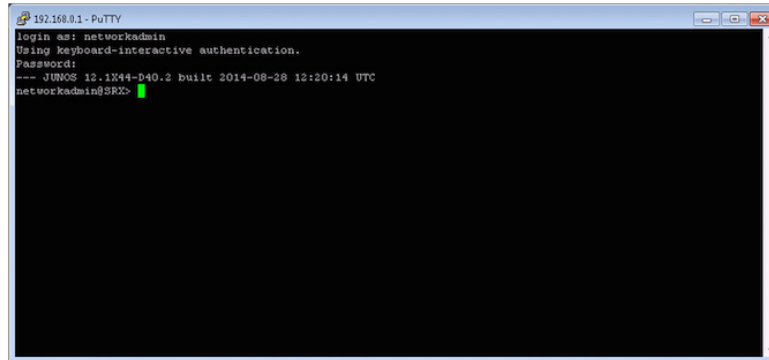


Figure 5.4 SSH Session to an SRX Using Putty

As with the Telnet session, network traffic was captured using Wireshark while the SSH session was being initiated. Wireshark was then set to filter all traffic apart from SSH. Just as before, the Follow TCP Stream tool was utilized and the packets were collated and joined and Figure 5.5 is the result.

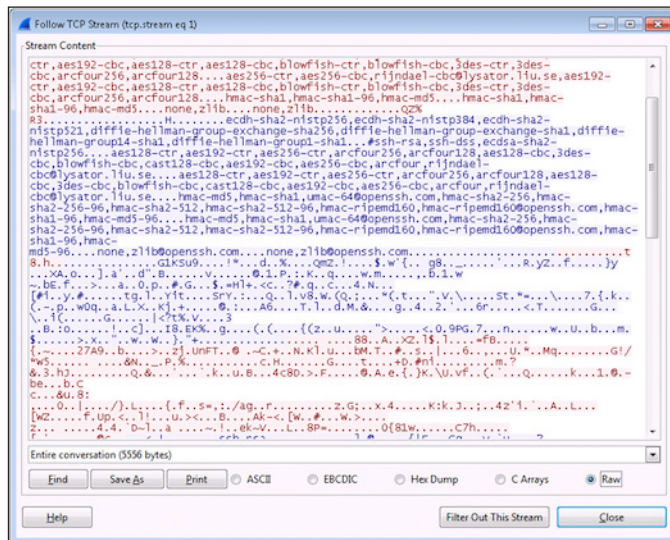


Figure 5.5 Results of the TCP Stream From the SSH Session

The first thing to notice is the amount of preamble as the session starts and in among the black-colored text, names such as *aes192*, and *aes128* can be clearly seen while in among the blue text, *diffie-helman-group-exchange-sha256* and *diffie-helman-group14-sha1* are visible. These phrases are the client and device negotiating which encryption to use, and in the case of the black and the blue text, the authentication method being used. Both of these sections are important because it means that the data being sent between the client and device is encrypted, and in addition, the username and password are hashed, protecting both data and authentication details from being compromised.

After the preamble, there are a lot of seemingly random characters, which appear to be useless data. This useless data is actually the username and password in a hashed form and the first few lines of text in the PuTTY session in an encrypted form.

Should you perform exactly the same test, this time comparing HTTP and HTTPS in a web browser connecting to J-Web on the device, you would expect to see similar results. Only in this case, using HTTP would allow the username, password, and data to be sent between the web browser and device in clear text, whereas by using HTTPS, you would be confident that your data would be protected by SSL/TLS encryption.

## Configuring Device Management

Disabling Telnet and HTTP on a Junos OS device requires two delete statements. That said, by default the device is enabled for both SSH Version 1, or to be more precise, Version 1.5 and Version 2. The difference between the two is that Version 1 has weaker integrity checks and authentication methods in addition to combining the authentication, transport, and connection protocols into a single protocol. Version 2 introduces stronger integrity checks and removed insecure authentication methods, therefore it makes sense to allow Version 2 only. The following commands enable SSH Version 2 as well as deleting Telnet and HTTP. Finally it sets HTTPS for the correct VLAN as, following best security practices, the default VLAN is not in use:

```
set system services ssh protocol-version v2
delete system services telnet
delete system services web-management http
delete system services web-management https interface vlan.0
set system services web-management https interface vlan.192
```

After committing, a verifying test should be performed to confirm that it is not possible to open a Telnet session to the device, but just as importantly, that it's still possible to open an SSH session as well as a HTTPS connection. Figure 5.6 shows a screen capture of the SSH connection being accepted but the Telnet session being rejected.



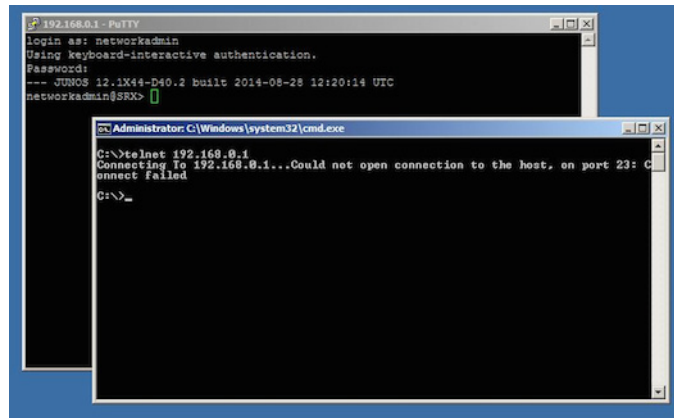


Figure 5.6 Testing Telnet Access Has Been Disabled

## Restricting Subnet Access

Once a person has the *Super User* username and password, there is very little to stop them from accessing the device from anywhere around the globe. As long as the attacker can reach the device, they have access. Thankfully, there are methods available to prevent anyone from outside your company, or indeed a single subnet, from creating an SSH or HTTPS session. In addition, as a best practice, you should also include Telnet, just in case an engineer accidentally enables it in system services.

In the following scenario, the attacker is now on the Internet somewhere. His device has the IP address 1.1.1.10 and he has already allowed himself access by editing the firewall rules so he is able to access your network. Figure 5.7 shows our rogue client out there in the cloud somewhere.

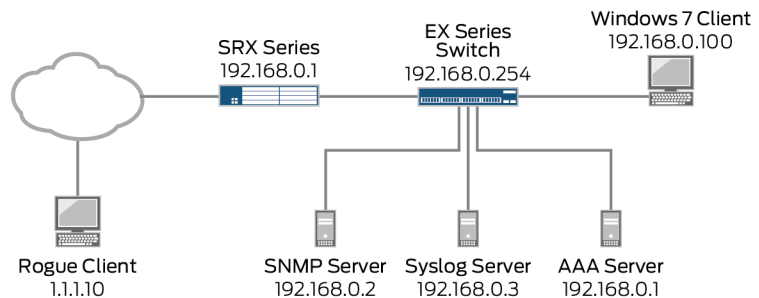


Figure 5.7 Rogue Client in Relation to the LAN

In Junos OS, the `show system users` command allows network engineers to see exactly who is logged in to the same device and from where:

```
networkreadonly@SRX> show system users
12:38PM up 3:36, 3 users, load averages: 0.05, 0.10, 0.09
USER      TTY      FROM              LOGIN@   IDLE   WHAT
networkadmin p1    192.168.0.3      11:14AM   -   -cli (cli)
networkreadonly p2 192.168.0.100    12:25PM   -   -cli (cli)
networkadmin p3    1.1.1.10         12:33PM   -   -cli (cli)
```

You can clearly see that there are three users logged in: one is logged in from our Syslog server, the second is logged in from our Windows 7 client, and the third is logged in from our rogue client. You need to perform four tasks:

- Change the password for the user *networkadmin*.
- Clear the rogue users SSH session.
- Undo the changes the user has applied to the firewall to allow him access.
- Configure Junos OS to only allow access to SSH and HTTPS from the subnet 192.168.0.0/24.

To reset the password, you need to either open the “Active Directory users and computers” tool within Windows to reset the password, or log in to a Windows computer on the domain and press Ctrl-Alt-Delete and select change password. If it were the case that the username was local, then the password can be changed using the following command in the configuration mode. In this instance, the password is changed to *Password2*:

```
set system login user networkadmin authentication plain-text-password Password2
```

Now that the password has been changed, you need to clear the user session. Run the following command:

```
request system logout terminal p3
```

P3 is the terminal the user is connected to, which is listed under the column TTY in the `show system users` command shown previously. Now, run the `show system users` again and it should show the user as having been disconnected:

```
networkadmin@SRX> show system users
12:53PM up 3:50, 2 users, load averages: 0.06, 0.04, 0.05
USER      TTY      FROM              LOGIN@   IDLE   WHAT
networkadmin p1    192.168.0.3      11:14AM   -   -cli (cli)
networkreadonly p2 192.168.0.100    12:25PM   2   -cli (cli)
```

For Step 3, you could have used the rollback option, or restored from an archive, both being procedures not covered here. But, let's drill down into procedure in Step 4 and how to create a firewall filter and assign this filter to a loopback interface.

## Configuring Device Management Restrictions

To prevent unauthorized personnel from trying to manage a firewall, a firewall filter can be applied to the control plane of the firewall. For those who aren't aware of these, firewall filters are a set of rules which contain terms. The terms within a given filter are processed in list fashion from top to bottom. The terms are objects within the individual filters and in the following configuration examples, the references to filters are actually referring to terms, not the entire filter. Whatever term matches a received packet is first used to decide what to do with the packet.

On this basis, you need to define what is allowed before you define what to reject. In this scenario, the filter is named *RESTRICTACCESS*. The first term that is created will have the label of *ALLOWED*. The following commands are set to match the source address from our own subnet, with the TCP protocol, and with the destination ports of SSH and HTTPS. The last set statement specifies that any packet matching the above should be accepted:

```
set firewall family inet filter RESTRICTACCESS term ALLOWED from source-  
address 192.168.0.0/24  
set firewall family inet filter RESTRICTACCESS term ALLOWED from protocol tcp  
set firewall family inet filter RESTRICTACCESS term ALLOWED from destination-port ssh  
set firewall family inet filter RESTRICTACCESS term ALLOWED from destination-  
port https  
set firewall family inet filter RESTRICTACCESS term ALLOWED then accept
```

Next a term called *NOTALLOWED* is created. This is set to match the protocol of TCP and the destination ports of SSH, Telnet, HTTP, and HTTPS. Note there is no source address specified, therefore, this term will match every address. The last three statements tell the device count the number of packets matching this term in a counter called *BAD-LOGIN*, reject the packet and then log it to the firewall log:

```
set firewall family inet filter RESTRICTACCESS term NOTALLOWED from protocol tcp  
set firewall family inet filter RESTRICTACCESS term NOTALLOWED from destination-  
port ssh  
set firewall family inet filter RESTRICTACCESS term NOTALLOWED from destination-  
port telnet  
set firewall family inet filter RESTRICTACCESS term NOTALLOWED from destination-  
port http  
set firewall family inet filter RESTRICTACCESS term NOTALLOWED from destination-
```

```

port https
set firewall family inet filter RESTRICTACCESS term NOTALLOWED then count BAD-LOGIN
set firewall family inet filter RESTRICTACCESS term NOTALLOWED then reject
set firewall family inet filter RESTRICTACCESS term NOTALLOWED then log

```

The next statement in this term is very important as terms have an implicit deny, meaning any packet that doesn't match the statement is denied automatically. Normally, in a firewall, this would be the desired result, but in this case if you don't specify a statement to accept everything else then the device prevents all packets from reaching the control plane, which in turn would mean SNMP, AAA, and routing protocols would all stop working:

```

set firewall family inet filter RESTRICTACCESS term EVERYTHINGELSE then accept

```

Finally, the filter needs to be assigned to an interface. If the filter was assigned to a VLAN interface or a physical port, then this would block not just traffic to the device, but also traffic that is destined for devices and servers beyond the device you are trying to protect. To resolve this, when Juniper designed Junos OS they set aside the interface lo0.0 or Loopback 0.0 as a virtual port that has a special relationship with the control plane of the device. By applying the filter to this interface it will affect only management traffic going to the device itself without affecting traffic going from say, clients to servers. The following command applies the filter *RESTRICTACCESS* to that interface:

```

set interfaces lo0 unit 0 family inet filter input RESTRICTACCESS

```

Before committing, it's important to take a step back and think about the consequences of what this change is about to do and that is, restrict access. If a mistake is made, then it's possible that an administrator could lock themselves out of the device. Thankfully, Juniper has already considered this scenario and as such included the `commit confirmed` command option. So if you don't enter the `commit` command again within 10 minutes, then the configuration is rolled back, thus reversing all the changes you have just made. Ten minutes is the default, but different times can be specified as well. In this case you don't need that much time to confirm, so therefore 5 minutes is specified:

```

commit confirmed 5

```

Once you have confirmed, you can access the device by using a separate SSH session. The second `commit` should be issued, which prevents the automatic rollback of `commit confirmed` command.

## Testing Device Management Restrictions

It should be obvious that to test whether the `RESTRICTACCESS` firewall filter is properly applied, an attempt to access the device from outside the subnet should be made, and if you cannot connect, but can still connect from inside the subnet, then all is working well.

In addition to testing access, you should test whether the *log* option you specified does indeed log failed attempts. To view these attempts, use the `show firewall log` command. In the example below, three attempts were made by the client 1.1.1.10 to the device. In the Action column, the *R* indicates the packet was rejected and under the Filter column, *pfe* indicates that the packet forwarding engine, or control plane, was where the filter that rejected the packet was applied:

```
networkadmin@SRX> show firewall log
```

Log :

Time	Filter	Action	Interface	Protocol	Src Addr	Dest Addr
14:43:48	pfe	R	fe-			
0/0/0.0	TCP		1.1.1.10	192.168.0.1		
14:43:42	pfe	R	fe-			
0/0/0.0	TCP		1.1.1.10	192.168.0.1		
14:43:39	pfe	R	fe-			
0/0/0.0	TCP		1.1.1.10	192.168.0.1		

Finally, by using the `show firewall filter` command followed by the name of the filter, the count of how many packets were rejected can be displayed:

```
networkadmin@SRX> show firewall filter RESTRICTACCESS
```

Filter: RESTRICTACCESS

Counters:

Name	Bytes	Packets
BAD-LOGIN	192	3

## Summary

This last finishing touch saves all your other hard work from being compromised. You should have a safe, connected, authorized, and time-sensitive device that can work within your network and be available to all the administrators that have access rights to it.

**NOTE** The purpose of this chapter is to begin your understanding of how to restrict access to the device. For a more comprehensive discussion on making your device more secure see *This Week: Hardening Junos Devices* at <http://www.juniper.net/ru/ru/training/jnbooks/day-one/fundamentals-series/hardening-junos-devices-checklist/>.

# Day One: Deploying Junos Finishing Touches

## Pre-Deployment Checklist



NTP			
Device IP added to Syslog Server	<input type="checkbox"/> None	<input type="checkbox"/> MDS	
Redundant NTP	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Delivery	<input type="checkbox"/> Unicast	<input type="checkbox"/> Multicast	<input type="checkbox"/> Broadcast
Junos NTP Server	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
[show ntp associations]	<input type="checkbox"/> Output as expected		

SNMP			
Version	<input type="checkbox"/> 1	<input type="checkbox"/> 2c	<input type="checkbox"/> 3
Community	<input type="checkbox"/> Read Only	<input type="checkbox"/> Read-Write	<input type="checkbox"/> N/A
Authentication	<input type="checkbox"/> None/NA	<input type="checkbox"/> MDS	<input type="checkbox"/> SHA
Privacy	<input type="checkbox"/> None/NA	<input type="checkbox"/> AES	<input type="checkbox"/> 3DES
[show ntp statistics]	<input type="checkbox"/> Output as expected		

Syslog			
Device IP added to Syslog Server	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Source address set in Junos	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Logging level	<input type="checkbox"/> Any	<input type="checkbox"/> "Tuned"	
Facilities	<input type="checkbox"/> Any	<input type="checkbox"/> "Tuned"	
Year/Millisecond specified	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Syslog server receiving logs with correct level and facility	<input type="checkbox"/>		

AAA			
Root password set	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Second superuser created	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Classes linked to VendorID 1	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
AAA Server	<input type="checkbox"/> RADIUS	<input type="checkbox"/> TACACS+	
AAA Accounting enabled	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
Test logins work with correct permissions	<input type="checkbox"/>		
Transactions logged with AAA server	<input type="checkbox"/>		

Securing Device Management			
Telnet	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
http	<input type="checkbox"/> Yes	<input type="checkbox"/> No	
SSH Version	<input type="checkbox"/> 1	<input type="checkbox"/> 2	
Firewall Filter	<input type="checkbox"/> Local Subnet	<input type="checkbox"/> http	<input type="checkbox"/> telnet
[Show firewall log]	<input type="checkbox"/> Output as expected		
[show firewall filter <filter name>]	<input type="checkbox"/> Output as expected		