


DAY ONE: USING JSNAP TO AUTOMATE NETWORK VERIFICATIONS

An abstract background composed of various overlapping triangles in different shades of orange, creating a complex geometric pattern.

Building High-IQ Networks means automating your network administration. Start scripting with JSNAP and verify your network's cutovers in minutes instead of hours.

By Diogo Montagner

DAY ONE: USING JSNAP TO AUTOMATE NETWORK VERIFICATIONS

Network engineers are constantly involved in planning and executing network changes and they are always concerned about the state of the network *after* the changes have been applied. The truth is, no one wants to go home and receive a call from the NOC saying there is a problem with the network, especially in the area where your changes were applied.

In order to reduce the risks of getting into an unpleasant situation after a change, many engineers have developed procedures and tools to verify their networks. The good news is that there is JSNAP – an automation tool that details pre- and post-verifications. JSNAP is a collection of SLAX scripts that runs on top of *jvise*, the environment that runs SLAX scripts off-the-box.

From setup, to sample scripts, to complete SLAX configurations, this *Day One* has it all – and you can put what you've learned to use in a matter of hours.

“WOW! This is an impressive document. Personally I think it goes beyond a “Day One” given the material and coverage. I am very, very impressed with the content and coverage.”

Jeremy Schulman, Director
Automation Concept Engineering, Juniper Networks

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

- Deploy automation for the network verification process.
- Understand the how to automate the verification process using JSNAP.
- Improve the network verification process by being more assertive during pre- and post-network verifications of a network change procedure.
- Create automated network verification tests.
- Use JSNAP in a snap.

Juniper Networks Books are singularly focused on network productivity and efficiency. Peruse the complete library at www.juniper.net/books.

Published by Juniper Networks Books



JUNIPER
NETWORKS

Day One: Using JSNAP to Automate Network Verifications

By Diogo Montagner

Chapter 1: Automating Network Verifications 9

Chapter 2: JSNAP Components 17

Chapter 3: Developing Automated Network Verifications 29

Chapter 4: Tips and Tricks 75

Chapter 5: Putting It All Together 93

Appendix 127

Building High-IQ Networks means automating your network administration. Start scripting with JSNAP and verify your network’s cutovers in minutes instead of hours.

© 2014 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: Diogo Montagner
Technical Reviewers: Jeremy Schulman, D.S.Satya Narsinga Rao
Editor in Chief: Patrick Ames
Copyeditor and Proofer: Nancy Koerbel
J-Net Community Manager: Julie Wider

ISBN: 978-1-936779-91-8 (print)
Printed in the USA by Vervante Corporation.

ISBN: 978-1-936779-92-5 (ebook)

Version History: v1, May 2014
2 3 4 5 6 7 8 9 10

About the Author

Diogo Montagner (JNCIE #1050 and PMP #1616862) holds a Bachelor's Degree in Computer Science from Universidade Federal de Santa Maria (UFSM) and MBA in Project Management from Fundação Getulio Vargas (FGV). He has been working in the Juniper Advanced Services Team since 2008.

Author's Acknowledgments

I would like to first thank my wife Raiça and my daughter Linda, who supported me throughout the many early mornings and few weekends that it took me to conclude this four-month-long project. Patrick Ames for his thorough developmental edit and all his efforts to make this project a reality. Antonio (Ato) Sanchez-Monge, Anton Bernal, and Gary Matthews who helped, guided, and mentored me on the path to publishing my first book. Jeremy Schulman, and D.S. Satya Narsinga Rao for allocating time in their busy schedules to do the technical review of the book and for all the technical help provided during its development. Damien Garros for his help with a few XPath expressions. Last but not least, Antonio (Ato) Sanchez-Monge and Geoffrey Younger for being my beta readers.

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) or Amazon (amazon.com) for between \$12-\$28, depending on page length.
- Note that Nook, iPad, and various Android apps can also view PDF files.
- If your device or ebook app uses .epub files, but isn't an Apple product, open iTunes and download the .epub file from the iTunes Store. You can now drag and drop the file out of iTunes onto your desktop and sync with your .epub device.

Audience

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

This book's chapters are numbered in a logical sequence to identify, plan, develop, and execute automated network verifications for network changes affecting the entire network.

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.

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

- You are a network engineer who is familiar with network protocols.
- You may or may not have programming knowledge.
- You may or may not understand the business impact of a network change.
- You want to automate your network verification process.
- You are responsible for planning or executing network changes.
- You are responsible for network monitoring and proactive verifications.

What You Will Learn by Reading This Book

- Understand the impact of a network change.
- Deploy automation for the network verification process.
- Understand the how to automate the verification process using JSNAP.
- Improve the network verification process by being more assertive during pre- and post- network verifications of a network change procedure.

- Create automated network verification tests.
- Use JSNAP in a snap.

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). Be sure to check them out, too.

MORE? This book was developed for people with minimal or zero knowledge in programming languages and XML. However, knowing a little bit of both will help speed up the development of network verifications using JSNAP. The following resources are a good starting point.

- XML and XPath online tutorials:
 - XML: <http://www.w3schools.com/xml/default.asp>
 - XPath: <http://www.w3schools.com/xpath/default.asp>
- SLAX Reference
 - *This Week: Junos Automation Reference for SLAX 1.0*, available at <http://www.juniper.net/dayone>

If you have feedback for Juniper Networks about this book, please send it to dayone@juniper.net.

Preface

Network engineers are constantly involved in planning and executing network changes. From the most basic to the most complex changes, network engineers are always concerned about the state of the network after the changes have been applied. Questions like, “Did I break something?” and, “Were our changes successfully deployed?” are always worried about after planned activities are completed.

In order to reduce the risks of getting into an unpleasant situation after a change, many engineers have developed procedures and tools to verify their networks. The truth is, no one wants to go home and

receive a call from the NOC saying there is a problem with the network, especially in the area where your changes were applied.

Throughout my network career, I have not only seen different tools and procedures for network verification, I have developed my own tools as well. This book will introduce you to a tool called JSNAP that can make your life much easier. Even if you already have tools and procedures in place, take the *Day One* tour. I am sure you will find it useful, just as I did when I was introduced to JSNAP.

Diogo Montagner, May 2014

Chapter 1

Automating Network Verifications

The pre- and post-verifications executed before and after a network change are important steps that must not be skipped. When preparing a network change, make sure you always include the pre- and post-verifications in the plan. And whenever possible, automate those verifications.

Most network engineers agree on the importance of pre- and post-verifications, however, there may be different opinions on whether they should be automated or not. Let's look at an example you might be familiar with:

"The network engineer started the pre-verification at 11:45 p.m. The verification procedure is quite long and took about 30 minutes to collect all commands on all devices that were affected by the change. Around 12:15 a.m. the engineer started to implement the network change, which was not a big change but had to be applied on many devices. After working for three consecutive hours, the engineer finished the change at 3:30 a.m. when he started the post-verification. By 4:00 a.m. he finished the collection of the commands and started to compare output by output. By now he was tired because he was awake for so many hours and decided to cut short the comparison after he found that a few of the devices he compared did not show any problem and he believed the rest of changes were successfully deployed the same as the ones he just checked. He skipped some of the comparisons to shorten the post-verification process and around 5:00 a.m. he completed the comparisons and moved to close the change, declaring it a success."

Despite the fact that the fictitious case presented here does not demonstrate whether there was a problem with the cut over, it does demonstrate two common problems of network changes: *bad planning and lack of automation with repetitive tasks*.

Without getting into too much detail, a better plan would have helped to avoid the risk that the engineer took when he decided to skip some verification steps. That plan would have identified that the verification steps were too long for a single engineer to execute, and that added resources were needed to run the verification process. Whether the resources needed were extra manpower, or automation tools, the verification process would not have taken that long to execute and the risk could have been avoided. Sound familiar?

Network automation is here to stay, and believe it or not, the verification process is one of the easiest processes to automate, especially when using a tool like JSNAP. But before jumping into how to use JSNAP, let's have a quick look at the Change Document and at the Network Change Process.

The Change Document

Generally speaking, whenever a network is undergoing a planned change, there must exist a document where these changes are documented. This document has different names in different organizations. Someone may call it MOP (Method of Procedures), while others may simply call it the Plan. No matter what you call it, always make sure you have this document prepared before you start the changes because the MOP is the document that presents the overview of the change, the objectives, the change procedure itself (step-by-step), the pre- and post-verifications, and, last but not least, the roll back procedure.

This *Day One* book focuses solely on the pre- and post-verifications because that is where JSNAP can automate your network verification process.

The Network Change Process

Let's have a look in the overall change process so you have a baseline for using JSNAP. Figure 1.1 presents an example of a change process.

Everything starts with MOP development. This is where you plan the changes, assess the risk and the impact, and prepare the verification procedures as well as the rollback procedures. Once all these items are packed in a single document (the MOP), you submit a change request and wait for approval.

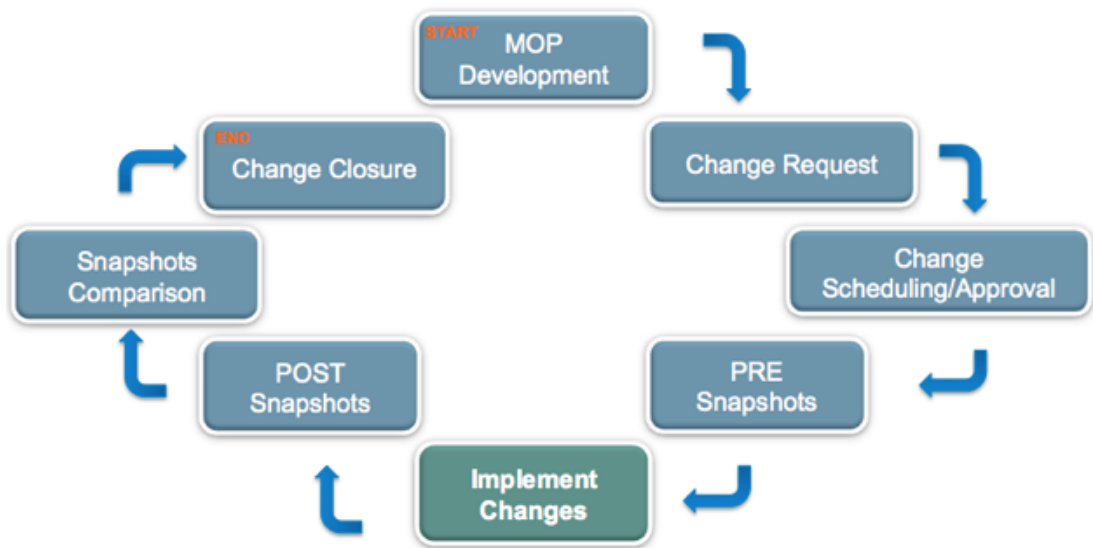


Figure1.1 Example of a Change Process

On the day of the change, you will first execute the pre-snapshots, and after that, you can apply the changes to the network. As soon as you finish the changes, you collect the post-snapshots. (Some network changes may require a waiting period before you start to collect the post-snapshots in order to allow the churn caused by changes to complete, for example, changes to an Internet BGP session that receives a full routing table from the remote side.)

Once you have both pre- and post-snapshots in hand, it's time to compare them. If the comparison shows that everything is in the same state, and this is what you were expecting as result of your changes, you can move on to the change closure. If the comparison identifies a problem, you must check what you have defined in the MOP for such situations. Should you try to fix the issue or should you rollback the change? That will depend on how the change was planned.

NOTE There is a small variant of the Network Change Process where there is no previous collection of snapshots that can also be automated with JSNAP. In that case, the snapshots are captured after the change and can have its compliance checked against a group of criteria. This approach is a bit more risky and should be used only in cases where it is impossible to compare pre- and post-snapshots. An example of where this approach fits is migrating the IGP of a network from OSPF

to ISIS because you can't compare OSPF snapshots (pre) with ISIS snapshots (post). This book focus solely on network changes cases where the pre- and post-snapshots can be compared against each other.

The closure of the change is an important step that is quite often ignored by network engineers. In this step, information about the duration of the change, the timeline of the change, what exactly happened during the change, and notification of stakeholders are performed and documented. This ensures that everyone affected by the change is on the same page in regard to what has happened in the change. Moreover, it stores the lessons learned from this change, which are an invaluable asset for planning future changes.

Okay, these are pretty basic best practices, so let's start to focus on the Network Verification process where this book concentrates.

The Network Verification Process

The pre- and post-verification process is usually a three-step process. The first and second steps are the same, yet executed at two different moments in time. The third step is where you analyze the data from the first and second steps and it is executed immediately after the second step:

1. Collect the pre-snapshot before executing any changes.
2. Collect the post-snapshot after the changes have been executed.
3. Compare the pre- and post-snapshots searching for differences.

Network engineers execute this three-step process in different ways. One may run it manually, command-by-command, while saving the CLI output in a text file. Another may use a script to automatically collect the output of the command and save it to a file. Our focus here is in the automated way.

There are different techniques that can be used to automate networks, and in each of them there are different ways to implement the interaction with routers. When interacting with routers running Junos, you can use Expect, Netconf, or SNMP.

Expect is a technique that relies on matching regular expressions to identify when the router is ready to receive commands and when the execution of a command has finished, but it is not the most reliable way to collect commands from routers because it is very easy to get incomplete collections. Netconf, on the other hand, when combined with RPC calls, can give you the most reliable way of collecting commands from routers.

Starting to sound complicated? Yes, sort of, but the good news is that when using JSNAP you don't need to worry about these details because the outputs are always correctly collected. JSNAP leverages the power of XML and XPath. While this technique may be new to you, it brings a concise, simple, and powerful way of accessing the information present in the output generated by the router. For instance, instead of writing a complex parser code to extract a few words of information from the router's output, a simple XPath query can be used to obtain the same information.

If you have ever found the automation of Steps 1 and 2 a problem, wait until you try to automate Step 3, because it is not straightforward to automate. Most engineers automate only the first and second steps, and then use a tool to compare the text outputs generated. If you are in this situation, you are actually in a good place because you already have some sort of automation in place and believe that automation is the way to go. If you are still on manual mode, then.... well, at least you are reading the right book. :-)

When only the first and second steps are automated, there are a lot of different comparison tools used by most engineers. One favorite, *fldiff*, comes with most *Linux* distributions. This book uses Apple's *FileMerge* tool that comes in the *XCode* package, and you'll see screen captures from that program.

In order to demonstrate how the third step works when using a text comparison tool, let's consider the change in the routers:

```
lab@carbon> show configuration | compare rollback 1
[edit interfaces]
+ ge-2/1/0 {
+   description "Connected to PE1";
+   disable;
+ }
lab@carbon>
```

Assuming you have collected a pre-snapshot before applying the change and a post-snapshot after the change has been applied, Figure 1.2 shows the output of a comparison between the pre-and the post-snapshots using Apple's *FileMerge* tool.

before.txt vs. after.txt				
before.txt - /Users/dmontagner/Documents/Diogo/JNPR/docs-jnpr/vBook,				
after.txt - /Users/dmontagner/Documents/Diogo/JNPR/docs-jnpr/vBook				
lab@carbon> show interfaces terse no-more				
Interface	Admin	Link	Proto	Local
ge-2/0/0	up	down		
gr-2/0/0	up	up		
gr-2/0/0.32769	up	up	inet	
			mpls	
ip-2/0/0	up	up		
lc-2/0/0	up	up		
lc-2/0/0.32769	up	up	vpls	
lt-2/0/0	up	up		
lt-2/0/0.0	up	up	inet	
lt-2/0/0.1	up	up	inet	
mt-2/0/0	up	up		
pd-2/0/0	up	up		
pe-2/0/0	up	up		
pfe-2/0/0	up	up		
pfe-2/0/0.16383	up	up	inet	
			inet6	
pfh-2/0/0	up	up		
pfh-2/0/0.16383	up	up	inet	
ut-2/0/0	up	up		
vt-2/0/0	up	up		
ge-2/0/1	up	up		
ge-2/0/2	up	down		
ge-2/0/3	up	down		
ge-2/0/4	up	down		
ge-2/0/5	up	down		
ge-2/0/6	up	down		
ge-2/0/7	up	down		
ge-2/0/8	up	down		
ge-2/0/9	up	down		
ge-2/1/0	up	up		
ge-2/1/1	up	down		
ge-2/1/2	up	down		
ge-2/1/3	up	down		
ge-2/1/4	up	down		
ge-2/1/5	up	down		
ge-2/1/6	up	down		
ge-2/1/7	up	down		
ge-2/1/8	up	down		
ge-2/1/9	up	down		
lc-2/2/0	up	up		
lc-2/2/0.32769	up	up	vpls	
pfe-2/2/0	up	up		

status: 2 differences

Actions

Figure 1.2 Text Comparison Tool Output

You can see after the application of the change in the router that there are two differences in the comparison. Maybe you were expecting to see only one difference: the interface ge-2/1/0 in a down state. However, the interface ge-2/0/1 also went to a down state. It definitely wasn't caused by the configuration change that was deployed to the router, but by something else that the network engineer responsible for the change needs to verify.

The comparison presented in Figure 1.2 is a simple and easy example because the change was simple and the output is clean, meaning there is only one command output in the text file. Now, let's check how the comparison tool behaves with outputs of many commands, in this case, the RSI (output from the **request support information** command) before and after the change. Figure 1.3 presents the comparison of both the pre- and post-RSI outputs.

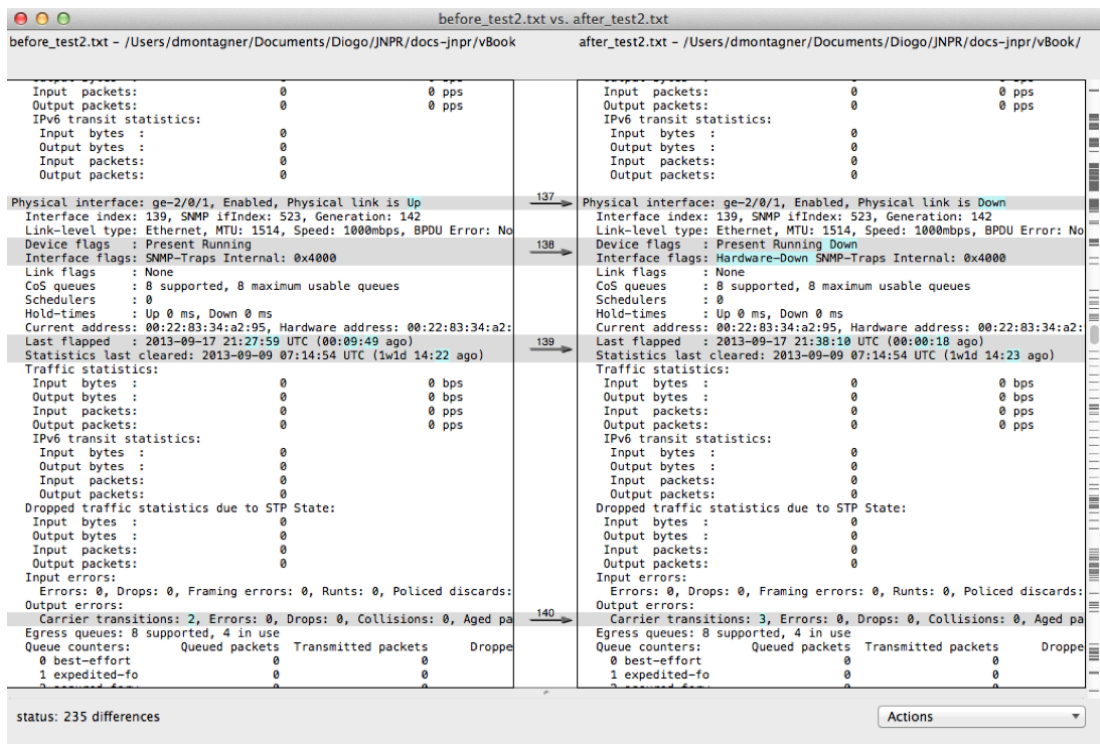


Figure 1.3 Pre- and Post-Comparison of RSI Outputs

According to information presented by FileMerge, Figure 1.3 shows that there are now 235 differences in the comparison. But you can't just trust the number of differences presented by the tool because it forces you to manually go through the entire comparison results to make sure that two out of the 235 have legitimate differences. If you think that's confusing, wait until you see the next example.

In the next use case the huge amount of differences are not the only problem. Figure 1.4 presents the same scenario as the previous example but for some unexplained reason, the outputs of a few commands are missing.

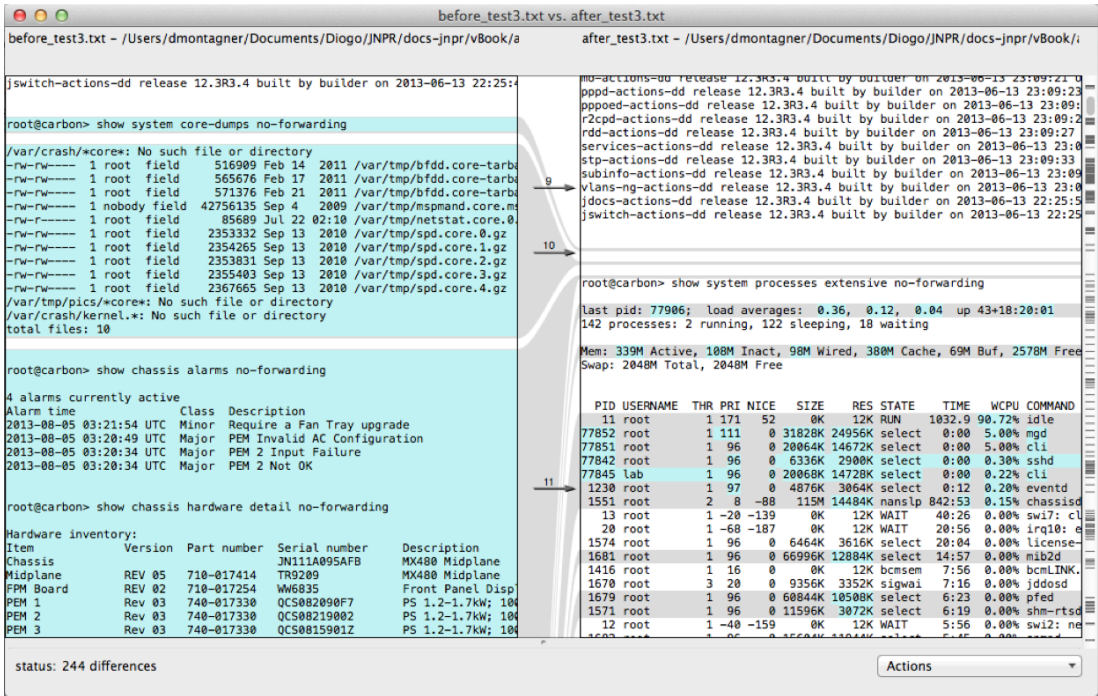


Figure 1.4 Missing Command Outputs in the Comparison

Missing commands are always a big issue but especially if the missing outputs are located in the pre-outputs (if the missing outputs are located in the post-outputs you still have a chance to collect them). The problem here is that during the comparison of the snapshots (i.e., *after* the change) you don't have the baseline of the state of the network prior to your change, and depending on the type of your change, it will be very difficult or it will take extra time to verify if everything went well.

In the next chapters, you will learn how to expedite network verification by developing automated verification procedures using JSNAP.

Chapter 2

JSNAP Components

To be succinct, JSNAP is a collection of SLAX scripts that runs on top of *juise*. For better understanding, this chapter will divide JSNAP into four components: SLAX, *juise*, JSNAP Core, and the Configuration File.

SLAX

The SLAX language was originally developed as part of Juniper Networks Junos OS to simplify manipulation of XML outputs produced by Junos. Before SLAX, the on-box script programming had to be done using XSLT syntax. The XSLT syntax is complex and requires a lot of typing to execute simple operations, hence SLAX.

SLAX is nothing more than an alternate syntax for XSLT. In fact, SLAX stands for *Stylesheet Language Alternative syntaX* while XSLT stands for “*eXtensible Stylesheet Language Transformation*”. The XSLT is the W3C’s (World Wide Web Consortium: <http://www.w3.org/Consortium/>) standard for XML-to-XML transformation. Here’s a simple code written in XSLT:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <xsl:variable name="VPNImportPolicies" select="/rpc-reply/configuration/
routing-instances/instance/vrf-import"/>
    <xsl:for-each select="$VPNImportPolicies">
      <xsl:variable name="tmp" select="."/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

It is not exactly straightforward, is it? Fortunately, SLAX can make a network administrator's life easier. Here's the simple code now written in SLAX:

```
version 1.1;

match / {

    var $VPNImportPolicies = /rpc-reply/configuration/routing-instances/
instance/vrf-import;

    for-each ($VPNImportPolicies) {
        var $tmp = .;
    }

}
```

You may not be able to judge if your life will be easier right now but stay with this book and it becomes quickly evident. What's cool is that because SLAX is just another syntax for XSLT, it is possible to use XSLT code inside SLAX. In fact, SLAX functions as a preprocessor for XSLT. Figure 2.1 illustrates what happens inside a Junos router when a SLAX script is used as input.

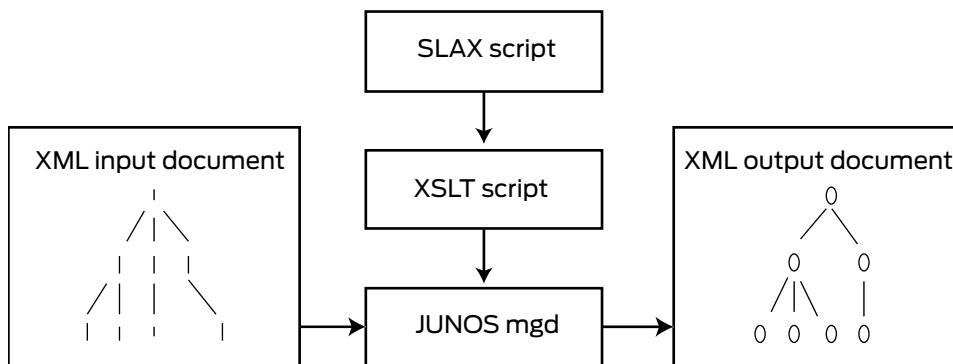


Figure 2.1 SLAX to XSLT Conversion in JUNOS

You can see there is a conversion going on. For now, that's all you need to know about SLAX. In fact, you don't even need to know SLAX to use JSNAP.

MORE? If you want to learn more about SLAX, look in the following *Day One* books: *Junos Automation Reference for SLAX 1.0*, *Mastering Junos Automation*, and *Applying Junos Automation*, all of which can be found at <http://www.juniper.net/dayone>.

The Junos User Interface Script Environment

What may sound to you like a refreshing drink is actually the environment needed to run JSNAP. The Junos User Interface Script Environment, or *juise*, is an environment developed by Phil Shafer at Juniper to run SLAX scripts off-the-box. Not only can you run SLAX scripts, *juise* also helps when developing SLAX scripts, providing a off-box way to test and debug them. Because JSNAP was developed using SLAX and runs off-the-box, you need *juise* in order to run JSNAP. Recent versions of Junos, such as 13.2, allow you to invoke the *slax* debugger (*sdb*) to troubleshoot your on-box SLAX scripts.

MORE? For more on invoking the *slax* debugger (*sdb*) see: http://junos.com/techpubs/en_US/junos13.2/topics/reference/command-summary/op-invoke-debugger-cli.html.

Another function provided by *juise* to JSNAP is the ability to invoke *netconf* environment. *Netconf* is the method used by *juise* to connect to Junos routers in order to send RPC commands and receive their respective replies. *Netconf* runs on top of SSH, which means that all communications between *juise* and a router are flowing through an encrypted channel.

Netconf Configuration

Here is the configuration required to enable *Netconf* support under SSH on Junos routers. Try it in your lab's sandbox:

```
[edit system]
services {
  ssh;
  netconf {
    ssh;
  }
}
```

If you are experiencing a problem connecting to Junos routers via *netconf*, you can use the following steps to isolate where the problem is:

1. Verify if you can access the router via SSH from the server where JSNAP is installed.
 - The most common problems seen in Step 1 are related to the exchange of SSH keys. Make sure you have configured the SSH client of the server where JSNAP is installed to automatically answer 'yes' when exchanging SSH keys. If not, you may need to answer 'yes' when *juise* attempts to connect to a router that does not have its key generated in the *known_hosts* file. You can read more about this in Chapter 4 of this book.

2. Try to manually establish a *netconf* session with the router.

- Problems seen in Step 2 are usually related to configuration issues in the Junos router. The procedure to manually establish a *netconf* connection and send RPC commands to the router is displayed here:

```
ssh lab@zim netconf
```

Finally, an example of an interaction with a Junos router using a *netconf* session is similar to the following:

```
dmontagner@querencia:~/jsnap$ ssh lab@zim netconf
lab@zim's password:
<!-- No zombies were killed during the creation of this user interface -->
<!-- user lab, class j-super-user -->
<hello>
  <capabilities>
    <capability>urn:ietf:params:xml:ns:netconf:base:1.0</capability>
    <capability>urn:ietf:params:xml:ns:netconf:capability:candidate:1.0</capability>
    <capability>urn:ietf:params:xml:ns:netconf:capability:confirmed-commit:1.0</
capability>
    <capability>urn:ietf:params:xml:ns:netconf:capability:validate:1.0</capability>
    <capability>urn:ietf:params:xml:ns:netconf:capability:url:1.0?protocol=http,ftp,f
ile</capability>
    <capability>http://xml.juniper.net/netconf/junos/1.0</capability>
    <capability>http://xml.juniper.net/dmi/system/1.0</capability>
  </capabilities>
  <session-id>6227</session-id>
</hello>
]]>]]>

<rpc><command>show version</command></rpc>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:junos="http://xml.
juniper.net/junos/11.4R7/junos">
  <software-information>
    <host-name>zim_re0</host-name>
    <product-model>m320</product-model>
    <product-name>m320</product-name>

  <... Omitted for brevity ...>
```

Downloading *Juise*

Juise is free and can be downloaded from GitHub (<https://github.com/Juniper/juise/>). When you download the JSNAP software from Juniper, *juise* comes packed with JSNAP and does not require a special installation for it.

One of the most important features of *juise* is the debugger. It will help you when creating complex JSNAP configuration files or when trou-

bleshooting problems with your JSNAP configuration files. The debugger requires a little bit of extra knowledge in SLAX but this book should make it easy for you.

Make sure you have your lab set up with *Netconf* and *juise*. You'll get back to them shortly. In the next two sections, you'll first see how JSNAP works and then you'll eventually start automation tests.

The JSNAP Core

The JSNAP core is a group of SLAX scripts that execute the three tasks that JSNAP was designed for: collecting, comparing, and evaluating snapshots. At the time of this writing, the current available version for download at Juniper Web Site is 1.0 sub-version 6 (1.0.6) as you can see here:

```
dmontagner@querencia:~$ jsnap --version
```

```
jsnap: ver: 1.0, 2013-JUN-13
```

```
e3d6994b82f1b4c2214fc438486ca06bf2b4dd82: jsnap/jsnap
2f1ae309464688152e8d108acf413eca91cc8d69: jsnap/jppc-exec.slax
c17dc9384c58bf7d8258f07f1a43874f09b351a4: jsnap/jppc-snap.slax
36ac8351b8bb6e292e0faa2b6f73d23465a37cca: jsnap/jppc-tests.slax
ab54be6ff257858d94cff054a406baf7f462d827: jsnap/jppc-utils.slax
401faf121d99f95fcc675192f6d2f656be11b012: lib/libjfile.slax
0c2c4680ba28ab915c4c85e377b05cc6dc25c50f: lib/libcbd.slax
c0852d740be86d60d9c56461625d7fdaac9d4a9a: lib/libxns.slax
e3d6994b82f1b4c2214fc438486ca06bf2b4dd82: bin/jsnap
```

The files listed in this output are the core files of JSNAP.

TIP

If you want to check all files including *juise* and SLAX files, issue the following command in your Linux box: **dpkg -L jsnap**. (**bin/jsnap** is a symbolic link to **jsnap/jsnap**.)

The file **bin/jsnap** is the main script of JSNAP. This is the script you must invoke to create snapshots, execute comparisons, and evaluate snapshots against a predefined set of criteria. It has built-in help that is displayed when JSNAP is invoked without any parameter or with wrong parameters, such as those shown here:

```
dmontagner@querencia:~$ jsnap
```

```
jsnap: snapshot data collection and validation
```

```
jsnap --snap <name> [lts] <conf-file>
    Snapshot data and save as collection 'name'
```

```
jsnap --check <name1>,<name2> [ts] <conf-file>
```

Check results of two snapshot collections 'name1' and 'name2'

```
jsnap --snapcheck <name> [lts] <conf-file>
```

Take a single snapshot as collection 'name' and checks results

NOTE: does not compare two collections

OPTIONS:

```
-l | --login <login>
-p | --passwd <passwd>
-t | --target <target>
-s | --section <section-name>
--version
```

When you're working with JSNAP, you'll notice that it does not give you an error message if you have entered a combination of wrong parameters. Therefore, here are two examples of invoking JSNAP, one for each operation.

1. The first example is a snapshot generation, such as this snapshot collection from the router *zim*:

```
dmontagner@querencia:~/jsnap$ jsnap --snap pre_mw -l lab -t zim -p lab123 mw.conf
Connecting to lab@zim ...
CONNECTED.
EXEC: 'show chassis routing-engine' ...
SAVE: 'zim__show_chassis_re__pre_mw.xml' ...
dmontagner@querencia:~/jsnap$
```

In this example, **pre_mw** is the name of the snapshot, **lab** is the username, **lab123** is the user password, and **mw.conf** is the JSNAP configuration file (which is discussed in the next section, *The JSNAP Configuration File*).

2. The second example is a comparison between two snapshots, and the following demonstrates the comparison between the **pre_mw** and **post_mw** snapshots for the router *zim*:

```
dmontagner@querencia:~/jsnap$ jsnap --check pre_mw,post_mw -t zim -l lab -p lab123 mw.conf
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: zim
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-----
CHECKING SECTION: show_chassis_re
-----
- TEST FAILED: "Checking Routing Engine mastership state."
```

The status of Routing Engine 0 has changed from backup to master

The status of Routing Engine 1 has changed from master to backup

```
dmontagner@querencia:~/jsnap$
```

And you can see in the results of the snapshot comparison where the mastership state of *zim*'s routing engines has changed during the maintenance window.

The snapshot evaluation is very similar to the snapshot comparison, the difference being that you run the tests only against a single snapshot. Test operators, which require two snapshots as input, cannot be used in the snapshot evaluation mode, but when using the snapshot comparison mode, all test operators can be used.

Now that you know how to operate JSNAP, let's learn about its configuration file.

The JSNAP Configuration File

The configuration file is where you define the commands that are used to generate the snapshots. It is also used to define the tests that will be used for comparing and evaluating snapshots.

The configuration file has a very simple structure. It contains two sections: the *Do Section* and the *Test Section*. Here is a very simple JSNAP configuration file:

```
do {
    show_chassis_re;
}

show_chassis_re {
    command show chassis routing-engine;
    iterate route-engine {
        id slot;
        no-diff mastership-state {
            info "Checking Routing Engine mastership state.";
            err "The status of Routing Engine %s has changed from %s to %s", $ID.1,
$PRE/mastership-state, $POST/mastership-state;
        }
    }
}

show_chassis_fpc {
    command show chassis fpc;
    iterate fpc {
        id slot;
        no-diff state {
            info "Checking the FPC state.";
            err "The state of FPC %s has changed from %s to %s.", $ID.1, $PRE/state,
$POST/state;
        }
    }
}
```

In the **do** section, you configure which test sections will be invoked if JSNAP is executed without specifying a section name. The separation

per section allows you to invoke JSNAP to create a snapshot of a single command, even if your configuration file has many sections defined.

In the configuration there are two test sections but only one is invoked in the **do** section. When JSNAP is invoked using this configuration file, only the **show_chassis_re** section will generate snapshots. The only way to generate snapshots for the **show_chassis_fpc** section, using this same configuration file, is invoking JSNAP using the **-s** option and passing the **show_chassis_fpc** section name as a parameter. This will instruct JSNAP to collect snapshots only for the **show_chassis_fpc** section, ignoring the other *Test* sections that are invoked in the *Do* section.

NOTE When defining names for sections, avoid using a whitespace character in the name. JSNAP will understand it as a delimiter and you may have undesirable results when you check the snapshots produced by JSNAP. You can use the underscore symbol in the section name as in: **show_chassis_fpc**.

Analyzing a Sample Configuration File

Now that you are more familiar with the configuration file, let's analyze the test section in detail. The test section has the structure presented here:

```
<TEST_SECTION_NAME> {
  command <JUNOS_COMMAND>;
  (item | iterate) XPATH {
    id <ID1>;
    <TEST_OPERATOR> <XML_ELEMENT> {
      info "<INFO_MESSAGE>";
      err "<ERROR_MESSAGE>";
    }
  }
}
```

NOTE As you can see, you are dealing with XML outputs. While having a basic XML knowledge does help, it is not mandatory. There are many, many XML tutorials available on the Internet, so if you need them, or want to use them as a refresher, do so now because this *Day One* book does not cover the basics of XML.

The name of the test section and the Junos command should be straightforward for you. Just remember that not all Junos commands will generate XML outputs.

NOTE Output modifiers are not supported in JSNAP. For example, **show interfaces | match MTU** cannot be used in JSNAP.

The **iterate XPATH** block can appear multiple times inside the same command. The same is true for the referenced **XPATH**. However, it is good practice grouping all tests over the same **XPATH** inside the same **iterate XPATH** block.

The **iterate** command is used when the **XPATH** refers to a *node-set* with multiple elements while the **item** can only reference one element.

The **id** is an **XPATH** expression relative to the *node-set* being iterated that specifies a unique data element that maps the first snapshot data item with the second snapshot data item. It is possible to use single or multiple IDs under the same **iterate** or **item** XPATH blocks.

NOTE One of the reasons that **id** was introduced in the JSNAP Configuration File was to simplify complex XPath expressions when using them in the **err** messages.

Finally, the test operators are used to instruct JSNAP how to analyse the XML element when comparing it between two snapshots or when using the checking mode. The next section of this chapter explains these operators in detail.

The JSNAP Configuration File – Test Operators

The JSNAP Test Operators are divided into five categories according to their applicability. This helps you to easily identify which operator you should use for a particular test case:

- Compare Elements or Element Values in Two Snapshots
- Operate on Elements with Numeric or String Values
- Operate on Elements with Numeric Values
- Operate on Elements with String Values
- Operate on XML Elements

MORE? For an example of a use case for each test operator, please refer to the Junos Snapshot Administrator documentation available at: http://www.juniper.net/techpubs/en_US/junos-snapshot1.0/topics/reference/general/automation-junos-snapshot-operators-summary.html.

Test Operators Used to Compare Elements or Element Values In Two Snapshots

This category has four test operators: delta, list-not-less, list-not-more, and no-diff.

Table 2.1 Compare Elements or Element Values in Two Snapshots

Operator	Description
delta	<p>Use: Compare the change in value of an element to a delta.</p> <p>Requirement: Must be present in both snapshots.</p> <p>The delta can be specified as:</p> <ul style="list-style-type: none">• an absolute percentage• a positive or negative percentage• an absolute fixed value• a positive or negative fixed value <p>Example of Use: Identify if the number of routes received on a BGP session has changed more than the specified delta.</p>
list-not-less	<p>Use: Check if the item is present in the first snapshot but not present in the second snapshot.</p> <p>Requirement: None.</p> <p>Example of Use: Check if interfaces were removed from the router.</p>
list-not-more	<p>Use: Check if the item is present in the second snapshot but not present in the first snapshot.</p> <p>Requirement: None.</p> <p>Example of Use: Check if new interfaces have been installed in the router.</p>
no-diff	<p>Use: Compare data elements present in both snapshots, and verify if their value is the same.</p> <p>Requirement: Must be present in both snapshots.</p> <p>Example of Use: Check if the operating state for all FPCs remains the same after the change has been implemented.</p>

Test Operators to Execute Tests Over Elements with Numeric or String Values

This category has three test operators: all-same, is-equal, and not-equal.

Table 2.2 Execute Tests Over Elements with Numeric or String Values

Operator	Description
all-same	Use: Check if all content values for the specified element are the same. It can also be used to compare all content values against another specified element. Requirement: None. Example of Use: Check if the OSPF interface priority are all the same or are all equal to a specific interface.
is-equal	Use: Check if the value (integer or string) of the specified element matches a given value. Requirement: None. Example of Use: Check if the Member 0 is the Master member on a Virtual Chassis.
not-equal	Use: Check if the value (integer or string) of the specified element does not match a given value. Requirement: None. Example of Use: Check if the Member 1 is NOT the Master member on a Virtual-Chassis.

Test Operators to Execute Tests Over Elements with Numeric Values

This category has four test operators: in-range, is-gt, is-lt, and not-range.

Table 2.3 Execute Tests Over Elements with Numeric Values

Operator	Description
in-range	Use: Check if the value of a specified element is in the given numeric range. Requirement: None. Example of Use: Check if the CPU Idle is within 20% ~ 99% range.
is-gt	Use: Check if the value of a specified element is greater than a given numeric value. Requirement: None. Example of Use: Check if the CPU Idle is greater than 20%.
is-lt	Use: Check if the value of a specified element is lesser than a given numeric value. Requirement: None. Example of Use: Check if the CPU Idle is less than 20%.
not-range	Use: Check if the value of a specified element is outside of a given numeric range. Requirement: None. Example of Use: Check if the CPU Idle is outside of 20% ~ 99% range.

Test Operators to Execute Tests Over Elements with String Values

This category has three test operators: contains, is-in, and not-in.

Table 2.4 Execute Tests Over Elements with String Values

Operator	Description
contains	Use: Check if the specified element string value contains a given string value. Requirement: None. Example of Use: Check if all routing-engines of a Virtual-Chassis are running a particular Junos version.
is-in	Use: Check if the specified element string value is included in a given list of strings. Requirement: None. Example of Use: Check if the BGP peer state is in Established state.
not-in	Use: Check if the specified element string value is NOT included in a given list of strings. Requirement: None. Example of Use: Check if the BGP peer state is NOT in Established state.

That’s everything you need to know in order to create the JSNAP configuration file, collect and compare snapshots, and start automating your network verification procedures. If you need to review, revisit this brief but to-the-point review of all the JSNAP components.

The next chapter guides you through the development of a set of automated tests for network verification.

Chapter 3

Developing Automated Network Verifications

At this point, you should readily understand the importance of having an automated network verification process during the deployment of any network changes. This chapter guides you in the process of developing a group of automated tests to be used in most (if not all) of your network changes.

The Network

The first thing you have to do before getting your hands dirty with JSNAP is to define the coverage of the automated tests, because frankly, not all networks are the same. For example, you may use OSPF as an IGP while another reader may use IS-IS, and so on.

So, let's define the network that the automated tests have to check for this chapter, by first defining a network topology and then the network architecture.

The topology to be used in this chapter to develop our set of automated network verification tests is illustrated in Figure 3.1.



Figure 3.1 Network Topology for Chapter 3

Our IGP will be ISIS, full-mesh RSVP, and MP-iBGP. The network service being delivered to the CE routers is MPLS VPN. Figure 3.2 shows the logical diagram for this chapter’s network architecture.

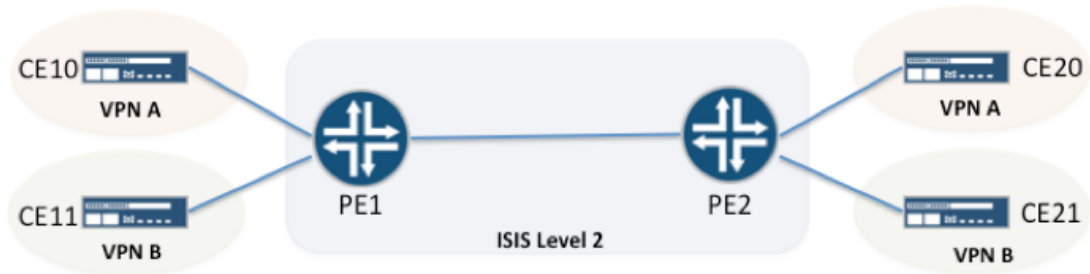


Figure 3.2 Logical Topology

Now let’s start to plan the tests.

Identifying The Network Areas

One of the most important things in project management is to not skip the planning process.

First, you should write down the logical components that need to be tested. A simplified version of the 5Ws and 2Hs approach (What, When, Where, Why, Who) (How, How Much) is used to help identify what tests need to be developed. In this case it’s 2Ws and 1H, or, *What, Where, and How*. These are the repeating columns in Tables 3.1 through 3.7 that list the components of each major category.

Table 3.1 IGP Components

	What	Where	How
IGP	ISIS Adjacency	PE1 and PE2	The status of ISIS adjacency must be the same before and after the change.
	ISIS Interfaces	PE1 and PE2	There should be no change in the IGP topology before and after the change.

Table 3.2 Interface Components

	What	Where	How
Interfaces	Operating Status	PE1 and PE2	The operating status of the GE interfaces must remain the same before and after the change.
	Admin Status	PE1 and PE2	The admin status of the GE interfaces must remain the same before and after the change.
	Total Interfaces	PE1 and PE2	The number of GE interfaces in the router must remain the same after the change.
	IP Addresses	PE1 and PE2	The IP addresses of all GE interfaces must remain unchanged.

Table 3.3 Chassis Components

	What	Where	How
Chassis	Routing Engine State	PE1 and PE2	All Routing Engines must be online after the change.
	Routing Engine 0 is the Master RE	PE1 and PE2	The RE0 should be the Master RE.
	FPCs State	PE1 and PE2	The state and number of the FPCs must be the same before and after the change.
	PICs State	PE1 and PE2	The state and number of the PICs must be the same before and after the change.
	Hardware Components	PE1 and PE2	All hardware components must be the same before and after the change.

Table 3.4 MP-iBGP Components

	What	Where	How
MP-iBGP	MP-iBGP Sessions	PE1 and PE2	The state of the MP-iBGP sessions must be the same before and after the change.
	MP-iBGP Sessions	PE1 and PE2	The number of routes on each MP-iBGP session must not change more than 30%.
	MP-iBGP Sessions	PE1 and PE2	The families enabled on all MP-iBGP sessions must remain the same before and after the change.

Table 3.5 CE Router Components

	What	Where	How
CE Routers	BGP Sessions	PE1 and PE2	The BGP session between PE and CE must be UP after the change.
	BGP Routes	PE1 and PE2	The number of active, received, and advertised BGP routes should remain the same after the change.

Table 3.6 RSVP Components

	What	Where	How
RSVP	Interfaces	PE1 and PE2	There should be no change in the RSVP topology after the change.
	Sessions	PE1 and PE2	All RSVP sessions must be UP after the change.

Table 3.7 MPLS Components

	What	Where	How
MPLS	Interfaces	PE1 and PE2	The interfaces configured for MPLS must remain the same after the change.
	LSPs	PE1 and PE2	All LSPs must be UP after the change.
	LSPs	PE1 and PE2	All LSPs must be running over its primary path after the change.
	LSPs	PE1 and PE2	All secondary LSPs must be UP after the change.

Obviously, this chapter is not creating all possible tests – if this were a real production network there would be many more JSNAP tests than the ones listed here. By way of comparison, the last JSNAP configuration file the author developed for a production network contained more than 250 tests using about 49 Junos operational commands. Imagine how long it would take to compare the pre- and post-outputs of 49 Junos commands for one single router. Now imagine that your change spans across five routers. That would be almost 250 outputs to compare. Will you have time to check each of them in detail? Surely not!

Writing the JSNAP Configuration File

Our focus on creating the configuration file for this book is to demonstrate the power of JSNAP as well as giving you something meaningful that can be applied to any network running a similar configuration. The idea is to give you a starting point and then enable you to further develop the configuration file on your own.

The network verifications this chapter will be conducting with JSNAP have been separated into the following sequence:

- IGP Network Verifications
- Interface Verifications
- Chassis Verifications

- BGP Network Verifications (Note that BGP and MP-iBGP will be merged into a single configuration file here.)
- RSVP Network Verifications
- MPLS Network Verifications

Developing the JSNAP Configuration File – IGP Network Verifications

In the IGP area, there are two items to check. These two items will possibly have more than two tests, but as the first step in this development, you need to analyse the XML output of the commands that will be used to generate the snapshots. The commands used to generate the snapshots are: **show isis adjacency** and **show isis interface**. The following shows the output of the **show isis adjacency** command, as well as its output in XML format collected by JSNAP:

```
/*
 * Output of show isis adjacency – Text Format
 */
juniper@PE1> show isis adjacency
Interface          System      L State      Hold (secs) SNPA
ge-0/0/3.0         PE2        2 Up         7 56:68:28:33:47:1b

juniper@PE1>

/*
 * Output of show isis adjacency – XML Format – Collected from CLI
 */
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/12.3I0/junos">
  <isis-adjacency-information xmlns="http://xml.juniper.net/junos/12.3I0/junos-
routing" junos:style="brief">
    <isis-adjacency>
      <interface-name>ge-0/0/3.0</interface-name>
      <system-name>PE2</system-name>
      <level>2</level>
      <adjacency-state>Up</adjacency-state>
      <holdtime>6</holdtime>
      <snpa>56:68:28:33:4d:42</snpa>
    </isis-adjacency>
  </isis-adjacency-information>
</cli>
  <banner></banner>
</cli>
</rpc-reply>

/*
 * Output of show isis adjacency – XML Format – Collected by JSNAP
 */
<?xml version="1.0"?>
```

```
<jppc:section conf="isis.conf" mode="sample_isis_snap" name="show_isis_adjacency"
target="PE1" ts="2014-01-23T12:16:25-08:00" xmlns:jppc="http://xml.juniper.net/jppc">
  <isis-adjacency>
    <interface-name>ge-0/0/3.0</interface-name>
    <system-name>PE2</system-name>
    <level>2</level>
    <adjacency-state>Up</adjacency-state>
    <holdtime>7</holdtime>
    <snpa>56:68:28:33:47:1b</snpa>
  </isis-adjacency>
</jppc:section>
```

By the way, you should be interested in checking the elements that are presented in bold. As you probably have noted, the XML output collected directly from the router's CLI and the one collected by JSNAP are slightly different. Can you spot the difference? It is important to note.

JSNAP has modified the XML output produced by the router, replacing the XML root element. It not only replaced the root element but also removed the XML tag **<isis-adjacency-information>**. This is an important detail because it can confuse you when developing the JSNAP configuration file.

TIP When developing the JSNAP configuration file, it is a good practice to first collect an XML output with JSNAP of all Junos commands that you will be using in the configuration file. In order to do that, you can create a JSNAP configuraton file like this:

```
do {
  show_isis_adjacency_snapshot;
  show_isis_interface_snapshot;
}

show_isis_adjacency_snapshot {
  command show isis adjacency;
}

show_isis_interface {
  command show isis interface;
}
```

Okay, before continuing, let's analyze the outputs for the **show isis interface** command presented here:

```
/*
 * Output of show isis interface - Text Format
 */
juniper@PE1> show isis interface
IS-IS interface database:
Interface          L CirID Level 1 DR      Level 2 DR      L1/L2 Metric
```

ge-0/0/3.0	2	0x1 Disabled	PE2.02	10/10
lo0.0	0	0x1 Passive	Passive	0/0

```
juniper@PE1>
```

```
/*
 * Output of show isis interface - XML Format - Collected from CLI
 */
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/12.3I0/junos">
  <isis-interface-information xmlns="http://xml.juniper.net/junos/12.3I0/junos-
routing" junos:style="brief">
    <isis-interface heading="IS-IS interface database:">
      <interface-name>ge-0/0/3.0</interface-name>
      <circuit-type>2</circuit-type>
      <circuit-id>0x1</circuit-id>
      <isis-interface-state-one>Disabled</isis-interface-state-one>
      <dr-id-two>PE2.02</dr-id-two>
      <metric-one>10</metric-one>
      <metric-two>10</metric-two>
    </isis-interface>
    <isis-interface>
      <interface-name>lo0.0</interface-name>
      <circuit-type>0</circuit-type>
      <circuit-id>0x1</circuit-id>
      <isis-interface-state-one>Passive</isis-interface-state-one>
      <isis-interface-state-two>Passive</isis-interface-state-two>
      <metric-one>0</metric-one>
      <metric-two>0</metric-two>
    </isis-interface>
  </isis-interface-information>
</cli>
  <banner></banner>
</cli>
</rpc-reply>

/*
 * Output of show isis interface - XML Format - Collected by JSNAP
 */
<?xml version="1.0"?>
<jppc:section conf="isis.conf" mode="sample_isis_snap" name="show_isis_interface"
target="PE1" ts="2014-01-23T12:16:25-08:00" xmlns:jppc="http://xml.juniper.net/jppc">
  <isis-interface heading="IS-IS interface database:">
    <interface-name>ge-0/0/3.0</interface-name>
    <circuit-type>2</circuit-type>
    <circuit-id>0x1</circuit-id>
    <isis-interface-state-one>Disabled</isis-interface-state-one>
    <dr-id-two>PE2.02</dr-id-two>
    <metric-one>10</metric-one>
    <metric-two>10</metric-two>
  </isis-interface>
  <isis-interface>
    <interface-name>lo0.0</interface-name>
    <circuit-type>0</circuit-type>
```

```

<circuit-id>0x1</circuit-id>
<isis-interface-state-one>Passive</isis-interface-state-one>
<isis-interface-state-two>Passive</isis-interface-state-two>
<metric-one>0</metric-one>
<metric-two>0</metric-two>
</isis-interface>
</jppc:section>

```

When analyzing the XML outputs produced by JSNAP for **show isis adjacency** (omitted for brevity) and **show isis interface** commands, they have identified the following XPath:

```

Command: show isis adjacency
/isis-adjacency
/isis-adjacency/interface-name
/isis-adjacency/system-name
/isis-adjacency/adjacency-state

```

```

Command: show isis interface
/isis-interface
/isis-interface/interface-name
/isis-interface/isis-interface-state-one
/isis-interface/isis-interface-state-two
/isis-interface/metric-one
/isis-interface/metric-two

```

These XPaths are the ones to be used in these tests.

The first thing you need to do is to identify the IDs. For both commands, the XML element you will choose as ID is **interface-name**. This choice makes sense because when you issue these commands in the router, the things you look for are the adjacencies established by each interface.

Once the IDs are identified, you need to identify what kind of tests you will execute against the elements present in these outputs. In the case of **show isis adjacency**, you need to make sure that the adjacencies that are present in the pre-snapshots are present in the post-snapshot and with their same state. Moreover, the neighbor system on that interface, as well as the type of the adjacency and its metric, must remain the same. Finally, you must have the same number of interfaces and adjacencies in the pre- and post-snapshots (assuming your change is not to activate a new backbone link).

Now that the IDs and the tests have been identified, it is time to select the JSNAP operators. The operators that fit in the tests are: **no-diff**, **list-not-less**, and **list-not-more**.

Okay, with everything in place, it's time to create the JSNAP Configuration file for our IGP tests. Here is the complete JSNAP configuration file for the IGP tests:

```

do {
    check_show_isis_interface;
    check_show_isis_adjacency;
}

check_show_isis_interface {
    command show isis interface;
    iterate isis-interface {
        id ./interface-name;
        no-diff interface-name {
            info "Checking the ISIS interface names ...";
            err " ERROR: the interface %s has changed its name from %s to %s.", $ID.1,
$PRE/interface-name, $POST/interface-name;
        }
        no-diff circuit-type {
            info "Checking the ISIS circuit type ...";
            err " ERROR: the interface %s has changed its circuit type from %s to
%s.", $ID.1, $PRE/circuit-type, $POST/circuit-type;
        }
        no-diff isis-interface-state-one {
            info "Checking the L1 interface state ...";
            err " ERROR: the interface %s has changed its L1 interface state from %s
to %s.", $ID.1, $PRE/isis-interface-state-one, $POST/isis-interface-state-one;
        }
        no-diff isis-interface-state-two {
            info "Checking the L2 interface state ...";
            err " ERROR: the interface %s has changed its L2 interface state from %s
to %s.", $ID.1, $PRE/isis-interface-state-two, $POST/isis-interface-state-two;
        }
        no-diff metric-one {
            info "Checking the interface L1 metric ...";
            err " ERROR: the L1 metric for interface %s has changed from %s to %s.",
$ID.1, $PRE/metric-one, $POST/metric-one;
        }
        no-diff metric-two {
            info "Checking the interface L2 metric ...";
            err " ERROR: the L2 metric for interface %s has changed from %s to %s.",
$ID.1, $PRE/metric-two, $POST/metric-two;
        }
        list-not-less interface-name {
            info "Checking for missing ISIS interfaces ...";
            err " ERROR: the interface %s is missing.", $ID.1;
        }
        list-not-more interface-name {
            info "Checking for new ISIS interfaces ...";
            err " ERROR: the interface %s was not configured before.", $ID.1;
        }
    }
}

check_show_isis_adjacency {
    command show isis adjacency;
    iterate isis-adjacency {
        id ./interface-name;
    }
}

```

```

        no-diff interface-name {
            info "Checking the ISIS interface names ...";
            err " ERROR: the interface %s has changed from %s to %s.", $ID.1, $PRE/
interface-name, $POST/interface-name;
        }
        no-diff system-name {
            info "Checking the ISIS neighbour ...";
            err " ERROR: the ISIS neighbour on interface %s has changed from %s to
%s.", $ID.1, $PRE/system-name, $POST/system-name;
        }
        no-diff level {
            info "Checking the Level of the ISIS adjacency ...";
            err " ERROR: the ISIS Level on interface %s has changed from % to %s.",
$ID.1, $PRE/level, $POST/level;
        }
        no-diff snpa {
            info "Checking the Subnetwork Point of Attachment (SNPA) ...";
            err " ERROR: the SNPA for interface %s has changed from %s to %s.", $ID.1,
$PRE/snpa, $POST/snpa;
        }
        list-not-less {
            info "Checking for missing ISIS adjacencies ...";
            err " ERROR: the ISIS adjacency for interface %s is missing.", $ID.1;
        }
        list-not-more {
            info "Checking for new ISIS adjacencies ...";
            err " ERROR: the ISIS adjacency for interface %s was not configured
before.", $ID.1;
        }
    }
}

```

If you test this configuration file, depending on the ISIS configuration you have in your network, you will find that some of the tests may fail even if there are no changes in the ISIS before and after the network change. This case and few others throughout the book will be covered in the Chapter 4.

MORE? For easier cutting and pasting into your lab devices, the configuration files for this book can be downloaded on this book's landing page at <http://www.juniper.net/dayone>.

Okay, you've completed the IGP part. Now, let's develop the configuration file for the interface tests.

Developing the JSNAP Configuration File – Interface Verifications

There are four tests to be executed in Interfaces: operating status, admin status, number of interfaces, and logical addresses. There is more than one Junos command that can be used to achieve the objec-

tive of the Interface verifications. For instance, one could choose the **show interfaces** command while another could choose the **show interfaces terse** command.

TIP

Whenever you are designing, operating, or automating a network, you need to always think ahead. That means you always need to optimize whatever you are doing. Since you have more than one option to use, you should choose the most optimized option. In this case, it will be the **show interfaces terse** command. Be aware that under some types of deployments, for instance BNG, your router may have dozens of thousands interfaces. In those scenarios, you have to add a selector to your command. Let's say you want to limit this analysis to all GE interfaces only. The command used for this case would be **show interfaces terse ge-***.

After you have identified the command to be used, let's look at its outputs. The following is from the **show interfaces terse** output and its XML format collected by JSNAP:

```
juniper@PE1> show interfaces terse ge-*
Interface      Admin Link Proto  Local          Remote
ge-0/0/0       up    up
ge-0/0/0.0     up    up    inet    10.233.255.203/20
ge-0/0/1       up    up
ge-0/0/1.0     up    up    inet    192.168.1.1/30
ge-0/0/2       up    up
ge-0/0/2.0     up    up    inet    192.168.2.1/30
ge-0/0/3       up    up
ge-0/0/3.0     up    up    inet    10.1.1.1/30
               iso
               mpls
```

```
juniper@PE1>
```

```
<?xml version="1.0"?>
<jppc:section conf="network_verifications.conf" mode="before" name="check_show_
interfaces_terse" target="pe1" ts="2014-03-11T06:28:47+11:00" xmlns:jppc="http://xml.
juniper.net/jppc">
  <physical-interface>
    <name>ge-0/0/0</name>
    <admin-status>up</admin-status>
    <oper-status>up</oper-status>
  </physical-interface>
  <logical-interface>
    <name>ge-0/0/0.0</name>
    <admin-status>up</admin-status>
    <oper-status>up</oper-status>
    <filter-information/>
    <address-family>
      <address-family-name>inet</address-family-name>
      <interface-address>
```



```

        <ifa-local junos:emit="emit" xmlns:junos="http://xml.juniper.net/
junos/*/junos">10.233.248.40/20</ifa-local>
        </interface-address>
    </address-family>
</logical-interface>
</physical-interface>
<physical-interface>
    <name>ge-0/0/1</name>
    <admin-status>up</admin-status>
    <oper-status>up</oper-status>
    <logical-interface>
        <name>ge-0/0/1.0</name>
        <admin-status>up</admin-status>
        <oper-status>up</oper-status>
        <description>Connection to CE10</description>
        <filter-information/>
        <address-family>
            <address-family-name>inet</address-family-name>
            <interface-address>
                <ifa-local junos:emit="emit" xmlns:junos="http://xml.juniper.net/
junos/*/junos">192.168.1.1/30</ifa-local>
                </interface-address>
            </address-family>
        </logical-interface>
    </physical-interface>
<physical-interface>
    <name>ge-0/0/2</name>
    <admin-status>up</admin-status>
    <oper-status>up</oper-status>
    <logical-interface>
        <name>ge-0/0/2.0</name>
        <admin-status>up</admin-status>
        <oper-status>up</oper-status>
        <description>Connection to CE11</description>
        <filter-information/>
        <address-family>
            <address-family-name>inet</address-family-name>
            <interface-address>
                <ifa-local junos:emit="emit" xmlns:junos="http://xml.juniper.net/
junos/*/junos">192.168.2.1/30</ifa-local>
                </interface-address>
            </address-family>
        </logical-interface>
    </physical-interface>
<physical-interface>
    <name>ge-0/0/3</name>
    <admin-status>up</admin-status>
    <oper-status>up</oper-status>
    <logical-interface>
        <name>ge-0/0/3.0</name>
        <admin-status>up</admin-status>
        <oper-status>up</oper-status>
        <description>Connection to PE2</description>
        <filter-information/>

```

```

    <address-family>
      <address-family-name>inet</address-family-name>
      <interface-address>
        <ifa-local junos:emit="emit" xmlns:junos="http://xml.juniper.net/
junos/*//junos">10.1.1.1/30</ifa-local>
      </interface-address>
    </address-family>
  </address-family>
  <address-family-name>iso</address-family-name>
</address-family>
<address-family>
  <address-family-name junos:emit="emit" xmlns:junos="http://xml.
juniper.net/junos/*//junos">mpls</address-family-name>
</address-family>
</logical-interface>
</physical-interface>
</jppc:section>

```

The tests identified for the Interfaces section are highlighted in the text and XML output.. If you analyze these outputs, the following XPathS are identified:

```

Command: show interfaces terse ge-*
/physical-interface
/physical-interface/name
/physical-interface/admin-status
/physical-interface/oper-status
/physical-interface/logical-interface
/physical-interface/logical-interface/name
/physical-interface/logical-interface/admin-status
/physical-interface/logical-interface/oper-status
/physical-interface/logical-interface/address-family/
/physical-interface/logical-interface/address-family/address-family-name
/physical-interface/logical-interface/address-family/interface-address
/physical-interface/global-interface/address-family/interface-address/ifa-local

```

Following the same procedure used on the IGP tests, the first thing to do is the identification of the IDs to be used in the tests. For the Interface tests, the XML element identified for ID is **name**. As you noted, there are two different XML elements for **name**. Although they have the same name, they are in different levels of the XML tree.

Now that we have the IDs identified, it is time to identify the tests. As defined in Table 3.1 in Chapter 3, the administrative and operational status of all GE interfaces must remain the same before and after the change. Moreover, you must not have missing GE interfaces or new GE interfaces after the change. Finally, and importantly, the IP addresses of all interfaces must remain the same.

One important thing to note in this verification is the fact the scope of the **show interfaces terse** command has been limited to GE interfaces only. The explanation for this is covered in the Chapter 4.

Having the IDs and tests identified, you can now select the JSNAP test operators. The operators that best fit in the interface tests are: **no-diff**, **list-not-less**, and **list-not-more**.

With all elements identified, let's run the JSNAP Configuration file for the Interface tests. The following output shows the complete JSNAP configuration file:.

```
do {
    check_show_interfaces_terse;
}

check_show_interfaces_terse {
    command show interfaces terse ge-*;
    iterate physical-interface {
        id ./name;
        no-diff oper-status {
            info "Checking PHY operational status of interfaces ...";
            err "  ERROR: the operational status for interface %s has changed
from %s to %s.", $ID.1, $PRE/oper-status, $POST/oper-status;
        }
        no-diff admin-status {
            info "Checking PHY admin status of interfaces ...";
            err "  ERROR: the admin status for interface %s has changed from %s
to %s.", $ID.1, $PRE/admin-status, $POST/admin-status;
        }
        list-not-less name {
            info "Checking for missing interfaces at PHY level ...";
            err "  ERROR: the interface %s is missing.", $ID.1;
        }
        list-not-more {
            info "Checking for new interfaces at PHY level ...";
            err "  ERROR: the interface %s is new.", $ID.1;
        }
    }
    iterate physical-interface/logical-interface {
        id ./name;
        id ./address-family/address-family-name;
        id ./address-family/interface-address/ifa-local;
        no-diff oper-status {
            info "Checking LOGICAL operational status of interfaces ...";
            err "  ERROR: the operational status for interface %s has changed
from %s to %s.", $ID.1, $PRE/oper-status, $POST/oper-status;
        }
        no-diff admin-status {
            info "Checking LOGICAL admin status of interfaces ...";
            err "  ERROR: the admin status of interface %s has changed from %s
to %s.", $ID.1, $PRE/admin-status, $POST/admin-status;
        }
        list-not-less {
            info "Checking for missing interfaces at LOGICAL level ...";
            err "  ERROR: the interface %s is missing.", $ID.1;
        }
        list-not-more {
```

```

        info "Checking for new interfaces at LOGICAL level ...";
        err "    ERROR: the interface %s is new.", $ID.1;
    }
    no-diff address-family/address-family-name {
        info "Checking the address family configured in the interfaces
...";
        err "    ERROR: the address family for interface %s has changed from
%s to %s.", $ID.1, $PRE/address-family/address-family-name, $POST/address-family/
address-family-name;
    }
    list-not-less address-family/address-family-name {
        info "Checking for missing address family ...";
        err "    ERROR: the address family %s is missing on interface %s.",
$ID.2, $ID.1;
    }
    list-not-more address-family/address-family-name {
        info "Checking for new family address ...";
        err "    ERROR: the address family %s has been added to the interface
%s.", $ID.2, $ID.1;
    }
    no-diff address-family/interface-address/ifa-local {
        info "Checking the interface address configured under the interface
...";
        err "    ERROR: the interface address for interface %s has changed
from %s to %s.", $ID.1, $PRE/address-family/interface-address/ifa-local, $POST/
address-family/interface-address/ifa-local;
    }
    list-not-less address-family/interface-address/ifa-local {
        info "Checking for missing interface address ...";
        err "    ERROR: the interface address %s has gone missing on
interface %s.", $ID.3, $ID.1;
    }
    list-not-more address-family/interface-address/ifa-local {
        info "Checking for new interface address ...";
        err "    ERROR: the interface address %s has been added to the
interface %s.", $ID.3, $ID.1;
    }
}
}

```

MORE? More information about XML can be found at <http://www.w3.org/XML/Datamodel.html>.

Before moving to the next group of tests, let's analyze some new things that appeared in this Interface configuration file.

The first thing to note is that there are two iterations inside the same Junos command, and this is perfectly possible and in fact, you can have as many iterations as you need. The second thing to note is the number of IDs being used in the second iteration – additional IDs help to make your informational and error messages much more clear and precise.

The next group of tests focus on the router's components.

Developing the JSNAP Configuration File – Chassis Verifications

The Chassis verification is a very particular case, especially if you are planning to share a single JSNAP configuration file across different Junos platforms. For example, MX Series routers will have SCB (or SCBE) fabrics while T Series will have SIBs. In general, Junos routers will have a service PIC while the SRX will have a SPC card, and so on. As long as you stay with the common commands for checking the chassis across all platforms, there should be no problem in sharing the same JSNAP configuration file. If you do not, you may want to create different files for each architecture; or, frankly, you can pack them all together and ignore the errors when a Junos hardware specific command is executed in the wrong platform.

The Chassis verifications tests chosen for this book use general Junos commands that are available in all Junos platforms. There are four tests that have been identified to be automated: the RE's state, the RE0 as Master, the FPC's state, and the PIC's state. The respective Junos commands will be: **show chassis routing-engine**, **show chassis fpc**, **show chassis hardware**, and **show chassis fpc pic-status**.

NOTE *Junosphere* is being used to create the network environment used in this book, and while all Chassis verifications commands are available there, their output is not as complete as the one collected from real routers. For the sake of helping you getting the most out of this book, for this case, a real router is used to collect the outputs.

Here is the output of **show chassis hardware** for a real router in both text and XML formats:

```
lab@kenny> show chassis hardware
```

```
Hardware inventory:
```

Item	Version	Part number	Serial number	Description
Chassis			52061	M10
Midplane	REV 03	710-001950	HF0237	M10 Backplane
Power Supply A	Rev 04	740-002497	LL13962	AC Power Supply
Display	REV 04	710-001995	HE6561	M10 Display Board
Routing Engine	REV 08	740-003877	9000006124	RE-2.0
FEB	REV 07	710-003310	HH4020	E-FEB
FPC 0				E-FPC
PIC 0	REV 02	750-003072	HC4419	1x OC-48 SONET, SMSR
FPC 1				E-FPC
PIC 3	REV 01	750-002982	HB9184	1x Tunnel
Fan Tray				Rear Left Fan Tray

```
lab@kenny>
```

```
<?xml version="1.0"?>
```

```
<jppc:section conf="chassis.conf" mode="chassis_before" name="check_show_chassis_
```

```

hardware" target="kenny-au" ts="2014-02-09T07:51:18+11:00" xmlns:jppc="http://xml.
juniper.net/jppc">
  <chassis junos:style="inventory" xmlns:junos="http://xml.juniper.net/
junos/*/junos">
    <name>Chassis</name>
    <serial-number>52061</serial-number>
    <description>M10</description>
    <chassis-module>
      <name>Midplane</name>
      <version>REV 03</version>
      <part-number>710-001950</part-number>
      <serial-number>HF0237</serial-number>
      <description>M10 Backplane</description>
      <model-number>CHAS-MP-M10-S</model-number>
    </chassis-module>
    <chassis-module>
      <name>Power Supply A</name>
      <version>Rev 04</version>
      <part-number>740-002497</part-number>
      <serial-number>LL13962</serial-number>
      <description>AC Power Supply</description>
      <model-number>PWR-M10-M5-AC-S</model-number>
    </chassis-module>
    <chassis-module>
      <name>Display</name>
      <version>REV 04</version>
      <part-number>710-001995</part-number>
      <serial-number>HE6561</serial-number>
      <description>M10 Display Board</description>
    </chassis-module>
    <chassis-module>
      <name>Routing Engine</name>
      <version>REV 08</version>
      <part-number>740-003877</part-number>
      <serial-number>9000006124</serial-number>
      <description>RE-2.0</description>
      <model-number>RE-333-256-S</model-number>
    </chassis-module>
    <chassis-module>
      <name>FEB</name>
      <version>REV 07</version>
      <part-number>710-003310</part-number>
      <serial-number>HH4020</serial-number>
      <description>E-FEB</description>
      <model-number>FEB-M10-E-S</model-number>
    </chassis-module>
    <chassis-module>
      <name>FPC 0</name>
      <description>E-FPC</description>
      <chassis-sub-module>
        <name>PIC 0</name>
        <version>REV 02</version>
        <part-number>750-003072</part-number>
        <serial-number>HC4419</serial-number>
      </chassis-sub-module>
    </chassis-module>
  </chassis>
</junos>

```

```

        <description>1x OC-48 SONET, SMSR</description>
        <model-number>PE-10C48-SON-SMSR</model-number>
      </chassis-sub-module>
    </chassis-module>
    <chassis-module>
      <name>FPC 1</name>
      <description>E-FPC</description>
      <chassis-sub-module>
        <name>PIC 3</name>
        <version>REV 01</version>
        <part-number>750-002982</part-number>
        <serial-number>HB9184</serial-number>
        <description>1x Tunnel</description>
        <model-number>PE-TUNNEL</model-number>
      </chassis-sub-module>
    </chassis-module>
    <chassis-module>
      <name>Fan Tray</name>
      <description>Rear Left Fan Tray</description>
      <model-number>FANTRAY-M10-M5-S</model-number>
    </chassis-module>
  </chassis>
</jppc:section>

```

This next output shows the **show chassis fpc** in both formats:

```
lab@kenny> show chassis fpc | no-more
```

Slot	State	Temp	CPU Utilization (%)		Memory	Utilization (%)	
		(C)	Total	Interrupt	DRAM (MB)	Heap	Buffer
0	Online	27	3	0	64	26	48
1	Online	27	3	0	64	26	48

```
lab@kenny>
```

```

<?xml version="1.0"?>
<jppc:section conf="chassis.conf" mode="chassis_before" name="check_show_chassis_fpc"
target="kenny-au" ts="2014-02-09T07:51:18+11:00" xmlns:jppc="http://xml.juniper.net/
jppc">
  <fpc>
    <slot>0</slot>
    <state>Online</state>
    <temperature junos:celsius="28" xmlns:junos="http://xml.juniper.net/
junos/*/junos">28</temperature>
    <cpu-total>2</cpu-total>
    <cpu-interrupt>0</cpu-interrupt>
    <memory-dram-size>64</memory-dram-size>
    <memory-heap-utilization>26</memory-heap-utilization>
    <memory-buffer-utilization>48</memory-buffer-utilization>
  </fpc>
  <fpc>
    <slot>1</slot>
    <state>Online</state>
    <temperature junos:celsius="27" xmlns:junos="http://xml.juniper.net/

```

```

junos/*/junos">27</temperature>
    <cpu-total>2</cpu-total>
    <cpu-interrupt>0</cpu-interrupt>
    <memory-dram-size>64</memory-dram-size>
    <memory-heap-utilization>26</memory-heap-utilization>
    <memory-buffer-utilization>48</memory-buffer-utilization>
  </fpc>
</jppc:section>

```

Last but not least, here's the status of the PICs:.

```

lab@kenny> show chassis fpc pic-status | no-more
Slot 0  Online      E-FPC
  PIC 0  Online      1x OC-48 SONET, SMSR
Slot 1  Online      E-FPC
  PIC 3  Online      1x Tunnel
lab@kenny>

```

```

<?xml version="1.0"?>
<jppc:section conf="chassis.conf" mode="chassis_before" name="check_show_chassis_fpc_
pic_status" target="kenny-au" ts="2014-02-09T07:51:18+11:00" xmlns:jppc="http://xml.
juniper.net/jppc">
  <fpc>
    <slot>0</slot>
    <state>Online</state>
    <description>E-FPC</description>
    <pic>
      <pic-slot>0</pic-slot>
      <pic-state>Online</pic-state>
      <pic-type>1x OC-48 SONET, SMSR</pic-type>
    </pic>
  </fpc>
  <fpc>
    <slot>1</slot>
    <state>Online</state>
    <description>E-FPC</description>
    <pic>
      <pic-slot>3</pic-slot>
      <pic-state>Online</pic-state>
      <pic-type>1x Tunnel</pic-type>
    </pic>
  </fpc>
</jppc:section>

```

The output of **show chassis hardware** presented here introduces a level of complexity to our JSNAP configuration file. First of all, you have items that do not have version or part numbers. Second, if you look at the XML output, you can observe two hierarchies: *chassis-module* and *chassis-sub-module*. The trick here is that not all modules will have this chassis-sub-module section, and while it may not make sense to you right now, you will better understand in Chapter 4.

Finally, as you probably have noticed, the XML output of **show chassis hardware** has some elements that are not presented in its text form. In order to display these extra elements in the normal form of the output, you need to add an output modifier such as | **detail** or | **extensive**.

Just like the hardware components of the router, the FPC status output also introduces something new. This is the first output that you can't simply check for an exact match on these numbers because they are likely to be different between one snapshot and another.

The last output of this section, the PIC status, is the simplest output to be analyzed. It can be processed by JSNAP in the same way as for the IGP and Interfaces sections.

Now let's list all the XPathS needed to be analyzed with JSNAP. The first group of XPathS is related to the hardware components:

```
Command: show chassis hardware
/chassis/chassis-module
/chassis/chassis-module/name
/chassis/chassis-module/version
/chassis/chassis-module/part-number
/chassis/chassis-module/serial-number
/chassis/chassis-module/description
/chassis/chassis-module/model-number
/chassis/chassis-module/chassis-sub-module
/chassis/chassis-module/chassis-sub-module/name
/chassis/chassis-module/chassis-sub-module/version
/chassis/chassis-module/chassis-sub-module/part-number
/chassis/chassis-module/chassis-sub-module/serial-number
/chassis/chassis-module/chassis-sub-module/description
/chassis/chassis-module/chassis-sub-module/model-number
```

The best operator to test these XPathS is the **no-diff** operator. You may also want to test if new chassis modules and sub-modules were inserted or removed. For this, you have to use the **list-not-less** and **list-not-more** operators. The IDs for these XPathS will be named on both the **chassis-module** and **chassis-sub-module** hierarchies.

The second group of XPath for this section is the **show chassis fpc** XPathS.

```
Command: show chassis fpc
/fpc
/fpc/slot
/fpc/state
/fpc/temperature
/fpc/cpu-total
/fpc/memory-heap-utilization
/fpc/memory-buffer-utilization
```

As mentioned before, the **show chassis fpc** output has numeric values. There are a few different ways of comparing numeric values: you can verify if the number is within a specified range, if it is equal to a number, or if it has changed more than a specified value.

For temperature values, use the operator **is-lt** to check if the FPC's temperature is less than 55-Celsius degrees (131-Fahrenheit).

NOTE We are using the value of 55-Celsius degrees here because this is the value that will trigger the Yellow alarm for temperature on an M10i router. (See http://www.juniper.net/techpubs/en_US/release-independent/junos/information-products/topic-collections/hardware/m-series/m10i/hwguide/m10i-hwguide.pdf.) Remember to verify the environmental conditions of the hardware you are working with before defining this parameter.

The CPU utilization is a little tricky because it is very hard to define what is the correct number, or range, the values should be, in order to say that CPU is *fine*. The reason is that CPU spikes are sometimes normal and expected. Best practice says that the CPU must be monitored when the router has reached the steady state: routing protocols are UP, running, and stable, the routing tables have converged, and the traffic is flowing normally. (Note that the definition of *steady state* includes much more than the examples cited here and this varies from network to network.) So, what's the number? Let's use 70% and use the **is-lt** operator to make sure the CPU value is below 70%. You may want to fine-tune this number to best fit in your lab or network.

Monitoring memory utilization is very important and a bit easier to define than the CPU. Usually, two verifications are used for memory: one to verify if the memory has changed more than X% during the change, and another to check if the memory is below Y% (or within the range of, if you prefer). Again, defining the X and Y numbers comes down to your network design and the hardware involved. A general and safe rule of thumb is 70% for Y. If you want to check if the utilization is within a range, you could check for 20% ~ 70% range. The selection of X may depend on the type of the change you are doing because some changes are expected to cause more churn in the memory than others (for example, routing protocol changes). As a rule of thumb, let's go with 15% of variation. If you are asking yourself why 15% and not 20%, the explanation is that 70% + 15% will give you 85%, which is considered very high but still gives you some room to take action before the system cracks down. Following the same idea of the CPU values, you should adjust this value to best fit your network.

The heap and buffer utilization follow the same rules of memory

monitoring. In addition, the same range and variation can be used for both items.

NOTE Each ASIC family has its own memory management algorithm. For instance, *I-Chip*, *Trio*, and *Cassis* have different internal memory architecture, which may affect the maximum value selected for this test. Moreover, in case your network is unstable, this value is expected to oscillate within a wider range.

Finally, yet importantly, the state of each FPC must be monitored. For this test, you can use the well-known **no-diff** operator.

The ID identified for the **show chassis fpc** command is **slot**.

The last test of this section is the verification of the PIC cards. The third group of XPath's is listed below:

```
Command: show chassis fpc pic-status
/fpc
/fpc/slot
/fpc/pic
/fpc/pic/pic-slot
/fpc/pic/pic-state
/fpc/pic/pic-type
```

The output of **show chassis fpc pic-status** was a very simple one – it can be verified using the **no-diff** operator. The difference from this output from the all others you have seen so far is the number of IDs that are required to identify the element being tested. For example, identifying the test just by using the **pic-slot** is not a good option as a router may contain many PIC slots 0. In this case, you have to define two IDs: one for the FPC slot (chassis slot) where the PIC is inserted, and the other for the slot of the FPC (sub-slot) where the PIC is inserted. Therefore, the IDs identified for this test are: **slot** and **pic-slot**.

Now that you have learned how to use some new JSNAP operators, it's time to create our JSNAP configuration file for the Chassis tests:

```
do {
    check_show_chassis_hardware;
    check_show_chassis_fpc;
    check_show_chassis_fpc_pic_status;
}

check_show_chassis_hardware {
    command show chassis hardware;
    iterate chassis/chassis-module {
        id name;
        no-diff name {
            info "Checking chassis modules names ...";
```

```

    err " ERROR: the module %s has changed from %s to %s.", $ID.1, $PRE/name,
$POST/name;
}
no-diff version {
    info "Checking chassis module version ...";
    err " ERROR: the version of the module %s has changed from %s to %s.", $ID.1,
$PRE/name, $POST/name;
}
no-diff part-number {
    info "Checking chassis module part-number ...";
    err " ERROR: the part-number of module %s has changed from %s to %s.", $ID.1,
$PRE/part-number, $POST/part-number;
}
no-diff serial-number {
    info "Checking chassis module serial-number ...";
    err " ERROR: the serial-number of module %s has changed from %s to %s.",
$ID.1, $PRE/serial-number, $POST/serial-number;
}
no-diff description {
    info "Checking chassis module description ...";
    err " ERROR: the description of module %s has changed from %s to %s.", $ID.1,
$PRE/description, $POST/description;
}
no-diff model-number {
    info "Checking chassis module model-number ...";
    err " ERROR: the model-number of module %s has changed from %s to %s.", $ID.1,
$PRE/model-number, $POST/model-number;
}
list-not-less name {
    info "Checking for missing modules ...";
    err " ERROR: the module %s is missing.", $ID.1;
}
list-not-more name {
    info "Checking for new modules ...";
    err " ERROR: the module %s was installed.", $ID.1;
}
}
iterate chassis/chassis-module/chassis-sub-module {
    id name;
    id ../name;
    no-diff name {
        info "Checking chassis sub-modules names ...";
        err " ERROR: the sub-module %s of module %s has changed from %s to %s.",
$ID.1, $ID.2, $PRE/name, $POST/name;
    }
    no-diff version {
        info "Checking chassis sub-module version ...";
        err " ERROR: the version of the sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/name, $POST/name;
    }
    no-diff part-number {
        info "Checking chassis sub-module part-number ...";
        err " ERROR: the part-number of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/part-number, $POST/part-number;
    }
}

```

```

    }
    no-diff serial-number {
        info "Checking chassis sub-module serial-number ...";
        err " ERROR: the serial-number of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/serial-number, $POST/serial-number;
    }
    no-diff description {
        info "Checking chassis sub-module description ...";
        err " ERROR: the description of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/description, $POST/description;
    }
    no-diff model-number {
        info "Checking chassis sub-module model-number ...";
        err " ERROR: the model-number of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/model-number, $POST/model-number;
    }
    list-not-less name {
        info "Checking for missing sub-modules ...";
        err " ERROR: the sub-module %s of module %s is missing.", $ID.1, $ID.2;
    }
    list-not-more name {
        info "Checking for new sub-modules ...";
        err " ERROR: the sub-module %s of module %s was installed.", $ID.1, $ID.2;
    }
}

check_show_chassis_fpc {
    command show chassis fpc;
    iterate fpc {
        id slot;
        no-diff state {
            info "Checking FPC state ...";
            err " ERROR: the FPC %s has changed its state from %s to %s.", $ID.1, $PRE/
state, $POST/state;
        }
        list-not-less {
            info "Checking for missing FPCs ...";
            err " ERROR: the FPC %s is missing.", $ID.1;
        }
        list-not-more {
            info "Checking for new FPCs ...";
            err " ERROR: the FPC %s was installed.", $ID.1;
        }
    }
    is-lt temperature, 55 {
        info "Checking if the temperature of the FPCs is below 55-Celsius degrees
(131-Farenheit).";
        err " ERROR: the temperature of FPC %s is %s (before was %s) Celsius
degrees.", $ID.1, $POST/temperature, $PRE/temperature;
    }
    is-lt cpu-total, 70 {
        info "Checking if the CPU utilisation of the FPCs is below 70%.";
        err " ERROR: the CPU utilisation of FPC %s is %s (before was %s).", $ID.1,
$POST/cpu-total, $PRE/cpu-total;
    }
}

```

```

    }
    in-range memory-heap-utilization, 20, 70 {
        info "Checking if the memory heap utilisation of the FPCs is within the
range of 20% ~ 70%.";
        err " ERROR: the memory heap utilisation of FPC %s is out-of-range (
Before = %s / After = %s ).", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
    delta memory-heap-utilization, 15% {
        info "Checking if the memory heap utilisation of the FPCs has changed more
than 15%.";
        err " ERROR: the memory heap utilisation of the FPC %s has changed from %s
to %s.", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
    in-range memory-buffer-utilization, 20, 70 {
        info "Checking if the memory buffer utilisation of the FPCs is within the
range of 20% ~ 70%.";
        err " ERROR: the memory buffer utilisation of FPC %s is out-of-range (
Before = %s / After = %s ).", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
    delta memory-buffer-utilization, 15% {
        info "Checking if the memory buffer utilisation of the FPCs has changed
more than 15%.";
        err " ERROR: the memory buffer utilisation of the FPC %s has changed from
%s to %s.", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
}

check_show_chassis_fpc_pic_status {
    command show chassis fpc pic-status;
    iterate fpc/pic {
        id pic-slot;
        id ../slot;
        no-diff pic-type {
            info "Checking the PIC types ...";
            err " ERROR: the PIC type of PIC %s of FPC slot %s has changed from %s to
%s.", $ID.1, $ID.2, $PRE/pic-type, $POST/pic-type;
        }
        no-diff pic-state {
            info "Checking the PIC state ...";
            err " ERROR: the state of PIC %s of FPC slot %s has changed from %s to
%s.", $ID.1, $ID.2, $PRE/pic-state, $POST/pic-state;
        }
        list-not-less pic-slot {
            info "Checking for missing PICs ...";
            err " ERROR: the PIC %s of FPC slot %s is missing.", $ID.1, $ID.2;
        }
        list-not-more pic-slot {
            info "Checking for new PICs ...";
            err " ERROR: the PIC %s of FPC slot %s was installed.", $ID.1, $ID.2;
        }
    }
}

```

The Chassis verification required a wider range of JSNAP operators.

You now have examples of dealing with numbers and multiple IDs.

Verifying the Routing Engines

You may have noticed that the routine to verify the Routing Engines of the router was not included. Let's make it an exercise of this chapter.

The list below contains the items you need to verify in the output of **show chassis routing-engine**:

- CPU Utilisation (Idle)
- Mastership on RE0
- Routing-Engine Temperature below 55 °C

The answer for this exercise is presented in the Appendix at the end of this *Day One* book.

By the way, if you test this Chassis configuration file in your lab, you will find that few tests are failing even though the pre- and post-snapshots are identical. That is an expected situation and it is covered in detail in Chapter 4.

Now, let's move to the next section of the configuration file: MP-iBGP.

Developing the JSNAP Configuration File – BGP Network Verifications

Although this chapter separates the BGP verifications into MP-iBGP sessions *and* PE-CE BGP sessions, from the PE perspective they are *all* BGP sessions. The differences between them are a type of the session (*internal* versus *external*) and the instance where they run. Creating a separate section to check MP-iBGP and another for PE-CE BGP sessions may be too complicated for the brevity desired in this *Day One* book, therefore the tests will be merged into a single section in the JSNAP configuration file.

To start with, let's select the BGP operational command to be used in these BGP verifications. The most appropriate command is **show bgp summary**, which is presented in both normal and XML (and it's back to using Junosphere for the output):.

```
juniper@PE1> show bgp summary
Groups: 3 Peers: 3 Down peers: 0
Table      Tot Paths  Act Paths Suppressed  History Damp State   Pending
inet.0
          0         0         0         0         0         0
bgp.13vpn.0
          6         6         0         0         0         0
bgp.mvpn.0
          0         0         0         0         0         0
```

Peer	AS	InPkt	OutPkt	OutQ	Flaps	Last Up/Dwn	State #Active/
10.100.100.2	65000	35	34	0	0	11:41	Establ
inet.0: 0/0/0/0							
bgp.l3vpn.0: 6/6/6/0							
bgp.rtarget.0: 0/1/1/0							
bgp.mvpn.0: 0/0/0/0							
VPNA.inet.0: 3/3/3/0							
VPNB.inet.0: 3/3/3/0							
192.168.1.2	65100	31	32	0	0	12:19	Establ
VPNA.inet.0: 2/2/2/0							
192.168.2.2	65100	31	32	0	0	12:22	Establ
VPNB.inet.0: 2/2/2/0							

```
<?xml version="1.0"?>
<jppc:section conf="bgp.conf" mode="pre_bgp" name="bgp-summary" target="pe1" ts="2014-
02-14T07:05:02+11:00" xmlns:jppc="http://xml.juniper.net/jppc">
  <group-count>3</group-count>
  <peer-count>3</peer-count>
  <down-peer-count>0</down-peer-count>
  <bgp-rib junos:style="brief" xmlns:junos="http://xml.juniper.net/junos/*/
junos">
    <name>inet.0</name>
    <total-prefix-count>0</total-prefix-count>
    <received-prefix-count>0</received-prefix-count>
    <accepted-prefix-count>0</accepted-prefix-count>
    <active-prefix-count>0</active-prefix-count>
    <suppressed-prefix-count>0</suppressed-prefix-count>
    <history-prefix-count>0</history-prefix-count>
    <damped-prefix-count>0</damped-prefix-count>
    <total-external-prefix-count>0</total-external-prefix-count>
    <active-external-prefix-count>0</active-external-prefix-count>
    <accepted-external-prefix-count>0</accepted-external-prefix-count>
    <suppressed-external-prefix-count>0</suppressed-external-prefix-count>
    <total-internal-prefix-count>0</total-internal-prefix-count>
    <active-internal-prefix-count>0</active-internal-prefix-count>
    <accepted-internal-prefix-count>0</accepted-internal-prefix-count>
    <suppressed-internal-prefix-count>0</suppressed-internal-prefix-count>
    <pending-prefix-count>0</pending-prefix-count>
    <bgp-rib-state>BGP restart is complete</bgp-rib-state>
  </bgp-rib>
  <bgp-rib junos:style="brief" xmlns:junos="http://xml.juniper.net/junos/*/
junos">
    <name>bgp.l3vpn.0</name>
    <total-prefix-count>6</total-prefix-count>
    <received-prefix-count>6</received-prefix-count>
    <accepted-prefix-count>6</accepted-prefix-count>
    <active-prefix-count>6</active-prefix-count>
    <suppressed-prefix-count>0</suppressed-prefix-count>
    <history-prefix-count>0</history-prefix-count>
    <damped-prefix-count>0</damped-prefix-count>
    <total-external-prefix-count>0</total-external-prefix-count>
    <active-external-prefix-count>0</active-external-prefix-count>
```



```

    <accepted-external-prefix-count>0</accepted-external-prefix-count>
    <suppressed-external-prefix-count>0</suppressed-external-prefix-count>
    <total-internal-prefix-count>6</total-internal-prefix-count>
    <active-internal-prefix-count>6</active-internal-prefix-count>
    <accepted-internal-prefix-count>6</accepted-internal-prefix-count>
    <suppressed-internal-prefix-count>0</suppressed-internal-prefix-count>
    <pending-prefix-count>0</pending-prefix-count>
    <bgp-rib-state>BGP restart is complete</bgp-rib-state>
    <vpn-rib-state>VPN restart is complete</vpn-rib-state>
  </bgp-rib>
  <bgp-rib junos:style="brief" xmlns:junos="http://xml.juniper.net/junos/*"
junos">
    <name>bgp.mvpn.0</name>
    <total-prefix-count>0</total-prefix-count>
    <received-prefix-count>0</received-prefix-count>
    <accepted-prefix-count>0</accepted-prefix-count>
    <active-prefix-count>0</active-prefix-count>
    <suppressed-prefix-count>0</suppressed-prefix-count>
    <history-prefix-count>0</history-prefix-count>
    <damped-prefix-count>0</damped-prefix-count>
    <total-external-prefix-count>0</total-external-prefix-count>
    <active-external-prefix-count>0</active-external-prefix-count>
    <accepted-external-prefix-count>0</accepted-external-prefix-count>
    <suppressed-external-prefix-count>0</suppressed-external-prefix-count>
    <total-internal-prefix-count>0</total-internal-prefix-count>
    <active-internal-prefix-count>0</active-internal-prefix-count>
    <accepted-internal-prefix-count>0</accepted-internal-prefix-count>
    <suppressed-internal-prefix-count>0</suppressed-internal-prefix-count>
    <pending-prefix-count>0</pending-prefix-count>
    <bgp-rib-state>BGP restart is complete</bgp-rib-state>
    <vpn-rib-state>VPN restart is complete</vpn-rib-state>
  </bgp-rib>
  <bgp-peer heading="Peer          AS      InPkt    OutPkt    OutQ
Flaps Last Up/Dwn State|#Active/Received/Accepted/Damped..." junos:style="terse"
xmlns:junos="http://xml.juniper.net/junos/*/junos">
    <peer-address>10.100.100.2</peer-address>
    <peer-as>65000</peer-as>
    <input-messages>48</input-messages>
    <output-messages>47</output-messages>
    <route-queue-count>0</route-queue-count>
    <flap-count>0</flap-count>
    <elapsed-time junos:seconds="1055">17:35</elapsed-time>
    <peer-state junos:format="Establish">Established</peer-state>
  <bgp-rib junos:style="terse">
    <name>inet.0</name>
    <active-prefix-count>0</active-prefix-count>
    <received-prefix-count>0</received-prefix-count>
    <accepted-prefix-count>0</accepted-prefix-count>
    <suppressed-prefix-count>0</suppressed-prefix-count>
  </bgp-rib>
  <bgp-rib junos:style="terse">
    <name>bgp.13vpn.0</name>
    <active-prefix-count>6</active-prefix-count>
    <received-prefix-count>6</received-prefix-count>

```

```

        <accepted-prefix-count>6</accepted-prefix-count>
        <suppressed-prefix-count>0</suppressed-prefix-count>
    </bgp-rib>
    <bgp-rib junos:style="terse">
        <name>bgp.rtarget.0</name>
        <active-prefix-count>0</active-prefix-count>
        <received-prefix-count>1</received-prefix-count>
        <accepted-prefix-count>1</accepted-prefix-count>
        <suppressed-prefix-count>0</suppressed-prefix-count>
    </bgp-rib>
    <bgp-rib junos:style="terse">
        <name>bgp.mvpn.0</name>
        <active-prefix-count>0</active-prefix-count>
        <received-prefix-count>0</received-prefix-count>
        <accepted-prefix-count>0</accepted-prefix-count>
        <suppressed-prefix-count>0</suppressed-prefix-count>
    </bgp-rib>
    <bgp-rib junos:style="terse">
        <name>VPNA.inet.0</name>
        <active-prefix-count>3</active-prefix-count>
        <received-prefix-count>3</received-prefix-count>
        <accepted-prefix-count>3</accepted-prefix-count>
        <suppressed-prefix-count>0</suppressed-prefix-count>
    </bgp-rib>
    <bgp-rib junos:style="terse">
        <name>VPNB.inet.0</name>
        <active-prefix-count>3</active-prefix-count>
        <received-prefix-count>3</received-prefix-count>
        <accepted-prefix-count>3</accepted-prefix-count>
        <suppressed-prefix-count>0</suppressed-prefix-count>
    </bgp-rib>
</bgp-peer>
<bgp-peer junos:style="terse" xmlns:junos="http://xml.juniper.net/junos/*/
junos">
    <peer-address>192.168.1.2</peer-address>
    <peer-as>65100</peer-as>
    <input-messages>44</input-messages>
    <output-messages>44</output-messages>
    <route-queue-count>0</route-queue-count>
    <flap-count>0</flap-count>
    <elapsed-time junos:seconds="1093">18:13</elapsed-time>
    <peer-state junos:format="Establ">Established</peer-state>
    <bgp-rib junos:style="terse">
        <name>VPNA.inet.0</name>
        <active-prefix-count>2</active-prefix-count>
        <received-prefix-count>2</received-prefix-count>
        <accepted-prefix-count>2</accepted-prefix-count>
        <suppressed-prefix-count>0</suppressed-prefix-count>
    </bgp-rib>
</bgp-peer>
<bgp-peer junos:style="terse" xmlns:junos="http://xml.juniper.net/junos/*/
junos">
    <peer-address>192.168.2.2</peer-address>
    <peer-as>65100</peer-as>

```

```

<input-messages>44</input-messages>
<output-messages>44</output-messages>
<route-queue-count>0</route-queue-count>
<flap-count>0</flap-count>
<elapsed-time junos:seconds="1096">18:16</elapsed-time>
<peer-state junos:format="Establ">Established</peer-state>
<bgp-rib junos:style="terse">
  <name>VPNB.inet.0</name>
  <active-prefix-count>2</active-prefix-count>
  <received-prefix-count>2</received-prefix-count>
  <accepted-prefix-count>2</accepted-prefix-count>
  <suppressed-prefix-count>0</suppressed-prefix-count>
</bgp-rib>
</bgp-peer>
</jppc:section>

```

As you can see, the **show bgp summary** has all the information you need to complete your BGP verifications. The items needed to be checked were also identified.

NOTE This can be the trickiest output to deal with, partially caused by the way Junos organizes the XML output of the **show bgp summary** command. For example, the MP-iBGP sessions will have many *ribs*, including the VPNA and VPNB *ribs*, while the PE-CE BGP sessions will have only its own *rib*. This can sometimes create confusion while deciphering the output.

Analysing the XML output and the elements highlighted in the **show bgp summary** output, you can identify the following XPath paths that you need to work with:

```

Command: show bgp summary
/group-count
/peer-count
/down-peer-count
/bgp-peer
/bgp-peer/peer-address
/bgp-peer/peer-as
/bgp-peer/flap-count
/bgp-peer/peer-state
/bgp-peer/bgp-rib
/bgp-peer/bgp-rib/name
/bgp-peer/bgp-rib/active-prefix-count
/bgp-peer/bgp-rib/received-prefix-count
/bgp-peer/bgp-rib/accepted-prefix-count
/bgp-rib/bgp-rib/suppressed-prefix-count

```

NOTE You probably haven't noticed, but the BGP tests were carefully chosen in order to be able to use the same BGP JSNAP configuration file for both PE and CE routers. The situation would be different if, for

example, you wanted to check to see if the **inet**, **inet-vpn**, and **inet-mvpn** families are enabled on the MP-iBGP peers. It would be very hard to create these tests using a single JSNAP file for BGP for both PE and CE routers. The reason for this additional complexity is because both types of BGP peers share the same XPath structure in the XML output.

When analysing the identified XPaths for these BGP tests, you can identify a good mix of JSNAP operators to be used in these tests. For the first three XPaths, which relate to number of peers, you should use the **no-diff** and **is-gt** JSNAP operators. In fact, the most appropriate operator here is the **no-diff**. However, **is-gt** was introduced here for two reasons: to demonstrate its use, and to show how to create rules like *Each BGP group has to have at least X neighbors configured*.

The next four XPaths relate to the state of the BGP peer. Again, the **no-diff** operator is the best choice here. You could use the **delta** operator for checking the number of times that the BGP session has flapped during the pre- and post-snapshots in a scenario where you are expecting them to flap at least once during the change. One preference is to use **no-diff** because you may want to be alerted anyway whenever a BGP session has flapped. In addition, the state of the BGP peer could be checked with the operator **is-equal**, because as you may know, there is only one state of the BGP session that indicates it is operational: *Established*.

Finally, but not less important, are the the BGP ribs verifications. First, you can check if the ribs remain the same before and after the change using the **no-diff**, **list-not-more**, and **list-not-less** operators. Then, for each rib, you can check if the changes in the number of routes is within a delta using the **delta** operator. Also, if you have expected minimum and maximum numbers for routes on each rib, that can be checked with the **in-range** operator.

NOTE To avoid surprises, monitoring the changes on the number of routes on your network is an important verification. Each network has its own dynamics, so some will have higher churn than others, therefore, you need to identify what is the variation that is considered normal, and, which one is the best delta value to choose for the JSNAP operator.

The last item to identify before getting your hands dirty writing the JSNAP configuration for the BGP tests, are the IDs. Three IDs have been identified for these tests: **peer-address**, **peer-as**, and **rib-name**. With them, you can now develop the JSNAP configuration file for BGP, presented here:

```

do {
    check_bgp_summary;
}

check_bgp_summary {
    command show bgp summary;
    iterate . {
        no-diff group-count {
            info "Checking the number of BGP groups ...";
            err " ERROR: the number of BGP groups has changed from %s to %s.", $PRE/
group-count, $POST/group-count;
        }
        no-diff peer-count {
            info "Checking the number of BGP peers ...";
            err " ERROR: the number of BGP peers has changed from %s to %s.", $PRE/
peer-count, $POST/peer-count;
        }
        is-gt peer-count, 0 {
            info "Checking if the BGP configuration has at least 1 BGP peer configured
...";
            err " ERROR: the BGP configuration does not have any peer configured!";
        }
        no-diff down-peer-count {
            info "Checking the number of BGP peers down ...";
            err " ERROR: the number of BGP peers down has changed from %s to %s.", $PRE/
down-peer-count, $POST/down-peer-count;
        }
    }
    iterate bgp-peer {
        id peer-address;
        id peer-as;
        no-diff peer-address {
            info "Checking if the BGP peers addresses are still the same ...";
            err " ERROR: the BGP peer %s (ASN %s) has changed its address from %s to
%s.", $ID.1, $ID.2, $PRE/peer-address, $POST/peer-address;
        }
        no-diff peer-as {
            info "Checking if the BGP peers ASNs are still the same ...";
            err " ERROR: the ASN for the BGP peer %s has changed from %s to %s.", $ID.1.
$PRE/peer-asn, $POST/peer-asn;
        }
        no-diff flap-count {
            info "Checking if the BGP peer has flapped ...";
            err " ERROR: the BGP peer %s (ASN %s) has flapped.", $ID.1, $ID.2;
        }
        is-equal peer-state, "Established" {
            info "Checking if the BGP peers are in Established state ...";
            err " ERROR: the BGP peer %s (ASN %s) is not in Established state.", $ID.1,
$ID.2;
        }
    }
}
iterate bgp-peer/bgp-rib {
    id name;
    id ../peer-address;
}

```

```

    id ../peer-as;
    no-diff name {
        info "Checking if the BGP RIB name has changed ...";
        err " ERROR: the RIB %s of BGP peer %s (ASN %s) has changed from %s to %s.",
$ID.1, $ID.2, $ID.3, $PRE/name, $POST/name;
    }
    list-not-less name {
        info "Checking for missing RIBs ...";
        err " ERROR: the RIB %s for the BGP peer %s (ASN %s) has gone missing.",
$ID.1, $ID.2, $ID.3;
    }
    list-not-more name {
        info "Checking for new RIBs ...";
        err " ERROR: the RIB %s was not present before for the BGP peer %s (ASN
%s).", $ID.1, $ID.2, $ID.3;
    }
    delta active-prefix-count, 20% {
        info "Checking if the number of BGP active prefix has changed more than
20%.";
        err " ERROR: the number of BGP active prefixes for RIB %s on the BGP peer %s
(ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1, $ID.2,
$ID.3, $PRE/active-prefix-count, $POST/active-prefix-count;
    }
    delta received-prefix-count, 20% {
        info "Checking if the number of BGP received prefix has changed more than
20%.";
        err " ERROR: the number of BGP received prefixes for RIB %s on the BGP peer
%s (ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1, $ID.2,
$ID.3, $PRE/received-prefix-count, $POST/received-prefix-count;
    }
    delta accepted-prefix-count, 20% {
        info "Checking if the number of BGP accepted prefix has changed more than
20%.";
        err " ERROR: the number of BGP accepted prefixes for RIB %s on the BGP peer
%s (ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1, $ID.2,
$ID.3, $PRE/accepted-prefix-count, $POST/accepted-prefix-count;
    }
    delta suppressed-prefix-count, 20% {
        info "Checking if the number of BGP suppressed prefix has changed more than
20%.";
        err " ERROR: the number of BGP suppressed prefixes for RIB %s on the BGP
peer %s (ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1,
$ID.2, $ID.3, $PRE/suppressed-prefix-count, $POST/suppressed-prefix-count;
    }
}
}

```

NOTE This configuration file has something different than the previous ones: the tests against **group-count**, **peer-count**, and **down-peer-count** use a different XPath, which is covered in Chapter 4.

One of the good things about using JSNAP to automate BGP verifications like these is the automation of the calculation methods. Imagine

that the router you are executing these tests upon is an *Internet Edge* router or a *Route Reflector* router where you may have many dozens of BGP peers. Checking each and calculating the deviations one-by-one is something you definitely don't want to do after executing a network change.

With the completion of the JSNAP configuration file for the BGP tests, the development of a full set of automated network verification tests is almost done. The next section will guide you through the development of the network verification tests for the RSVP protocol.

Developing the JSNAP Configuration File – RSVP Network Verifications

Networks that require traffic engineering and MPLS fast-reroute capabilities will make use of the RSVP protocol in order to signal the MPLS LSPs. This feature is commonly found in Service Provider networks but its use on Enterprise networks is increasing. Therefore it's important to include the RSVP network verifications here.

There are two tests identified for RSVP: *interfaces* and *sessions*. Let's start with the verifications of the RSVP sessions. The Junos operational command to be used for the RSVP tests is **show rsvp session**, and here are its normal and XML outputs:

```
juniper@PE1> show rsvp session
Ingress RSVP: 1 sessions
To          From          State  Rt Style Labelin Labelout LSPname
10.100.100.2 10.100.100.1 Up      0  1 FF      -       3 to-PE2
Total 1 displayed, Up 1, Down 0
```

```
Egress RSVP: 1 sessions
To          From          State  Rt Style Labelin Labelout LSPname
10.100.100.1 10.100.100.2 Up      0  1 FF      3       - to-PE1
Total 1 displayed, Up 1, Down 0
```

```
Transit RSVP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

```
<?xml version="1.0"?>
<jppc:section conf="rsvp.conf" mode="rsvp_before" name="check_show_rsvp_sessions"
target="10.233.255.181" ts="2014-02-25T06:13:10+11:00" xmlns:jppc="http://xml.
juniper.net/jppc">
  <rsvp-session-data>
    <session-type>Ingress</session-type>
    <count>1</count>
    <rsvp-session junos:style="brief" xmlns:junos="http://xml.juniper.net/
junos/*/junos">
      <destination-address>10.100.100.2</destination-address>
      <source-address>10.100.100.1</source-address>
```

```

    <lsp-state>Up</lsp-state>
    <route-count>0</route-count>
    <rsb-count>1</rsb-count>
    <resv-style>FF</resv-style>
    <label-in>-</label-in>
    <label-out>3</label-out>
    <name>to-PE2</name>
  </rsvp-session>
  <display-count>1</display-count>
  <up-count>1</up-count>
  <down-count>0</down-count>
</rsvp-session-data>
<rsvp-session-data>
  <session-type>Egress</session-type>
  <count>1</count>
  <rsvp-session junos:style="brief" xmlns:junos="http://xml.juniper.net/
junos/*/junos">
    <destination-address>10.100.100.1</destination-address>
    <source-address>10.100.100.2</source-address>
    <lsp-state>Up</lsp-state>
    <route-count>0</route-count>
    <rsb-count>1</rsb-count>
    <resv-style>FF</resv-style>
    <label-in>3</label-in>
    <label-out>-</label-out>
    <name>to-PE1</name>
  </rsvp-session>
  <display-count>1</display-count>
  <up-count>1</up-count>
  <down-count>0</down-count>
</rsvp-session-data>
<rsvp-session-data>
  <session-type>Transit</session-type>
  <count>0</count>
  <display-count>0</display-count>
  <up-count>0</up-count>
  <down-count>0</down-count>
</rsvp-session-data>
</jppc:section>

```

One thing important to note here is related to the *Transit* RSVP sessions. In case your network topology has many redundant paths where the LSP can pass through, a change in this counter may not indicate a problem. However, a constant change may indicate a problem. For this reason, the test checks if the number changed slightly instead of checking if it remains the same.

Analyzing the XML output identifies the following XPath to build our JSNAP tests:

```

Command: show rsvp session
/rsvp-session-data
/rsvp-session-data/session-type

```



```

/rsvp-session-data/rsvp-session
/rsvp-session-data/rsvp-session/destination-address
/rsvp-session-data/rsvp-session/source-address
/rsvp-session-data/rsvp-session/lsp-state
/rsvp-session-data/rsvp-session/name
/rsvp-session-data/display-count
/rsvp-session-data/up-count
/rsvp-session-data/down-count

```

The XPath paths identified for the RSVP tests are very similar to others already identified in this chapter. The JSNAP test operators will be **no-diff**, **list-not-less**, **list-not-more**, and **delta**. The IDs identified for these XPath paths are **session-type** and **name**.

The other group of RSVP tests is related to the interfaces configured under the RSVP protocol. The test requirements predict no changes in the RSVP topology. In order to check that, the JUNOS operational command **show rsvp interfaces** is used and here are the normal and XML outputs of this command:.

```
juniper@PE1> show rsvp interface
```

RSVP interface: 1 active

Interface	State	Active resv	Subscription	Static BW	Available BW	Reserved BW	Highwater mark
ge-0/0/3.0	Up	1	100%	1000Mbps	1000Mbps	0bps	0bps

```

<?xml version="1.0"?>
<jppc:section conf="rsvp.conf" mode="rsvp_before" name="check_show_rsvp_interface"
target="10.233.248.17" ts="2014-02-26T07:06:46+11:00" xmlns:jppc="http://xml.juniper.
net/jppc">
  <active-count>1</active-count>
  <rsvp-interface junos:style="brief" xmlns:junos="http://xml.juniper.net/
junos/*/junos">
    <interface-name>ge-0/0/3.0</interface-name>
    <index>75</index>
    <rsvp-status>Up</rsvp-status>
    <rsvp-telink>
      <active-reservation>1</active-reservation>
      <subscription>100</subscription>
      <static-bandwidth>1000Mbps</static-bandwidth>
      <available-bandwidth>1000Mbps</available-bandwidth>
      <total-reserved-bandwidth>0bps</total-reserved-bandwidth>
      <high-watermark>0bps</high-watermark>
    </rsvp-telink>
  </rsvp-interface>
</jppc:section>

```

NOTE If you noticed there is no checking on the traffic engineering information here, it's because this is a simple setup of RSVP topology.

In the analysis of the **show rsvp interface** XML output, you can identify the following XPath:

```
Command: show rsvp interface
active-count
rsvp-interface/interface-name
rsvp-interface/rsvp-status
```

For this simple XML output, the ID is **interface-name** and the JSNAP operators to be used on these verifications are **no-diff**, **list-not-more**, and **list-not-less**.

With the XPath, operators, and IDs identified, it's time to write the JSNAP configuration file for the RSVP tests:

```
do {
    check_show_rsvp_sessions;
    check_show_rsvp_interface;
}

check_show_rsvp_sessions {
    command show rsvp session;
    iterate rsvp-session-data {
        id session-type;
        no-diff count {
            info "Checking if the number of RSVP sessions has changed ...";
            err " ERROR: the number of | %s | RSVP sessions has changed from %s to
%s.", $ID.1, $PRE/count, $POST/count;
        }
        no-diff display-count {
            info "Checking if the number of displayed RSVP sessions has changed ...";
            err " ERROR: the number of | %s | displayed RSVP sessions has changed from
%s to %s.", $ID.1, $PRE/display-count, $POST/display-count;
        }
        no-diff up-count {
            info "Checking if the number of active (UP) RSVP sessions has changed
...";
            err " ERROR: the number of | %s | active (UP) RSVP sessions has changed
from %s to %s.", $ID.1, $PRE/up-count, $POST/up-count;
        }
        no-diff down-count {
            info "Checking if the number of inactive (DOWN) RSVP sessions has changed
...";
            err " ERROR: the number of | %s | inactive (DOWN) RSVP sessions has
changed from %s to %s.", $ID.1, $PRE/down-count, $POST/down-count;
        }
    }
    iterate rsvp-session-data/rsvp-session {
        id ../session-type;
        id name;
        no-diff source-address {
            info "Checking the source address of the RSVP sessions ...";
            err " ERROR: the | %s | RSVP session %s has changed its source address from
%s to %s", $ID.1, $ID.2, $PRE/source-address, $POST/source-address;
```

```

    }
    no-diff destination-address {
        info "Checking the destination address of the RSVP sessions ...";
        err " ERROR: the | %s | RSVP session %s has changed its destination
address from %s to %s", $ID.1, $ID.2, $PRE/destination-address, $POST/destination-
address;
    }
    no-diff name {
        info "Checking the RSVP session names ...";
        err " ERROR: the | %s | RSVP session %s has changed its name from %s to
%s.", $ID.1, $ID.2, $PRE/name, $POST/name;
    }
    no-diff lsp-state {
        info "Checking the RSVP session state ...";
        err " ERROR: the | %s | RSVP session %s has changed its state from %s to
%s.", $ID.1, $ID.2, $PRE/lsp-state, $POST/lsp-state;
    }
    list-not-less name {
        info "Checking for missing RSVP sessions ...";
        err " ERROR: the | %s | RSVP session %s has gone missing.", $ID.1, $ID.2;
    }
    list-not-more name {
        info "Checking for new RSVP sessions ...";
        err " ERROR: the | %s | RSVP session %s was not present before.", $ID.1,
$ID.2;
    }
}
}

check_show_rsvp_interface {
    command show rsvp interface;
    iterate . {
        no-diff active-count {
            info "Checking the number of active RSVP interfaces ...";
            err " ERROR: the number of active RSVP interfaces has changed from %s to
%s.", $PRE/active-count $POST/active-count;
        }
    }
    iterate rsvp-interface {
        id interface-name;
        no-diff interface-name {
            info "Checking if the name of the RSVP interface has changed ...";
            err " ERROR: the name of the RSVP interface %s has changed from %s to %s.",
$ID.1, $PRE/interface-name, $POST/interface-name;
        }
        no-diff rsvp-status {
            info "Checking the RSVP status for each interface ...";
            err " ERROR: the status of the RSVP interface %s has changed from %s to
%s.", $ID.1, $PRE/rsvp-status, $POST/rsvp-status;
        }
        list-not-less interface-name {
            info "Checking for missing RSVP interfaces ...";
            err " ERROR: the RSVP interface %s has gone missing.", $ID.1;
        }
    }
}

```

```

list-not-more interface-name {
    info "Checking for new RSVP interfaces ...";
    err " ERROR: the RSVP interface %s was not present before.", $ID.1;
}
}
}

```

Okay that's the completion of the configuration file for the RSVP network verification tests, and there are only the MPLS tests left, which are covered next. Almost there.

Developing the JSNAP Configuration File – MPLS Network Verifications

The MPLS group is the last group of network verifications developed in this book, and in the beginning of this chapter, four items were identified to be tested under the MPLS verification. The first MPLS test is related to the interfaces and uses **show mpls interface**. The remaining MPLS tests are related to MPLS LSPs and use the **show mpls lsp extensive** command.

Here are the normal and XML outputs for the **show mpls interface** command:

```

juniper@PE1> show mpls interface
Interface      State      Administrative groups (x: extended)
ge-0/0/3.0     Up         <none>

```

```

<?xml version="1.0"?>
<jppc:section conf="mpls.conf" mode="mpls_before" name="check_show_mpls_interface"
target="10.233.255.192" ts="2014-02-28T07:23:10+11:00" xmlns:jppc="http://xml.
juniper.net/jppc">
  <mpls-interface>
    <interface-name>ge-0/0/3.0</interface-name>
    <mpls-interface-state>Up</mpls-interface-state>
    <no-group-flag/>
  </mpls-interface>
</jppc:section>

```

The following XPath paths present the normal and XML outputs for the **show mpls lsp extensive** command:

```

juniper@PE1> show mpls lsp extensive
Ingress LSP: 1 sessions

10.100.100.2
  From: 10.100.100.1, State: Up, ActiveRoute: 0, LSPname: to-PE2
  ActivePath: (primary)
  LSPTYPE: Static Configured, Penultimate hop popping
  LoadBalance: Random
  Encoding type: Packet, Switching type: Packet, GPID: IPv4
  *Primary                               State: Up

```

```

Priorities: 7 0
SmartOptimizeTimer: 180
Computed ERO (S [L] denotes strict [loose] hops): (CSPF metric: 10)
10.1.1.2 S
  Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W 8=Node 10=SoftPreempt
20=Node-ID):
    10.1.1.2
      5 Feb 28 13:10:09.269 Selected as active path
      4 Feb 28 13:10:09.268 Record Route: 10.1.1.2
      3 Feb 28 13:10:09.265 Up
      2 Feb 28 13:10:09.237 Originate Call
      1 Feb 28 13:10:09.237 CSPF: computation result accepted 10.1.1.2
Created: Fri Feb 28 13:09:10 2014
Total 1 displayed, Up 1, Down 0

```

Egress LSP: 1 sessions

```

10.100.100.1
  From: 10.100.100.2, LSPstate: Up, ActiveRoute: 0
  LSPname: to-PE1, LSPpath: Primary
  Suggested label received: -, Suggested label sent: -
  Recovery label received: -, Recovery label sent: -
  Resv style: 1 FF, Label in: 3, Label out: -
  Time left: 121, Since: Fri Feb 28 13:10:08 2014
  Tspec: rate 0bps size 0bps peak Infbps m 20 M 1500
  Port number: sender 1 receiver 28834 protocol 0
  PATH rcvfrom: 10.1.1.2 (ge-0/0/3.0) 89 pkts
  Adspec: received MTU 1500
  PATH sentto: localclient
  RESV rcvfrom: localclient
  Record route: 10.1.1.2 <self>
Total 1 displayed, Up 1, Down 0

```

Transit LSP: 0 sessions

Total 0 displayed, Up 0, Down 0

juniper@PE1>

```

<?xml version="1.0"?>
<jppc:section conf="mpls.conf" mode="mpls_before" name="check_show_mpls_lsp_
extensive" target="10.233.255.181" ts="2014-03-01T09:02:55+11:00" xmlns:jppc="http://
xml.juniper.net/jppc">
  <rsvp-session-data>
    <session-type>Ingress</session-type>
    <count>1</count>
    <rsvp-session junos:style="detail" xmlns:junos="http://xml.juniper.net/
junos/*/junos">
      <mpls-lsp>
        <destination-address>10.100.100.2</destination-address>
        <source-address>10.100.100.1</source-address>
        <lsp-state>Up</lsp-state>
        <route-count>0</route-count>
        <name>to-PE2</name>
      </mpls-lsp>
    </rsvp-session>
  </rsvp-session-data>
</jppc:section>

```

```

    <lsp-description/>
    <active-path>(primary)</active-path>
    <lsp-type>Static Configured</lsp-type>
    <egress-label-operation>Penultimate hop popping</egress-label-
operation>
    <load-balance>random</load-balance>
    <mpls-lsp-attributes>
      <encoding-type>Packet</encoding-type>
      <switching-type>Packet</switching-type>
      <gpId>IPv4</gpId>
    </mpls-lsp-attributes>
    <mpls-lsp-path>
      <title>Primary</title>
      <name/>
      <path-active/>
      <path-state>Up</path-state>
      <setup-priority>7</setup-priority>
      <hold-priority>0</hold-priority>
      <smart-optimize-timer>180</smart-optimize-timer>
      <cspf-status>Computed ERO (S [L] denotes strict [loose] hops): (CSPF
metric: 10)</cspf-status>
      <explicit-route heading="          ">
        <address>10.1.1.2</address>
        <explicit-route-type>S</explicit-route-type>
      </explicit-route>
      <received-rro>Received RRO (ProtectionFlag 1=Available 2=InUse 4=B/W
8=Node 10=SoftPreempt 20=Node-ID):
        10.1.1.2</received-rro>
      <path-history>
        <sequence-number>5</sequence-number>
        <time>Feb 28 13:10:09.269</time>
        <log>Selected as active path</log>
      </path-history>
      <path-history>
        <sequence-number>4</sequence-number>
        <time>Feb 28 13:10:09.268</time>
        <log>Record Route:</log>
        <route>10.1.1.2</route>
      </path-history>
      <path-history>
        <sequence-number>3</sequence-number>
        <time>Feb 28 13:10:09.265</time>
        <log>Up</log>
        <route/>
      </path-history>
      <path-history>
        <sequence-number>2</sequence-number>
        <time>Feb 28 13:10:09.237</time>
        <log>Originate Call</log>
        <route/>
      </path-history>
      <path-history>
        <sequence-number>1</sequence-number>

```

```
<time>Feb 28 13:10:09.237</time>  
<log>CSPF: computation result accepted</log>  
<route>10.1.1.2</route>  
</path-history>  
</mpls-lsp-path>  
<lsp-creation-time>Fri Feb 28 13:09:10 2014</lsp-creation-time>  
</mpls-lsp>  
</rsvp-session>  
<display-count>1</display-count>  
<up-count>1</up-count>  
<down-count>0</down-count>  
</rsvp-session-data>  
<rsvp-session-data>  
  <session-type>Egress</session-type>  
  <count>1</count>  
</rsvp-session junos:style="detail" xmlns:junos="http://xml.juniper.net/  
junos/*/junos">  
  <destination-address>10.100.100.1</destination-address>  
  <source-address>10.100.100.2</source-address>  
  <lsp-state>Up</lsp-state>  
  <route-count>0</route-count>  
  <name>to-PE1</name>  
  <lsp-path-type>Primary</lsp-path-type>  
  <suggested-label-in>-</suggested-label-in>  
  <suggested-label-out>-</suggested-label-out>  
  <recovery-label-in>-</recovery-label-in>  
  <recovery-label-out>-</recovery-label-out>  
  <rsb-count>1</rsb-count>  
  <resv-style>FF</resv-style>  
  <label-in>3</label-in>  
  <label-out>-</label-out>  
  <psb-lifetime>150</psb-lifetime>  
  <psb-creation-time>Fri Feb 28 13:10:08 2014</psb-creation-time>  
  <sender-tspec>rate 0bps size 0bps peak Infbps m 20 M 1500</sender-tspec>  
  <lsp-id>1</lsp-id>  
  <tunnel-id>28834</tunnel-id>  
  <proto-id>0</proto-id>  
  <lsp-attribute-flags/>  
  <packet-information heading=" PATH">  
    <previous-hop>10.1.1.2</previous-hop>  
    <interface-name>ge-0/0/3.0</interface-name>  
    <count>73</count>  
  </packet-information>  
  <adspec>received MTU 1500 </adspec>  
  <packet-information heading=" PATH">  
    <next-hop>localclient</next-hop>  
  </packet-information>  
  <packet-information heading=" RESV">  
    <previous-hop>localclient</previous-hop>  
  </packet-information>  
  <record-route heading=" Record route: ">  
    <address>10.1.1.2</address>  
    <self/>  
  </record-route>
```

```

    </rsvp-session>
    <display-count>1</display-count>
    <up-count>1</up-count>
    <down-count>0</down-count>
  </rsvp-session-data>
  <rsvp-session-data>
    <session-type>Transit</session-type>
    <count>0</count>
    <display-count>0</display-count>
    <up-count>0</up-count>
    <down-count>0</down-count>
  </rsvp-session-data>
</jppc:section>

```

The following XPathS can be identified by analyzing the first output:

```

Command: show mpls interface
/mpls-interface
/mpls-interface/mpls-interface-state

```

The ID identified for this XML output is **interface-name**.

The first XML output is very simple, and the JSNAP operators to be used to analyze it are: **no-diff**, **list-not-less**, and **list-not-more**.

The second output of the MPLS verifications, the **show mpls lsp extensive** outputs, however, are a bit more complex. If you look at the XML elements boldfaced, you will find that their analysis can be done in the same way all other previous analyses have been done throughout this book. The following Xpaths can be identified:

```

Command: show mpls lsp extensive
/rsvp-session-data
/rsvp-session-data/session-type
/rsvp-session-data/count
/rsvp-session-data/display-count
/rsvp-session-data/up-count
/rsvp-session-data/down-count
/rsvp-session-data/rsvp-session
/rsvp-session-data/rsvp-session/mpls-lsp
/rsvp-session-data/rsvp-session/mpls-lsp/destination-address
/rsvp-session-data/rsvp-session/mpls-lsp/source-address
/rsvp-session-data/rsvp-session/mpls-lsp/name
/rsvp-session-data/rsvp-session/mpls-lsp/lsp-description
/rsvp-session-data/rsvp-session/mpls-lsp/active-path
/rsvp-session-data/rsvp-session/mpls-lsp/mpls-lsp-path
/rsvp-session-data/rsvp-session/mpls-lsp/mpls-lsp-path/title
/rsvp-session-data/rsvp-session/mpls-lsp/mpls-lsp-path/name
/rsvp-session-data/rsvp-session/mpls-lsp/mpls-lsp-path/path-active
/rsvp-session-data/rsvp-session/mpls-lsp/mpls-lsp-path/path-state

```

The IDs identified for these XPathS are **session-type** and **name**. It's important to note the role the **session-type** ID is playing as there are three types of MPLS LSPs: ingress, egress, and transit.

After the identification of the XPaths and their IDs, it's time to develop the JSNAP configuration file for the MPLS network verifications. Here's the completed configuration file for the JSNAP tests.

```
do {
    check_show_mpls_interface;
    check_show_mpls_lsp_extensive;
}

check_show_mpls_interface {
    command show mpls interface;
    iterate mpls-interface {
        id interface-name;
        no-diff interface-name {
            info "Checking if there are changes in the name of the MPLS interfaces
...";
            err " ERROR: the interface %s has changed its name from %s to %s.", $PRE/
interface-name, $POST/interface-name;
        }
        no-diff mpls-interface-state {
            info "Checking the MPLS interface state ...";
            err " ERROR: the interface %s has changed its state from %s to %s.",
$ID.1, $PRE/mps-interface-state, $POST/mps-interface-state;
        }
        list-not-less interface-name {
            info "Checking for missing MPLS interfaces ...";
            err " ERROR: the interface %s has gone missing.", $ID.1;
        }
        list-not-more interface-name {
            info "Checking for new MPLS interfaces ...";
            err " ERROR: the interface %s was not present before.", $ID.1;
        }
    }
}

check_show_mpls_lsp_extensive {
    command show mpls lsp extensive;
    iterate rsvp-session-data/rsvp-session/mps-lsp {
        id ../../session-type;
        id name;
        no-diff destination-address {
            info "Checking if the LSP has changed its destination address ...";
            err " ERROR: the %s LSP %s has changed its destination address from %s to
%s.", $ID.1, $ID.2, $PRE/destination-address, $POST/destination-address;
        }
        no-diff source-address {
            info "Checking if the LSP has changed its source address ...";
            err " ERROR: the %s LSP %s has changed its source address from %s to %s.",
$ID.1, $ID.2, $PRE/source-address, $POST/source-address;
        }
        no-diff lsp-state {
            info "Checking if the LSP state has changed ...";
            err " ERROR: the %s LSP %s has changed its state from %s to %s.", $ID.1,
$ID.2, $PRE/lsp-state, $POST/lsp-state;
        }
    }
}
```

```

    }
    no-diff name {
        info "Checking if the LSP name has changed ...";
        err " ERROR: the %s LSP %s has changed its name from %s to %s.", $ID.1,
$ID.2, $PRE/name, $POST/name;
    }
    list-not-less name {
        info "Checking for missing LSPs ...";
        err " ERROR: the %s LSP %s has gone missing.", $ID.1, $ID.2;
    }
    list-not-more name {
        info "Checking for new LSPs ...";
        err " ERROR: the %s LSP %s was not present before.", $ID.1, $ID.2;
    }
    contains active-path, "(primary)" {
        info "Checking if the PRIMARY path is the active path for the LSP ...";
        err " ERROR: the %s LSP %s is not running on its PRIMARY path. It is
currently running on %s.", $ID.1, $ID.2, $POST/active-path;
    }
}
iterate rsvp-session-data {
    id session-type;
    no-diff count {
        info "Checking the number of LSPs ...";
        err " ERROR: the number of %s LSPs has changed from %s to %s.", $ID.1,
$PRE/count, $POST/count;
    }
    no-diff up-count {
        info "Checking the number of LSP in UP state ...";
        err " ERROR: the number of %s LSPs in UP state has changed from %s to %s.",
$ID.1, $PRE/up-count, $POST/up-count;
    }
    no-diff down-count {
        info "Checking the number of LSP in DOWN state ...";
        err " ERROR: the number of %s LSPs in DOWN state has changed from %s to
%s.", $ID.1, $PRE/down-count, $POST/down-count;
    }
}
}

```

That completes the JSNAP configuration file for the MPLS network verifications, and ends this chapter.

In this chapter, you not only learned a methodology to develop your own JSNAP tests, but you have learned how to use the majority of the JSNAP test operators.

If you have tested each of the JSNAP files produced in this chapter, and noticed few errors here and there, don't worry, because they are intentional and they will be covered in Chapter 4.

Chapter 4

Tips and Tricks

In the previous chapter, you were guided through the development of a JSNAP configuration file. You learned a simple methodology to help you identify the tests you needed and how to write a JSNAP configuration file for those tests.

In this chapter, you will gain from lessons learned from the author's experience using JSNAP for daily activities. These lessons can be useful for you because they can help you reduce the time you spend developing the configuration file.

Tip #1

The first tip is related to how JSNAP modifies the XML outputs collected from the router. If you remember, this has been explained in detail at the beginning of Chapter 3. In summary, JSNAP removes the first XML element of the output and replaces it with JSNAP's XML header. This is crucial information because it affects the XPath expressions you are developing for each test. Therefore, the tip is to create a simple JSNAP configuration file containing only the commands (not JSNAP tests) in order to have sample outputs to start with the configuration file development.

As an example for this tip, the following is a JSNAP configuration file containing all the commands used in this book:

```
do {  
    check_bgp_summary;  
    check_show_chassis_hardware;  
    check_show_chassis_fpc;  
    check_show_chassis_fpc_pic_status;
```

```
        check_show_isis_interface;
        check_show_isis_adjacency;
    check_show_interfaces_terse;
        check_show_mpls_interface;
        check_show_mpls_lsp_extensive;
        check_show_rsvp_sessions;
        check_show_rsvp_interface;
}

check_bgp_summary {
    command show bgp summary;
}

check_show_chassis_hardware {
    command show chassis hardware;
}

check_show_chassis_fpc {
    command show chassis fpc;
}

check_show_chassis_fpc_pic_status {
    command show chassis fpc pic-status;
}

check_show_isis_interface {
    command show isis interface;
}

check_show_isis_adjacency {
    command show isis adjacency;
}

check_show_interfaces_terse {
    command show interfaces terse;
}

check_show_mpls_interface {
    command show mpls interface;
}

check_show_mpls_lsp_extensive {
    command show mpls lsp extensive;
}

check_show_rsvp_sessions {
    command show rsvp session;
}

check_show_rsvp_interface {
    command show rsvp interface;
}
```

This may sound trivial for users who are familiar with JSNAP and XML, but it will surely help those who are new to both.

TRICK #1

As pointed out in Chapter 3, during the development of the configuration file for the IGP and Chassis verifications, a few tests in the configuration will fail if you use the configuration file developed in Chapter 3. Let's get to the bottom of that problem.

Here is the verification of the two JSNAP snapshots: *chassis_before* and *chassis_after*:

```
dmontagner@querencia:~/jsnap$ jsnap --check chassis_before,chassis_after -t kenny-au
-l juniper chassis.conf
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: kenny-au
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-----
CHECKING SECTION: check_show_chassis_hardware
-----
+ TEST PASSED: "Checking chassis modules names ..."
- TEST FAILED: "Checking chassis module version ..."

    ERROR: the version of the module FPC 0 has changed from  to .

    ERROR: the version of the module FPC 1 has changed from  to .

    ERROR: the version of the module Fan Tray has changed from  to .
- TEST FAILED: "Checking chassis module part-number ..."

    ERROR: the part-number of module FPC 0 has changed from  to .

    ERROR: the part-number of module FPC 1 has changed from  to .

    ERROR: the part-number of module Fan Tray has changed from  to .
- TEST FAILED: "Checking chassis module serial-number ..."

    ERROR: the serial-number of module FPC 0 has changed from  to .

    ERROR: the serial-number of module FPC 1 has changed from  to .

    ERROR: the serial-number of module Fan Tray has changed from  to .

+ TEST PASSED: "Checking chassis module description ..."
- TEST FAILED: "Checking chassis module model-number ..."
```

ERROR: the model-number of module Display has changed from to .

ERROR: the model-number of module FPC 0 has changed from to .

ERROR: the model-number of module FPC 1 has changed from to .

```
+ TEST PASSED: "Checking for missing modules ..."
+ TEST PASSED: "Checking for new modules ..."
+ TEST PASSED: "Checking chassis sub-modules names ..."
+ TEST PASSED: "Checking chassis sub-module version ..."
+ TEST PASSED: "Checking chassis sub-module part-number ..."
+ TEST PASSED: "Checking chassis sub-module serial-number ..."
+ TEST PASSED: "Checking chassis sub-module description ..."
+ TEST PASSED: "Checking chassis sub-module model-number ..."
+ TEST PASSED: "Checking for missing sub-modules ..."
+ TEST PASSED: "Checking for new sub-modules ..."
-----
CHECKING SECTION: check_show_chassis_fpc
-----
+ TEST PASSED: "Checking FPC state ..."
+ TEST PASSED: "Checking for missing FPCs ..."
+ TEST PASSED: "Checking for new FPCs ..."
+ TEST PASSED: "Checking if the temperature of the FPCs is below 55-Celsius degrees
(131-Fahrenheit)."
```

```
+ TEST PASSED: "Checking if the CPU utilisation of the FPCs is below 70%."
+ TEST PASSED: "Checking if the memory' size of the FPCs remains unchanged."
+ TEST PASSED: "Checking if the memory heap utilisation of the FPCs is within the range
of 20% ~ 70%."
+ TEST PASSED: "Checking if the memory heap utilisation of the FPCs has changed more
than 15%."
+ TEST PASSED: "Checking if the memory buffer utilisation of the FPCs is within the
range of 20% ~ 70%."
+ TEST PASSED: "Checking if the memory buffer utilisation of the FPCs has changed more
than 15%."
-----
CHECKING SECTION: check_show_chassis_fpc_pic_status
-----
+ TEST PASSED: "Checking the PIC types ..."
+ TEST PASSED: "Checking the PIC state ..."
+ TEST PASSED: "Checking for missing PICs ..."
+ TEST PASSED: "Checking for new PICs ..."
dmontagner@querencia:~/jsnap$
```

Before continuing, let's note that these outputs are based on a M10 router, so what you see failing in these tests may not fail in other routers (for example, the MX Series or T Series).

There are four tests that have failed: module *version*, module *part-number*, module *serial-number*, and module *model-number*. Inspecting the output you can see that the error message shows the values of the two snapshots for all verifications that failed as *empty*. You may say to yourself: *Empty before and empty after is exactly the same*

thing, so it shouldn't fail. That's correct. But in order to understand what is happening here, let's review the **no-diff** JSNAP Test Operators documentation available at http://www.juniper.net/techpubs/en_US/junos-snapshot1.0/topics/reference/scripting/automation-junos-snapshot-operator-no-diff.html.

The description of the test operator says: “*Junos Snapshot Administrator test operator compares specified data elements that are present in both the first and second snapshot collections and verifies that the value is the same.*” In other words, the XPathS being compared using the **no-diff** operator must exist in both snapshots. So, how can you avoid this type of error?

Starting with JSNAP 1.0.5, there are three new test operators: **no-diff-in**, **is-lt-in**, and **is-gt-in**. The **in** suffix in the name of these operators stands for *ignore null*. These new operators will *not* fail the tests if both XPathS *do not exist* in *both* snapshots. The reason for creating new operators instead of modifying the behavior of the existing ones is to preserve the original behavior of the operators and to avoid impact on systems that are using JSNAP today.

Replacing the **no-diff** test operator with **no-diff-in** for those four tests that have failed will solve the problem and it will still check those four XPathS when they are present. The same applies for tests using **is-lt** and **is-gt**. Moreover, the tests executed over the snapshots of **show isis interface** may fail for the same reason when testing the XPath **isis-interface-state-one** and **isis-interface-state-two**, depending on your ISIS configuration.

TIP #2

This next tip is definitely not documented anywhere in terms of using it as presented here. It relies on the debug capabilities provided by *juise* and requires a bit of SLAX knowledge, so apologies in advance in case this tip gets too geeky.

If you're having some difficulties in finding the correct XPathS for your first JSNAP configuration files, and if you have a bit of programming knowledge, it will be easy for you.

NOTE

For the sake of demonstration, let's assume the **no-diff-in** operator does not exist and you want to figure out why the chassis verifications are failing.

First, you have to instruct *juise* to drop into the debugger as you invoke JSNAP. This is done by setting an environmental variable named *JDB* in your shell with the value *-d*. The following shows you how to use the *juise* debugger to troubleshoot JSNAP:

```

dmontagner@querencia:~/jsnap$ export JDB=-d
dmontagner@querencia:~/jsnap$ echo $JDB
-d

dmontagner@querencia:~/jsnap$ jsnap --check chassis_before,chassis_after -t kenny-au
-l juniper chassis.conf
sdb: The SLAX Debugger (version 0.12.3)
Type 'help' for help
(sdb)

```

Now, a bit of explanation about the JSNAP source code. All JSNAP operators are defined in the **jppc-tests.slax** source file. Therefore, any type of debug to locate problems related to XPaths used in the configuration file must be done at this source file. Here's an example of the execution flow (a.k.a. *stack trace*) for the **no-diff** operator:

```

(sdb) w
#0 match / at jppc-exec.slax:67
#1 template do_check_target()
#2 template do_check_cmd()
#3 template do_check_select()
#4 template do_cmd_test()
  $cmd-ns = [node-set] (1) <check_show_chassis_hardware> ....
  $check-ns = [node-set] (1) <iterate> ....
  $test-ns = [node-set] (1) <no-diff> ....
  $pre-ns = [node-set] (8) <chassis-module> ....
  $pre-iddb = [node-set] (8) <id> ....
  $post-ns = [node-set] (8) <chassis-module> ....
  $post-iddb = [node-set] (8) <id> ....
(sdb)

```

The **no-diff** operator starts in line 132 of the **jppc-tests.slax** source file (JSNAP 1.0.5 version). To list the source code from inside the debugger, use the command **l <NUMBER>**. Here's a snapshot of the no-diff operator source code viewed from the debugger's CLI:

```

(sdb) l 132
jppc-tests.slax:132: /* ##### */
jppc-tests.slax:133: /* ##### */
jppc-tests.slax:134: /* */
jppc-tests.slax:135: /*          no-diff          */
jppc-tests.slax:136: /* */
jppc-tests.slax:137: /* ##### */
jppc-tests.slax:138: /* ##### */
jppc-tests.slax:139:
jppc-tests.slax:140: <func:function name="jppc:EXEC_TEST_no-diff">
jppc-tests.slax:141: {
jppc-tests.slax:142:   param $cmd-ns;      /* entire section block */
jppc-tests.slax:143:   param $check-ns;    /* this collection within section */
(sdb) l 143
jppc-tests.slax:143:   param $check-ns; /* this collection within section */
jppc-tests.slax:144:   param $test-ns;  /* this test within dataset */
jppc-tests.slax:145:   param $pre-ns;   /* precheck node-set for collection */
jppc-tests.slax:146:   param $pre-iddb; /* precheck id database */

```



```

jppc-tests.slax:147: param $post-ns;    /* postcheck node-set for collection */
jppc-tests.slax:148: param $post-iddb; /* postcheck id database */
jppc-tests.slax:149:
jppc-tests.slax:150: if( $test-ns/count ) {
jppc-tests.slax:151:     /* ----- */
jppc-tests.slax:152:     /* checking to see if the count of the      */
jppc-tests.slax:153:     /* element(s) has changed ... this really is */
jppc-tests.slax:154:     /* only used with $check-ns/listof          */
(sdb)

```

The next step is setting a *breakpoint* at line 188 in the **jppc-tests.slax** source file. This line contains the following expression:

```
if( dyn:evaluate($boolexp) == false() ) {
```

This expression is responsible for executing the **no-diff** test you have defined in the configuration file. Whenever the execution reaches this line, it will stop allowing you to inspect the parameters being used in the test. To set a *breakpoint*, you have to use the command below:

```

(sdb) b jppc-tests.slax:188
Breakpoint 1 at file /usr/jawa/jsnap/jppc-tests.slax, line 188
(sdb)

```

Once the *breakpoint* is defined, you have to instruct the debugger to continue the execution of the program. This is done by entering the command **c**, as presented here:

```

(sdb) c
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: kenny-au
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-----
CHECKING SECTION: check_show_chassis_hardware
-----
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188: if( dyn:evaluate($boolexp) == false() ) {
(sdb)

```

As you can see in the output, once the execution flow reached the line 188, it suspended the execution yet kept all memory and program states. From this point you can start the inspection of the parameters, but first a brief explanation of the variables:

- **\$pre_ns**: Contains the XML *nodeset* from the pre-snapshot that will be used to compare against the POST snapshot.
- **\$post_ns**: Contains the XML *nodeset* from the post-snapshot that will be used to compare against the PRE snapshot.
- **\$xpath**: The XML XPath inside the *nodeset* that will be checked in both PRE and POST snapshot.

- **\$boolexp**: The boolean expression that will be evaluated to execute the test.

Now, let's inspect the variables:

```
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Midplane</name>
<version>REV 03</version>
<part-number>710-001950</part-number>
<serial-number>HF0237</serial-number>
<description>M10 Backplane</description>
<model-number>CHAS-MP-M10-S</model-number>
</chassis-module>
```

```
(sdb) p $post_ns
[node-set] (1)
<chassis-module>
<name>Midplane</name>
<version>REV 03</version>
<part-number>710-001950</part-number>
<serial-number>HF0237</serial-number>
<description>M10 Backplane</description>
<model-number>CHAS-MP-M10-S</model-number>
</chassis-module>
```

```
(sdb) p $xpath
[string] "/name"
```

```
(sdb) p $boolexp
[string] "boolean($pre_ns/name = $post_ns/name)"
```

```
(sdb)
```

Let's explain what's going on in this inspection. The program is executing the comparison of the **chassis-module Midplane**. The exact part of the *nodeset* that will be compared can be identified by the **\$xpath** variable, which in this case contains the value of the element **/name**. The value of this variable is **Midplane** in both pre- and post-snapshots, and our test was configured to detect if they were different. The test that will be executed can be seen in the **\$boolexp** (before its evaluation). The evaluation of the variable **\$boolexp** will produce the following logical expression:

```
boolean("Midplane" = "Midplane")
```

If the result of this expression is *true*, then our JSNAP test will not fail. If the result is *false*, then the JSNAP test will fail and present the error message defined in the configuration file. This shows you how to check what the value of the evaluation of this logical expression will be:

```
(sdb) p dyn:evaluate($boolexp)
[boolean] true
```

```
(sdb)
```

As you can see, the result was *true*, exactly as expected. Now you may be asking: Fine, but I wanted to check the failed tests, i.e., the version test against the FPC component.

Since you already have the *breakpoint* configured, you just need to continue the execution of the program until it reaches the test and the component it is looking for: **no-diff version** test executed against the FPC component. To achieve this, you need to use the command **c** in the debugger's CLI and each time the execution is suspended, you will inspect the variable **\$pre_ns**. Here's how to execute this procedure:

```
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Midplane</name>
<version>REV 03</version>
<part-number>710-001950</part-number>
<serial-number>HF0237</serial-number>
<description>M10 Backplane</description>
<model-number>CHAS-MP-M10-S</model-number>
</chassis-module>

(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Power Supply A</name>
<version>Rev 04</version>
<part-number>740-002497</part-number>
<serial-number>LL13962</serial-number>
<description>AC Power Supply</description>
<model-number>PWR-M10-M5-AC-S</model-number>
</chassis-module>

(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Display</name>
<version>REV 04</version>
<part-number>710-001995</part-number>
<serial-number>HE6561</serial-number>
<description>M10 Display Board</description>
</chassis-module>

(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
```

```
<name>Routing Engine</name>
<version>REV 08</version>
<part-number>740-003877</part-number>
<serial-number>9000006124</serial-number>
<description>RE-2.0</description>
<model-number>RE-333-256-S</model-number>
</chassis-module>
```

```
(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>FEB</name>
<version>REV 07</version>
<part-number>710-003310</part-number>
<serial-number>HH4020</serial-number>
<description>E-FEB</description>
<model-number>FEB-M10-E-S</model-number>
</chassis-module>
```

```
(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>FPC 0</name>
<description>E-FPC</description>
<chassis-sub-module>
<name>PIC 0</name>
<version>REV 02</version>
<part-number>750-003072</part-number>
<serial-number>HC4419</serial-number>
<description>1x OC-48 SONET, SMSR</description>
<model-number>PE-10C48-SON-SMSR</model-number>
</chassis-sub-module>
</chassis-module>
```

```
(sdb)
```

Once you reach this point, you now have to continue the execution line by line. This is done entering the command **n** in the debugger's CLI:

```
(sdb) n
jppc-tests.slax:174:      var $diffs := { for-each( $common_iddb ) { var $_id = .;
(sdb) n
jppc-tests.slax:181:      var $pre_ns = jppc:id-to-xml( $pre_ns, $_id );
(sdb) n
jppc-utils.slax:188:      var $dyn_xpath = { for-each( $id-ns/id ) {

<.... omitted for brevity ....>
```

```
(sdb) n
jppc-utils.slax:194:      <func:result select="$dynobj">;
```

```
(sdb) n
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $boolexp
[string] "boolean($pre_ns/version = $post_ns/version)"
```

<... omitted for brevity ...>

```
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Midplane</name>
<version>REV 03</version>
<part-number>710-001950</part-number>
<serial-number>HF0237</serial-number>
<description>M10 Backplane</description>
<model-number>CHAS-MP-M10-S</model-number>
</chassis-module>
```

```
(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Power Supply A</name>
<version>Rev 04</version>
<part-number>740-002497</part-number>
<serial-number>LL13962</serial-number>
<description>AC Power Supply</description>
<model-number>PWR-M10-M5-AC-S</model-number>
</chassis-module>
```

```
(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Display</name>
<version>REV 04</version>
<part-number>710-001995</part-number>
<serial-number>HE6561</serial-number>
<description>M10 Display Board</description>
</chassis-module>
```

```
(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>Routing Engine</name>
<version>REV 08</version>
<part-number>740-003877</part-number>
```

```
<serial-number>9000006124</serial-number>
<description>RE-2.0</description>
<model-number>RE-333-256-S</model-number>
</chassis-module>
```

```
(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>FEB</name>
<version>REV 07</version>
<part-number>710-003310</part-number>
<serial-number>HH4020</serial-number>
<description>E-FEB</description>
<model-number>FEB-M10-E-S</model-number>
</chassis-module>
```

```
(sdb) c
Reached breakpoint 1, at /usr/jawa/jsnap/jppc-tests.slax:188
jppc-tests.slax:188:      if( dyn:evaluate($boolexp) == false() ) {
(sdb) p $pre_ns
[node-set] (1)
<chassis-module>
<name>FPC 0</name>
<description>E-FPC</description>
<chassis-sub-module>
<name>PIC 0</name>
<version>REV 02</version>
<part-number>750-003072</part-number>
<serial-number>HC4419</serial-number>
<description>1x 0C-48 SONET, SMSR</description>
<model-number>PE-10C48-SON-SMSR</model-number>
</chassis-sub-module>
</chassis-module>
```

```
(sdb) p $post_ns
[node-set] (1)
<chassis-module>
<name>FPC 0</name>
<description>E-FPC</description>
<chassis-sub-module>
<name>PIC 0</name>
<version>REV 02</version>
<part-number>750-003072</part-number>
<serial-number>HC4419</serial-number>
<description>1x 0C-48 SONET, SMSR</description>
<model-number>PE-10C48-SON-SMSR</model-number>
</chassis-sub-module>
</chassis-module>
```

```
(sdb) p $xpath
[string] "/version"
```

```
(sdb) p $boolexp
[string] "boolean($pre_ns/version = $post_ns/version)"
```

```
(sdb) p dyn:evaluate($boolexp)
[boolean] false
```

```
(sdb) p $pre_ns/version
[node-set] (0)
```

```
(sdb) p $post_ns/version
[node-set] (0)
```

```
(sdb)
```

The boldfaced output shows the reason why the test is failing: both **\$pre_ns/version** and **\$post_ns/version** are *NULL*. Remember that the definition of the **no-diff** test operator requires the XPath to be present in both pre- and post-snapshots.

The procedure demonstrated here may seem complicated for someone who does not have programming knowledge, but if you manage to understand it, you will quickly find the reasons why some of your JSNAP tests are *not* working as you expect. Based on experience, you will only need this tip a few times when dealing with complex Junos XML structures. Most of the time, you won't need it.

TRICK #2

In the section on network verification for the router's Interfaces presented in Chapter 3, the scope of the **show interface terse** command was limited to list only GE interfaces. The reason for this is that the router has some special interfaces that are used internally by Junos that may or may not have all the elements that the normal interfaces (**ge**, **xe**, **lo**, **et**, **so**, etc.) have. Sometimes these special interfaces will not have the family address, address, or logical interface elements depending on what you have configured on your router, and the hardware type in use.

The best way to go around this problem is limiting the scope of the **show interfaces** command, guaranteeing that you have all the elements on the XML snapshots of this section. Therefore, no test should fail by the absence of an XPath that is being tested. Here's an example of an interface that can create problems if it is included in our interface verifications:

```
juniper@PE1> show interfaces pimd
Physical interface: pimd, Enabled, Physical link is Up
Interface index: 26, SNMP ifIndex: 11
Type: PIMD, Link-level type: PIM-Decapsulator, MTU: Unlimited, Speed: Unlimited
Device flags   : Present Running
Input packets : 0
```

Output packets: 0

juniper@PE1>

```
juniper@PE1> show interfaces pimd | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/12.3I0/junos">
  <interface-information xmlns="http://xml.juniper.net/junos/12.3I0/junos-interface"
    junos:style="normal">
    <physical-interface>
      <name>pimd</name>
      <admin-status junos:format="Enabled">up</admin-status>
      <oper-status>up</oper-status>
      <local-index>26</local-index>
      <snmp-index>11</snmp-index>
      <if-type>PIMD</if-type>
      <link-level-type>PIM-Decapsulator</link-level-type>
      <mtu>Unlimited</mtu>
      <speed>Unlimited</speed>

      <if-device-flags>
        <ifdf-present/>
        <ifdf-running/>
      </if-device-flags>
      <if-config-flags>
      </if-config-flags>
      <traffic-statistics junos:style="brief">
        <input-packets>0</input-packets>
        <output-packets>0</output-packets>
      </traffic-statistics>
    </physical-interface>
  </interface-information>
  <cli>
    <banner></banner>
  </cli>
</rpc-reply>
```

As you can see, there are many XPath elements being checked under the interface verifications that do not appear in the **pimd** interface. The situation can be different if you are using multicast in your network. And that is why the identification of the tests done in Chapter 3 is something that must be done prior to developing the JSNAP configuration file – in order to avoid surprises later.

TIP #3

This tip is simple but it is something that administrators ask how to do. It relates to invoking JSNAP to collect snapshots in multiple routers. JSNAP requires one CLI command to collect a snapshot from a router. So how do you collect the same snapshot from different routers?

Here is a simplified version of how to run JSNAP across multiple routers with one single command line:


```
#!/bin/bash
ROUTERS=`cat routers.txt`
SNAP=$1

if [ "$SNAP" = "" ]; then
    echo ""
    echo "    ERROR: you must provide a snapshot name !!!"
    echo ""
    echo "    Syntax:"
    echo ""
    echo ""
    echo "    ./collect_snapshots.sh <SNAPSHOT_NAME>"
    echo ""
else
    for router in $ROUTERS; do
        echo "Collecting snapshot '$1' for router $router"
        jsnap --snap $1 -l juniper -p Clouds -t $router mw_checks.conf
    done
fi
```

In this bash script, the list of routers that will be used to collect the snapshot is located inside the **routers.txt** file. This file is a text file that contains one router per line. (If you use names instead of IP addresses, make sure the names can be resolved.)

The result of the execution of the script is showed here:

```
dmontagner@querencia:~/jsnap$ ./collect_snapshots.sh before
Collecting snapshot 'before' for router ce10
Connecting to juniper@ce10 ...
The authenticity of host 'ce10 (10.233.255.196)' can't be established.
ECDSA key fingerprint is 47:d0:79:ec:57:90:ae:39:a6:51:b5:68:bc:a5:f9:56.
Are you sure you want to continue connecting (yes/no)?yes
CONNECTED.
EXEC: 'show bgp summary' ...
SAVE: 'ce10__check_bgp_summary__before.xml' ...
Collecting snapshot 'before' for router ce11
Connecting to juniper@ce11 ...
The authenticity of host 'ce11 (10.233.255.192)' can't be established.
ECDSA key fingerprint is 91:e7:e7:93:71:c0:21:24:96:db:30:93:05:35:36:9d.
Are you sure you want to continue connecting (yes/no)?yes
CONNECTED.
EXEC: 'show bgp summary' ...
SAVE: 'ce11__check_bgp_summary__before.xml' ...
Collecting snapshot 'before' for router ce20
Connecting to juniper@ce20 ...
The authenticity of host 'ce20 (10.233.255.195)' can't be established.
ECDSA key fingerprint is d7:ca:d2:8c:f6:33:9d:71:a3:c9:2b:c5:a4:4d:a5:11.
Are you sure you want to continue connecting (yes/no)?yes
CONNECTED.
EXEC: 'show bgp summary' ...
SAVE: 'ce20__check_bgp_summary__before.xml' ...
Collecting snapshot 'before' for router ce21
```

```

Connecting to juniper@ce21 ...
The authenticity of host 'ce21 (10.233.255.191)' can't be established.
ECDSA key fingerprint is 59:e1:ca:69:52:7b:9c:f2:b1:0b:a4:89:72:c6:5c:6c.
Are you sure you want to continue connecting (yes/no)?yes
CONNECTED.
EXEC: 'show bgp summary' ...
SAVE: 'ce21__check_bgp_summary__before.xml' ...
Collecting snapshot 'before' for router pe1
Connecting to juniper@pe1 ...
The authenticity of host 'pe1 (10.233.255.190)' can't be established.
ECDSA key fingerprint is 7d:c0:35:6c:f1:5f:54:51:c6:8e:5b:76:a2:61:5a:56.
Are you sure you want to continue connecting (yes/no)?yes
CONNECTED.
EXEC: 'show bgp summary' ...
SAVE: 'pe1__check_bgp_summary__before.xml' ...
Collecting snapshot 'before' for router pe2
Connecting to juniper@pe2 ...
The authenticity of host 'pe2 (10.233.255.189)' can't be established.
ECDSA key fingerprint is 6a:32:76:a4:27:b3:ae:6f:01:a4:7f:28:93:fc:6a:e9.
Are you sure you want to continue connecting (yes/no)?yes
CONNECTED.
EXEC: 'show bgp summary' ...
SAVE: 'pe2__check_bgp_summary__before.xml' ...
dmontagner@querencia:~/jsnap$

```

This script can be easily modified to support multiple routers, in case you need to use different JSNAP configuration files for different groups of routers. With another simple modification, the same script can be used to execute the snapshot collection as well as the comparison. Since the focus here is not *bash scripting*, you are charge of any improvements.

One thing to note, however, are the messages about the authenticity of the hosts. That is the standard SSH message displayed every time you want to connect to a device via SSH and your SSH **known_hosts** file does not contain an entry for the device. JSNAP connects to the devices via NETCONF over SSH, so it relies on the SSH client of your operational system. Once the **known_hosts** file has an entry for the device, it won't ask you for confirmation anymore.

There is one way to modify this behavior by instructing the SSH client to automatically add the host entry in the **known_hosts** file. To achieve this, you need to set the parameter in **/etc/ssh/ssh_config** to **no** as shown here:

```
StrictHostKeyChecking no
```

NOTE Do be careful changing this parameter because it can affect all SSH sessions initiated in the host where you made the change. If you use the SSH client from this host to access non-trusted devices, you should keep this parameter set to **yes**.

Tip #4

This tip is related to XML attributes. If you are not familiar with XML and tried to answer the exercise proposed in Chapter 3, you probably had some difficulties trying to figure out how to check if the temperature of the routing engine is higher than 55 °C. The problem of analyzing the routing engine temperature is related to the way the information presented. Let's check the relevant XML output:

```
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/10.4R8/junos">
  <route-engine-information xmlns="http://xml.juniper.net/junos/10.4R8/junos-
chassis">
    <route-engine>
      <slot>0</slot>
      <mastership-state>master</mastership-state>
      <mastership-priority>master</mastership-priority>
      <status>OK</status>
      <temperature junos:celsius="32">32 degrees C / 89 degrees F</temperature>
      <cpu-temperature junos:celsius="30">30 degrees C / 86 degrees F</cpu-
temperature>
      <memory-dram-size>2048</memory-dram-size>
    </route-engine>
  </route-engine-information>
</rpc-reply>

<... omitted for brevity ...>
```

If you use the XPath `/route-engine/temperature`, the value used for comparison will be “32 degrees C / 89 degrees F” rather than just “32”. Therefore, we won't be able to use **is-lt** or **is-gt** operators.

The problem here is that the number is actually presented using a XML attribute. The XPath `/route-engine/temperature` will access the element value. To access the value of the element's attribute, you need the following XPath:

```
/route-engine/temperature/@junos:celsius
```

NOTE The JSNAP configuration file to check if the temperature is higher than 55 °C can be found in Appendix A of this book.

Tip #5

The last item to present is related to XML. When developing the JSNAP configuration file for the BGP verifications in Chapter 3, the iteration to analyze the XPaths **peer-count**, **peer-group-count** and **active-count** uses an XPath not seen before in this book. Here's a snippet of the JSNAP BGP configuration file:

```
check_bgp_summary {
  command show bgp summary;
  iterate . {
    no-diff group-count {
      info "Checking the number of BGP groups ...";
    }
  }
}
```

```

        err "  ERROR: the number of BGP groups has changed from %s to %s.", $PRE/
group-count, $POST/group-count;
    }
    no-diff peer-count {
        info "Checking the number of BGP peers ...";
        err "  ERROR: the number of BGP peers has changed from %s to %s.", $PRE/
peer-count, $POST/peer-count;
    }
    is-gt peer-count, 0 {
        info "Checking if the BGP configuration has at least 1 BGP peer configured
...";
        err "  ERROR: the BGP configuration does not have any peer configured!";
    }
    no-diff down-peer-count {
        info "Checking the number of BGP peers down ...";
        err "  ERROR: the number of BGP peers down has changed from %s to %s.",
$PRE/down-peer-count, $POST/down-peer-count;
    }
}

<... omitted for brevity ...>

```

The XPath that is being iterated is “.”. Before more explanation, let’s first look in the part of the XML output that this XPath will be analyzing:

```

<?xml version="1.0"?>
<jppc:section xmlns:jppc="http://xml.juniper.net/jppc" name="check_bgp_summary"
mode="before" conf="network_verifications.conf" target="pe1" ts="2014-03-
11T06:28:47+11:00">
  <group-count>3</group-count>
  <peer-count>3</peer-count>
  <down-peer-count>0</down-peer-count>
  <bgp-rib xmlns:junos="http://xml.juniper.net/junos/*/junos" junos:style="brief">
<name>inet.0</name>
<total-prefix-count>0</total-prefix-count>

<... omitted for brevity ...>

```

The representation of the XPaths for the elements **group-count**, **peer-count**, and **down-peer-count** is:

```

./group-count
./peer-count
./down-peer-count

```

So the XPath that needs to be iterated to extract the values of these XML elements is “. ”.

Chapter 5

Putting It All Together

This chapter combines all individual JSNAP files developed in Chapter 3 into a single file. It allows the collection of all snapshots required for network verifications tests to be in a single JSNAP execution. The same is true for the snapshot comparison between the pre- and post-snapshots.

MORE? See this book's landing page on <http://www.juniper.net/dayone> for .txt files of the configuration in its entirety.

The Complete JSNAP Configuration

Here's the final and complete JSNAP configuration file for all network verifications tests presented in this book:

```
do {
    check_show_isis_interface;
    check_show_isis_adjacency;
    check_show_interfaces_terse;
    check_show_chassis_hardware;
    check_show_chassis_fpc;
    check_show_chassis_fpc_pic_status;
    check_bgp_summary;
    check_show_rsvp_sessions;
    check_show_rsvp_interface;
    check_show_mpls_interface;
    check_show_mpls_lsp_extensive;
}

check_show_isis_interface {
    command show isis interface;
    iterate isis-interface {
        id ./interface-name;
```

```

        no-diff interface-name {
            info "Checking the ISIS interface names ...";
            err " ERROR: the interface %s has changed its name from %s to %s.", $ID.1,
$PRE/interface-name, $POST/interface-name;
        }
        no-diff circuit-type {
            info "Checking the ISIS circuit type ...";
            err " ERROR: the interface %s has changed its circuit type from %s to
%s.", $ID.1, $PRE/circuit-type, $POST/circuit-type;
        }
        no-diff-in isis-interface-state-one {
            info "Checking the Level 1 interface state ...";
            err " ERROR: the interface %s has changed its Level 1 interface state from
%s to %s.", $ID.1, $PRE/isis-interface-state-one, $POST/isis-interface-state-one;
        }
        no-diff-in isis-interface-state-two {
            info "Checking the Level 2 interface state ...";
            err " ERROR: the interface %s has changed its Level 2 interface state from
%s to %s.", $ID.1, $PRE/isis-interface-state-two, $POST/isis-interface-state-two;
        }
        no-diff metric-one {
            info "Checking the interface Level 1 metric ...";
            err " ERROR: the Level 1 metric for interface %s has changed from %s to
%s.", $ID.1, $PRE/metric-one, $POST/metric-one;
        }
        no-diff metric-two {
            info "Checking the interface Level 2 metric ...";
            err " ERROR: the Level 2 metric for interface %s has changed from %s to
%s.", $ID.1, $PRE/metric-two, $POST/metric-two;
        }
        list-not-less interface-name {
            info "Checking for missing ISIS interfaces ...";
            err " ERROR: the interface %s is missing.", $ID.1;
        }
        list-not-more interface-name {
            info "Checking for new ISIS interfaces ...";
            err " ERROR: the interface %s was not configured before.", $ID.1;
        }
    }
}

check_show_isis_adjacency {
    command show isis adjacency;
    iterate isis-adjacency {
        id ./interface-name;
        no-diff interface-name {
            info "Checking the ISIS interface names ...";
            err " ERROR: the interface %s has changed from %s to %s.", $ID.1, $PRE/
interface-name, $POST/interface-name;
        }
        no-diff system-name {
            info "Checking the ISIS neighbour ...";
            err " ERROR: the ISIS neighbour on interface %s has changed from %s to
%s.", $ID.1, $PRE/system-name, $POST/system-name;
        }
    }
}

```

```

    }
    no-diff level {
        info "Checking the Level of the ISIS adjacency ...";
        err " ERROR: the ISIS Level on interface %s has changed from % to %s.",
$ID.1, $PRE/level, $POST/level;
    }
    no-diff snpa {
        info "Checking the Subnetwork Point of Attachment (SNPA) ...";
        err " ERROR: the SNPA for interface %s has changed from %s to %s.", $ID.1,
$PRE/snpa, $POST/snpa;
    }
    list-not-less {
        info "Checking for missing ISIS adjacencies ...";
        err " ERROR: the ISIS adjacency for interface %s is missing.", $ID.1;
    }
    list-not-more {
        info "Checking for new ISIS adjacencies ...";
        err " ERROR: the ISIS adjacency for interface %s was not configured
before.", $ID.1;
    }
}

}

check_show_interfaces_terse {
    command show interfaces terse;
    iterate physical-interface {
        id ./name;
        no-diff oper-status {
            info "Checking PHY operational status of interfaces ...";
            err " ERROR: the operational status for interface %s has changed
from %s to %s.", $ID.1, $PRE/oper-status, $POST/oper-status;
        }
        no-diff admin-status {
            info "Checking PHY admin status of interfaces ...";
            err " ERROR: the admin status for interface %s has changed from %s
to %s.", $ID.1, $PRE/admin-status, $POST/admin-status;
        }
        list-not-less name {
            info "Checking for missing interfaces at PHY level ...";
            err " ERROR: the interface %s is missing.", $ID.1;
        }
        list-not-more {
            info "Checking for new interfaces at PHY level ...";
            err " ERROR: the interface %s is new.", $ID.1;
        }
    }
}

iterate physical-interface/logical-interface {
    id ./name;
    id ./address-family/address-family-name;
    id ./address-family/interface-address/ifa-local;
    no-diff oper-status {
        info "Checking LOGICAL operational status of interfaces ...";
        err " ERROR: the operational status for interface %s has changed
from %s to %s.", $ID.1, $PRE/oper-status, $POST/oper-status;
    }
}

```

```

    }
    no-diff admin-status {
        info "Checking LOGICAL admin status of interfaces ...";
        err " ERROR: the admin status of interface %s has changed from %s
to %s.", $ID.1, $PRE/admin-status, $POST/admin-status;
    }
    list-not-less {
        info "Checking for missing interfaces at LOGICAL level ...";
        err " ERROR: the interface %s is missing.", $ID.1;
    }
    list-not-more {
        info "Checking for new interfaces at LOGICAL level ...";
        err " ERROR: the interface %s is new.", $ID.1;
    }
    no-diff address-family/address-family-name {
        info "Checking the address family configured in the interfaces
...";
        err " ERROR: the address family for interface %s has changed from
%s to %s.", $ID.1, $PRE/address-family/address-family-name, $POST/address-family/
address-family-name;
    }
    list-not-less address-family/address-family-name {
        info "Checking for missing address family ...";
        err " ERROR: the address family %s is missing on interface %s.",
$ID.2, $ID.1;
    }
    list-not-more address-family/address-family-name {
        info "Checking for new family address ...";
        err " ERROR: the address family %s has been added to the interface
%s.", $ID.2, $ID.1;
    }
    no-diff address-family/interface-address/ifa-local {
        info "Checking the interface address configured under the interface
...";
        err " ERROR: the interface address for interface %s has changed
from %s to %s.", $ID.1, $PRE/address-family/interface-address/ifa-local, $POST/
address-family/interface-address/ifa-local;
    }
    list-not-less address-family/interface-address/ifa-local {
        info "Checking for missing interface address ...";
        err " ERROR: the interface address %s has gone missing on
interface %s.", $ID.3, $ID.1;
    }
    list-not-more address-family/interface-address/ifa-local {
        info "Checking for new interface address ...";
        err " ERROR: the interface address %s has been added to the
interface %s.", $ID.3, $ID.1;
    }
}

check_show_chassis_hardware {
    command show chassis hardware;
    iterate chassis/chassis-module {

```



```

    id name;
    no-diff name {
        info "Checking chassis modules names ...";
        err " ERROR: the module %s has changed from %s to %s.", $ID.1, $PRE/name,
$POST/name;
    }
    no-diff version {
        info "Checking chassis module version ...";
        err " ERROR: the version of the module %s has changed from %s to %s.", $ID.1,
$PRE/version, $POST/version;
    }
    no-diff part-number {
        info "Checking chassis module part-number ...";
        err " ERROR: the part-number of module %s has changed from %s to %s.", $ID.1,
$PRE/part-number, $POST/part-number;
    }
    no-diff serial-number {
        info "Checking chassis module serial-number ...";
        err " ERROR: the serial-number of module %s has changed from %s to %s.",
$ID.1, $PRE/serial-number, $POST/serial-number;
    }
    no-diff description {
        info "Checking chassis module description ...";
        err " ERROR: the description of module %s has changed from %s to %s.", $ID.1,
$PRE/description, $POST/description;
    }
    no-diff model-number {
        info "Checking chassis module model-number ...";
        err " ERROR: the model-number of module %s has changed from %s to %s.", $ID.1,
$PRE/model-number, $POST/model-number;
    }
    list-not-less name {
        info "Checking for missing modules ...";
        err " ERROR: the module %s is missing.", $ID.1;
    }
    list-not-more name {
        info "Checking for new modules ...";
        err " ERROR: the module %s was installed.", $ID.1;
    }
}
iterate chassis/chassis-module/chassis-sub-module {
    id name;
    id ../name;
    no-diff name {
        info "Checking chassis sub-modules names ...";
        err " ERROR: the sub-module %s of module %s has changed from %s to %s.",
$ID.1, $ID.2, $PRE/name, $POST/name;
    }
    no-diff-in version {
        info "Checking chassis sub-module version ...";
        err " ERROR: the version of the sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/name, $POST/name;
    }
    no-diff-in part-number {

```

```

        info "Checking chassis sub-module part-number ...";
        err " ERROR: the part-number of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/part-number, $POST/part-number;
    }
    no-diff-in serial-number {
        info "Checking chassis sub-module serial-number ...";
        err " ERROR: the serial-number of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/serial-number, $POST/serial-number;
    }
    no-diff description {
        info "Checking chassis sub-module description ...";
        err " ERROR: the description of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/description, $POST/description;
    }
    no-diff model-number {
        info "Checking chassis sub-module model-number ...";
        err " ERROR: the model-number of sub-module %s of module %s has changed
from %s to %s.", $ID.1, $ID.2, $PRE/model-number, $POST/model-number;
    }
    list-not-less name {
        info "Checking for missing sub-modules ...";
        err " ERROR: the sub-module %s of module %s is missing.", $ID.1, $ID.2;
    }
    list-not-more name {
        info "Checking for new sub-modules ...";
        err " ERROR: the sub-module %s of module %s was installed.", $ID.1, $ID.2;
    }
}

check_show_chassis_fpc {
    command show chassis fpc;
    iterate fpc {
        id slot;
        no-diff state {
            info "Checking FPC state ...";
            err " ERROR: the FPC %s has changed its state from %s to %s.", $ID.1, $PRE/
state, $POST/state;
        }
        list-not-less {
            info "Checking for missing FPCs ...";
            err " ERROR: the FPC %s is missing.", $ID.1;
        }
        list-not-more {
            info "Checking for new FPCs ...";
            err " ERROR: the FPC %s was installed.", $ID.1;
        }
        is-lt temperature, 55 {
            info "Checking if the temperature of the FPCs is below 55-Celsius degrees
(131-Fahrenheit).";
            err " ERROR: the temperature of FPC %s is %s (before was %s) Celsius
degrees.", $ID.1, $POST/temperature, $PRE/temperature;
        }
        is-lt cpu-total, 70 {

```

```

        info "Checking if the CPU utilisation of the FPCs is below 70%.";
        err " ERROR: the CPU utilisation of FPC %s is %s (before was %s).", $ID.1;
$POST/cpu-total, $PRE/cpu-total;
    }
    in-range memory-heap-utilization, 20, 70 {
        info "Checking if the memory heap utilisation of the FPCs is within the
range of 20% ~ 70%.";
        err " ERROR: the memory heap utilisation of FPC %s is out-of-range (
Before = %s / After = %s ).", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
    delta memory-heap-utilization, 15% {
        info "Checking if the memory heap utilisation of the FPCs has changed more
than 15%.";
        err " ERROR: the memory heap utilisation of the FPC %s has changed from %s
to %s.", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
    in-range memory-buffer-utilization, 20, 70 {
        info "Checking if the memory buffer utilisation of the FPCs is within the
range of 20% ~ 70%.";
        err " ERROR: the memory buffer utilisation of FPC %s is out-of-range (
Before = %s / After = %s ).", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
    delta memory-buffer-utilization, 15% {
        info "Checking if the memory buffer utilisation of the FPCs has changed
more than 15%.";
        err " ERROR: the memory buffer utilisation of the FPC %s has changed from
%s to %s.", $ID.1, $PRE/memory-dram-size, $POST/memory-dram-size;
    }
}

check_show_chassis_fpc_pic_status {
    command show chassis fpc pic-status;
    iterate fpc/pic {
        id pic-slot;
        id ../slot;
        no-diff pic-type {
            info "Checking the PIC types ...";
            err " ERROR: the PIC type of PIC %s of FPC slot %s has changed from %s to
%s.", $ID.1, $ID.2, $PRE/pic-type, $POST/pic-type;
        }
        no-diff pic-state {
            info "Checking the PIC state ...";
            err " ERROR: the state of PIC %s of FPC slot %s has changed from %s to
%s.", $ID.1, $ID.2, $PRE/pic-state, $POST/pic-state;
        }
        list-not-less pic-slot {
            info "Checking for missing PICs ...";
            err " ERROR: the PIC %s of FPC slot %s is missing.", $ID.1, $ID.2;
        }
        list-not-more pic-slot {
            info "Checking for new PICs ...";
            err " ERROR: the PIC %s of FPC slot %s was installed.", $ID.1, $ID.2;
        }
    }
}

```

```

    }
}

check_bgp_summary {
    command show bgp summary;
    iterate . {
        no-diff group-count {
            info "Checking the number of BGP groups ...";
            err " ERROR: the number of BGP groups has changed from %s to %s.", $PRE/
group-count, $POST/group-count;
        }
        no-diff peer-count {
            info "Checking the number of BGP peers ...";
            err " ERROR: the number of BGP peers has changed from %s to %s.", $PRE/
peer-count, $POST/peer-count;
        }
        is-gt peer-count, 1 {
            info "Checking if the BGP configuration has at least 1 BGP peer configured
...";
            err " ERROR: the BGP configuration does not have any peer configured!";
        }
        no-diff down-peer-count {
            info "Checking the number of BGP peers down ...";
            err " ERROR: the number of BGP peers down has changed from %s to %s.", $PRE/
down-peer-count, $POST/down-peer-count;
        }
    }
    iterate bgp-peer {
        id peer-address;
        id peer-as;
        no-diff peer-address {
            info "Checking if the BGP peers addresses are still the same ...";
            err " ERROR: the BGP peer %s (ASN %s) has changed its address from %s to
%s.", $ID.1, $ID.2, $PRE/peer-address, $POST/peer-address;
        }
        no-diff peer-as {
            info "Checking if the BGP peers ASNs are still the same ...";
            err " ERROR: the ASN for the BGP peer %s has changed from %s to %s.", $ID.1.
$PRE/peer-asn, $POST/peer-asn;
        }
        no-diff flap-count {
            info "Checking if the BGP peer has flapped ...";
            err " ERROR: the BGP peer %s (ASN %s) has flapped.", $ID.1, $ID.2;
        }
        is-equal peer-state, "Established" {
            info "Checking if the BGP peers are in Established state ...";
            err " ERROR: the BGP peer %s (ASN %s) is not in Established state.", $ID.1,
$ID.2;
        }
    }
}
iterate bgp-peer/bgp-rib {
    id name;
    id ../peer-address;
    id ../peer-as;
}

```

```

no-diff name {
    info "Checking if the BGP RIB name has changed ...";
    err " ERROR: the RIB %s of BGP peer %s (ASN %s) has changed from %s to %s.",
$ID.1, $ID.2, $ID.3, $PRE/name, $POST/name;
}
list-not-less name {
    info "Checking for missing RIBs ...";
    err " ERROR: the RIB %s for the BGP peer %s (ASN %s) has gone missing.",
$ID.1, $ID.2, $ID.3;
}
list-not-more name {
    info "Checking for new RIBs ...";
    err " ERROR: the RIB %s was not present before for the BGP peer %s (ASN
%s).", $ID.1, $ID.2, $ID.3;
}
delta active-prefix-count, 20% {
    info "Checking if the number of BGP active prefix has changed more than
20%.";
    err " ERROR: the number of BGP active prefixes for RIB %s on the BGP peer %s
(ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1, $ID.2,
$ID.3, $PRE/active-prefix-count, $POST/active-prefix-count;
}
delta received-prefix-count, 20% {
    info "Checking if the number of BGP received prefix has changed more than
20%.";
    err " ERROR: the number of BGP received prefixes for RIB %s on the BGP peer
%s (ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1, $ID.2,
$ID.3, $PRE/received-prefix-count, $POST/received-prefix-count;
}
delta accepted-prefix-count, 20% {
    info "Checking if the number of BGP accepted prefix has changed more than
20%.";
    err " ERROR: the number of BGP accepted prefixes for RIB %s on the BGP peer
%s (ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1, $ID.2,
$ID.3, $PRE/accepted-prefix-count, $POST/accepted-prefix-count;
}
delta suppressed-prefix-count, 20% {
    info "Checking if the number of BGP suppressed prefix has changed more than
20%.";
    err " ERROR: the number of BGP suppressed prefixes for RIB %s on the BGP
peer %s (ASN %s) has changed more than 20 percent (before = %s / after = %s).", $ID.1,
$ID.2, $ID.3, $PRE/suppressed-prefix-count, $POST/suppressed-prefix-count;
}
}

check_show_rsvp_sessions {
    command show rsvp session;
    iterate rsvp-session-data {
        id session-type;
        no-diff count {
            info "Checking if the number of RSVP sessions has changed ...";
            err " ERROR: the number of | %s | RSVP sessions has changed from %s to
%s.", $ID.1, $PRE/count, $POST/count;
        }
    }
}

```

```

    }
    no-diff display-count {
        info "Checking if the number of displayed RSVP sessions has changed ...";
        err " ERROR: the number of | %s | displayed RSVP sessions has changed from
%s to %s.", $ID.1, $PRE/display-count, $POST/display-count;
    }
    no-diff up-count {
        info "Checking if the number of active (UP) RSVP sessions has changed
...";
        err " ERROR: the number of | %s | active (UP) RSVP sessions has changed
from %s to %s.", $ID.1, $PRE/up-count, $POST/up-count;
    }
    no-diff down-count {
        info "Checking if the number of inactive (DOWN) RSVP sessions has changed
...";
        err " ERROR: the number of | %s | inactive (DOWN) RSVP sessions has
changed from %s to %s.", $ID.1, $PRE/down-count, $POST/down-count;
    }
}
iterate rsvp-session-data/rsvp-session {
    id ../session-type;
    id name;
    no-diff source-address {
        info "Checking the source address of the RSVP sessions ...";
        err " ERROR: the | %s | RSVP session %s has changed its source address from
%s to %s", $ID.1, $ID.2, $PRE/source-address, $POST/source-address;
    }
    no-diff destination-address {
        info "Checking the destination address of the RSVP sessions ...";
        err " ERROR: the | %s | RSVP session %s has changed its destination
address from %s to %s", $ID.1, $ID.2, $PRE/destination-address, $POST/destination-
address;
    }
    no-diff name {
        info "Checking the RSVP session names ...";
        err " ERROR: the | %s | RSVP session %s has changed its name from %s to
%s.", $ID.1, $ID.2, $PRE/name, $POST/name;
    }
    no-diff lsp-state {
        info "Checking the RSVP session state ...";
        err " ERROR: the | %s | RSVP session %s has changed its state from %s to
%s.", $ID.1, $ID.2, $PRE/lsp-state, $POST/lsp-state;
    }
    list-not-less name {
        info "Checking for missing RSVP sessions ...";
        err " ERROR: the | %s | RSVP session %s has gone missing.", $ID.1, $ID.2;
    }
    list-not-more name {
        info "Checking for new RSVP sessions ...";
        err " ERROR: the | %s | RSVP session %s was not present before.", $ID.1,
$ID.2;
    }
}
}
}

```

```

check_show_rsvp_interface {
    command show rsvp interface;
    iterate . {
        no-diff active-count {
            info "Checking the number of active RSVP interfaces ...";
            err " ERROR: the number of active RSVP interfaces has changed from %s to
%s.", $PRE/active-count $POST/active-count;
        }
    }
    iterate rsvp-interface {
        id interface-name;
        no-diff interface-name {
            info "Checking if the name of the RSVP interface has changed ...";
            err " ERROR: the name of the RSVP interface %s has changed from %s to %s.",
$ID.1, $PRE/interface-name, $POST/interface-name;
        }
        no-diff rsvp-status {
            info "Checking the RSVP status for each interface ...";
            err " ERROR: the status of the RSVP interface %s has changed from %s to
%s.", $ID.1, $PRE/rsvp-status, $POST/rsvp-status;
        }
        list-not-less interface-name {
            info "Checking for missing RSVP interfaces ...";
            err " ERROR: the RSVP interface %s has gone missing.", $ID.1;
        }
        list-not-more interface-name {
            info "Checking for new RSVP interfaces ...";
            err " ERROR: the RSVP interface %s was not present before.", $ID.1;
        }
    }
}

check_show_mpls_interface {
    command show mpls interface;
    iterate mpls-interface {
        id interface-name;
        no-diff interface-name {
            info "Checking if there are changes in the name of the MPLS interfaces
...";
            err " ERROR: the interface %s has changed its name from %s to %s.", $PRE/
interface-name, $POST/interface-name;
        }
        no-diff mpls-interface-state {
            info "Checking the MPLS interface state ...";
            err " ERROR: the interface %s has changed its state from %s to %s.",
$ID.1, $PRE/mppls-interface-state, $POST/mppls-interface-state;
        }
        list-not-less interface-name {
            info "Checking for missing MPLS interfaces ...";
            err " ERROR: the interface %s has gone missing.", $ID.1;
        }
        list-not-more interface-name {
            info "Checking for new MPLS interfaces ...";
        }
    }
}

```

```

        err " ERROR: the interface %s was not present before.", $ID.1;
    }
}

check_show_mpls_lsp_extensive {
    command show mpls lsp extensive;
    iterate rsvp-session-data/rsvp-session/mpls-lsp {
        id ../../session-type;
        id name;
        no-diff destination-address {
            info "Checking if the LSP has changed its destination address ...";
            err " ERROR: the %s LSP %s has changed its destination address from %s to %s.", $ID.1, $ID.2, $PRE/destination-address, $POST/destination-address;
        }
        no-diff source-address {
            info "Checking if the LSP has changed its source address ...";
            err " ERROR: the %s LSP %s has changed its source address from %s to %s.", $ID.1, $ID.2, $PRE/source-address, $POST/source-address;
        }
        no-diff lsp-state {
            info "Checking if the LSP state has changed ...";
            err " ERROR: the %s LSP %s has changed its state from %s to %s.", $ID.1, $ID.2, $PRE/lsp-state, $POST/lsp-state;
        }
        no-diff name {
            info "Checking if the LSP name has changed ...";
            err " ERROR: the %s LSP %s has changed its name from %s to %s.", $ID.1, $ID.2, $PRE/name, $POST/name;
        }
        list-not-less name {
            info "Checking for missing LSPs ...";
            err " ERROR: the %s LSP %s has gone missing.", $ID.1, $ID.2;
        }
        list-not-more name {
            info "Checking for new LSPs ...";
            err " ERROR: the %s LSP %s was not present before.", $ID.1, $ID.2;
        }
        contains active-path, "(primary)" {
            info "Checking if the PRIMARY path is the active path for the LSP ...";
            err " ERROR: the %s LSP %s is not running on its PRIMARY path. It is currently running on %s.", $ID.1, $ID.2, $POST/active-path;
        }
    }
    iterate rsvp-session-data {
        id session-type;
        no-diff count {
            info "Checking the number of LSPs ...";
            err " ERROR: the number of %s LSPs has changed from %s to %s.", $ID.1, $PRE/count, $POST/count;
        }
        no-diff up-count {
            info "Checking the number of LSP in UP state ...";
            err " ERROR: the number of %s LSPs in UP state has changed from %s to %s.",

```



```

$ID.1, $PRE/up-count, $POST/up-count;
    }
    no-diff down-count {
        info "Checking the number of LSP in DOWN state ...";
        err " ERROR: the number of %s LSPs in DOWN state has changed from %s to
%s.", $ID.1, $PRE/down-count, $POST/down-count;
    }
}
}

```

Demonstration of the Configuration's Use

With the final configuration file in place, it's time to demonstrate its use in a simulated network change that creates at least one problem in each of the areas tested here. Since this simulation is executed in Junosphere, the Chassis verifications are skipped as some of the hardware details tested in the Chassis verifications are not presented in the VJX router in Junosphere.

The Pre-Snapshot

First of all, there are six routers (CE10, CE11, CE20, CE21, PE1, and PE2) to collect the snapshots. The snapshot name collected is called *before*. Here's the snapshot collection before the network change takes place:

```

dmontagner@querencia:~/jsnap$ ROUTERS=`cat routers.txt`dmontagner@querencia:~/jsnap/
final_conf_files/final_simulation$ for router in $ROUTERS; do jsnap --snap before -l
juniper -p Clouds -t $router network_verifications.conf ; done
Connecting to juniper@ce10 ...
CONNECTED.
EXEC: 'show isis interface' ...
SAVE: 'ce10__check_show_isis_interface__before.xml' ...
EXEC: 'show isis adjacency' ...
SAVE: 'ce10__check_show_isis_adjacency__before.xml' ...
EXEC: 'show interfaces terse' ...
SAVE: 'ce10__check_show_interfaces_terse__before.xml' ...
EXEC: 'show chassis hardware' ...
SAVE: 'ce10__check_show_chassis_hardware__before.xml' ...
EXEC: 'show chassis fpc' ...
SAVE: 'ce10__check_show_chassis_fpc__before.xml' ...
EXEC: 'show chassis fpc pic-status' ...
SAVE: 'ce10__check_show_chassis_fpc_pic_status__before.xml' ...
EXEC: 'show bgp summary' ...
SAVE: 'ce10__check_bgp_summary__before.xml' ...
EXEC: 'show rsvp session' ...
SAVE: 'ce10__check_show_rsvp_sessions__before.xml' ...
EXEC: 'show rsvp interface' ...
SAVE: 'ce10__check_show_rsvp_interface__before.xml' ...
EXEC: 'show mpls interface' ...
SAVE: 'ce10__check_show_mpls_interface__before.xml' ...
EXEC: 'show mpls lsp extensive' ...

```

```
SAVE: 'ce10__check_show_mpls_lsp_extensive__before.xml' ...
Connecting to juniper@ce11 ...
CONNECTED.
EXEC: 'show isis interface' ...
SAVE: 'ce11__check_show_isis_interface__before.xml' ...
EXEC: 'show isis adjacency' ...
SAVE: 'ce11__check_show_isis_adjacency__before.xml' ...
EXEC: 'show interfaces terse' ...
SAVE: 'ce11__check_show_interfaces_terse__before.xml' ...
EXEC: 'show chassis hardware' ...
SAVE: 'ce11__check_show_chassis_hardware__before.xml' ...
EXEC: 'show chassis fpc' ...
SAVE: 'ce11__check_show_chassis_fpc__before.xml' ...
EXEC: 'show chassis fpc pic-status' ...
SAVE: 'ce11__check_show_chassis_fpc_pic_status__before.xml' ...
EXEC: 'show bgp summary' ...
SAVE: 'ce11__check_bgp_summary__before.xml' ...
EXEC: 'show rsvp session' ...
SAVE: 'ce11__check_show_rsvp_sessions__before.xml' ...
EXEC: 'show rsvp interface' ...
SAVE: 'ce11__check_show_rsvp_interface__before.xml' ...
EXEC: 'show mpls interface' ...
SAVE: 'ce11__check_show_mpls_interface__before.xml' ...
EXEC: 'show mpls lsp extensive' ...
SAVE: 'ce11__check_show_mpls_lsp_extensive__before.xml' ...
Connecting to juniper@ce20 ...
CONNECTED.
EXEC: 'show isis interface' ...
SAVE: 'ce20__check_show_isis_interface__before.xml' ...
EXEC: 'show isis adjacency' ...
SAVE: 'ce20__check_show_isis_adjacency__before.xml' ...
EXEC: 'show interfaces terse' ...
SAVE: 'ce20__check_show_interfaces_terse__before.xml' ...
EXEC: 'show chassis hardware' ...
SAVE: 'ce20__check_show_chassis_hardware__before.xml' ...
EXEC: 'show chassis fpc' ...
SAVE: 'ce20__check_show_chassis_fpc__before.xml' ...
EXEC: 'show chassis fpc pic-status' ...
SAVE: 'ce20__check_show_chassis_fpc_pic_status__before.xml' ...
EXEC: 'show bgp summary' ...
SAVE: 'ce20__check_bgp_summary__before.xml' ...
EXEC: 'show rsvp session' ...
SAVE: 'ce20__check_show_rsvp_sessions__before.xml' ...
EXEC: 'show rsvp interface' ...
SAVE: 'ce20__check_show_rsvp_interface__before.xml' ...
EXEC: 'show mpls interface' ...
SAVE: 'ce20__check_show_mpls_interface__before.xml' ...
EXEC: 'show mpls lsp extensive' ...
SAVE: 'ce20__check_show_mpls_lsp_extensive__before.xml' ...
Connecting to juniper@ce21 ...
CONNECTED.
EXEC: 'show isis interface' ...
SAVE: 'ce21__check_show_isis_interface__before.xml' ...
EXEC: 'show isis adjacency' ...
```

```

SAVE: 'ce21__check_show_isis_adjacency__before.xml' ...
EXEC: 'show interfaces terse' ...
SAVE: 'ce21__check_show_interfaces_terse__before.xml' ...
EXEC: 'show chassis hardware' ...
SAVE: 'ce21__check_show_chassis_hardware__before.xml' ...
EXEC: 'show chassis fpc' ...
SAVE: 'ce21__check_show_chassis_fpc__before.xml' ...
EXEC: 'show chassis fpc pic-status' ...
SAVE: 'ce21__check_show_chassis_fpc_pic_status__before.xml' ...
EXEC: 'show bgp summary' ...
SAVE: 'ce21__check_bgp_summary__before.xml' ...
EXEC: 'show rsvp session' ...
SAVE: 'ce21__check_show_rsvp_sessions__before.xml' ...
EXEC: 'show rsvp interface' ...
SAVE: 'ce21__check_show_rsvp_interface__before.xml' ...
EXEC: 'show mpls interface' ...
SAVE: 'ce21__check_show_mpls_interface__before.xml' ...
EXEC: 'show mpls lsp extensive' ...
SAVE: 'ce21__check_show_mpls_lsp_extensive__before.xml' ...
Connecting to juniper@pe1 ...
CONNECTED.
EXEC: 'show isis interface' ...
SAVE: 'pe1__check_show_isis_interface__before.xml' ...
EXEC: 'show isis adjacency' ...
SAVE: 'pe1__check_show_isis_adjacency__before.xml' ...
EXEC: 'show interfaces terse' ...
SAVE: 'pe1__check_show_interfaces_terse__before.xml' ...
EXEC: 'show chassis hardware' ...
SAVE: 'pe1__check_show_chassis_hardware__before.xml' ...
EXEC: 'show chassis fpc' ...
SAVE: 'pe1__check_show_chassis_fpc__before.xml' ...
EXEC: 'show chassis fpc pic-status' ...
SAVE: 'pe1__check_show_chassis_fpc_pic_status__before.xml' ...
EXEC: 'show bgp summary' ...
SAVE: 'pe1__check_bgp_summary__before.xml' ...
EXEC: 'show rsvp session' ...
SAVE: 'pe1__check_show_rsvp_sessions__before.xml' ...
EXEC: 'show rsvp interface' ...
SAVE: 'pe1__check_show_rsvp_interface__before.xml' ...
EXEC: 'show mpls interface' ...
SAVE: 'pe1__check_show_mpls_interface__before.xml' ...
EXEC: 'show mpls lsp extensive' ...
SAVE: 'pe1__check_show_mpls_lsp_extensive__before.xml' ...
Connecting to juniper@pe2 ...
CONNECTED.
EXEC: 'show isis interface' ...
SAVE: 'pe2__check_show_isis_interface__before.xml' ...
EXEC: 'show isis adjacency' ...
SAVE: 'pe2__check_show_isis_adjacency__before.xml' ...
EXEC: 'show interfaces terse' ...
SAVE: 'pe2__check_show_interfaces_terse__before.xml' ...
EXEC: 'show chassis hardware' ...
SAVE: 'pe2__check_show_chassis_hardware__before.xml' ...
EXEC: 'show chassis fpc' ...

```

```

SAVE: 'pe2__check_show_chassis_fpc__before.xml' ...
EXEC: 'show chassis fpc pic-status' ...
SAVE: 'pe2__check_show_chassis_fpc_pic_status__before.xml' ...
EXEC: 'show bgp summary' ...
SAVE: 'pe2__check_bgp_summary__before.xml' ...
EXEC: 'show rsvp session' ...
SAVE: 'pe2__check_show_rsvp_sessions__before.xml' ...
EXEC: 'show rsvp interface' ...
SAVE: 'pe2__check_show_rsvp_interface__before.xml' ...
EXEC: 'show mpls interface' ...
SAVE: 'pe2__check_show_mpls_interface__before.xml' ...
EXEC: 'show mpls lsp extensive' ...
SAVE: 'pe2__check_show_mpls_lsp_extensive__before.xml' ...
dmontagner@querencia:~/jsnap$

```

In order to demonstrate JSNAP in action a problem in the network is introduced to simulate an issue found after the network change has been completed:.

```

[edit]
juniper@PE1# show interfaces ge-0/0/3
unit 0 {
    description "Connection to PE2";
    family inet {
        address 10.1.1.1/30;
    }
    family iso;
    family mpls;
}

[edit]
juniper@PE1# set interfaces ge-0/0/3 disable

[edit]
juniper@PE1# commit and-quit
commit complete
Exiting configuration mode

juniper@PE1>

```

As you can see, the problem is a very simple one that can easily happen in any network change. In this particular topology, this issue is very disruptive, as it will affect pretty much everything on the PE side.

The Post-Snapshot

At this point in our simulation, let's consider that the network change has finished and it is about to start the network verifications. Time to collect the post snapshot:

```

dmontagner@querencia:~/jsnap$ ROUTERS=`cat routers.txt`
dmontagner@querencia:~/jsnap$ for router in $ROUTERS; do jsnap --snap after -l juniper

```

```
-p Clouds -t $router network_verifications.conf ; done
Connecting to juniper@ce10 ...
CONNECTED.
EXEC: 'show isis interface' ...
SAVE: 'ce10__check_show_isis_interface__after.xml' ...
EXEC: 'show isis adjacency' ...
SAVE: 'ce10__check_show_isis_adjacency__after.xml' ...
EXEC: 'show interfaces terse' ...

<... omitted for brevity ...>

EXEC: 'show mpls interface' ...
SAVE: 'pe2__check_show_mpls_interface__after.xml' ...
EXEC: 'show mpls lsp extensive' ...
SAVE: 'pe2__check_show_mpls_lsp_extensive__after.xml' ...
dmontagner@querencia:~/jsnap$
```

For the sake of brevity, the outputs presented have been truncated. They are very similar to the ones collected in the pre-snapshot.

With the pre- (before) and post- (after) snapshots in hand, it is time to use our comprehensive set of JSNAP tests. The execution of the verification is presented here:

```
dmontagner@querencia:~/jsnap$ for router in $ROUTERS; do jsnap --check before,after -l
juniper -p Clouds -t $router network_verifications.conf; done
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: ce10
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-----
CHECKING SECTION: check_show_isis_interface
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS circuit type ..."
+ TEST PASSED: "Checking the Level 1 interface state ..."
+ TEST PASSED: "Checking the Level 2 interface state ..."
+ TEST PASSED: "Checking the interface Level 1 metric ..."
+ TEST PASSED: "Checking the interface Level 2 metric ..."
+ TEST PASSED: "Checking for missing ISIS interfaces ..."
+ TEST PASSED: "Checking for new ISIS interfaces ..."
-----
CHECKING SECTION: check_show_isis_adjacency
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS neighbour ..."
+ TEST PASSED: "Checking the Level of the ISIS adjacency ..."
+ TEST PASSED: "Checking the Subnetwork Point of Attachment (SNPA) ..."
+ TEST PASSED: "Checking for missing ISIS adjacencies ..."
+ TEST PASSED: "Checking for new ISIS adjacencies ..."
-----
CHECKING SECTION: check_show_interfaces_terse
-----
```

```

+ TEST PASSED: "Checking PHY operational status of interfaces ..."
+ TEST PASSED: "Checking PHY admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at PHY level ..."
+ TEST PASSED: "Checking for new interfaces at PHY level ..."
+ TEST PASSED: "Checking LOGICAL operational status of interfaces ..."
+ TEST PASSED: "Checking LOGICAL admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking for new interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking the address family configured in the interfaces ..."
+ TEST PASSED: "Checking for missing address family ..."
+ TEST PASSED: "Checking for new family address ..."
+ TEST PASSED: "Checking the interface address configured under the interface ..."
+ TEST PASSED: "Checking for missing interface address ..."
+ TEST PASSED: "Checking for new interface address ..."
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
-----
CHECKING SECTION: check_bgp_summary
-----
+ TEST PASSED: "Checking the number of BGP groups ..."
+ TEST PASSED: "Checking the number of BGP peers ..."
+ TEST PASSED: "Checking if the BGP configuration has at least 1 BGP peer configured
..."
+ TEST PASSED: "Checking the number of BGP peers down ..."
+ TEST PASSED: "Checking if the BGP peers addresses are still the same ..."
+ TEST PASSED: "Checking if the BGP peers ASNs are still the same ..."
+ TEST PASSED: "Checking if the BGP peer has flapped ..."
+ TEST PASSED: "Checking if the BGP peers are in Established state ..."
+ TEST PASSED: "Checking if the BGP RIB name has changed ..."
+ TEST PASSED: "Checking for missing RIBs ..."
+ TEST PASSED: "Checking for new RIBs ..."
- TEST FAILED: "Checking if the number of BGP active prefix has changed more than 20%."

      ERROR: the number of BGP active prefixes for RIB inet.0 on the BGP peer
192.168.1.1 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

- TEST FAILED: "Checking if the number of BGP received prefix has changed more than
20%."

      ERROR: the number of BGP received prefixes for RIB inet.0 on the BGP peer
192.168.1.1 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

- TEST FAILED: "Checking if the number of BGP accepted prefix has changed more than
20%."

      ERROR: the number of BGP accepted prefixes for RIB inet.0 on the BGP peer
192.168.1.1 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

+ TEST PASSED: "Checking if the number of BGP suppressed prefix has changed more than
20%."
-----
CHECKING SECTION: check_show_rsvp_sessions
-----
+ TEST PASSED: "Checking if the number of RSVP sessions has changed ..."

```

```
+ TEST PASSED: "Checking the Level 1 interface state ..."
```

```
+ TEST PASSED: "Checking the Level 2 interface state ..."
+ TEST PASSED: "Checking the interface Level 1 metric ..."
+ TEST PASSED: "Checking the interface Level 2 metric ..."
+ TEST PASSED: "Checking for missing ISIS interfaces ..."
+ TEST PASSED: "Checking for new ISIS interfaces ..."
```

CHECKING SECTION: check_show_isis_adjacency

```
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS neighbour ..."
+ TEST PASSED: "Checking the Level of the ISIS adjacency ..."
+ TEST PASSED: "Checking the Subnetwork Point of Attachment (SNPA) ..."
+ TEST PASSED: "Checking for missing ISIS adjacencies ..."
+ TEST PASSED: "Checking for new ISIS adjacencies ..."
```

CHECKING SECTION: check_show_interfaces_terse

```
+ TEST PASSED: "Checking PHY operational status of interfaces ..."
+ TEST PASSED: "Checking PHY admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at PHY level ..."
+ TEST PASSED: "Checking for new interfaces at PHY level ..."
+ TEST PASSED: "Checking LOGICAL operational status of interfaces ..."
+ TEST PASSED: "Checking LOGICAL admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking for new interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking the address family configured in the interfaces ..."
+ TEST PASSED: "Checking for missing address family ..."
+ TEST PASSED: "Checking for new family address ..."
+ TEST PASSED: "Checking the interface address configured under the interface ..."
+ TEST PASSED: "Checking for missing interface address ..."
+ TEST PASSED: "Checking for new interface address ..."
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
```

CHECKING SECTION: check_bgp_summary

```
+ TEST PASSED: "Checking the number of BGP groups ..."
+ TEST PASSED: "Checking the number of BGP peers ..."
+ TEST PASSED: "Checking if the BGP configuration has at least 1 BGP peer configured ..."
+ TEST PASSED: "Checking the number of BGP peers down ..."
+ TEST PASSED: "Checking if the BGP peers addresses are still the same ..."
+ TEST PASSED: "Checking if the BGP peers ASNs are still the same ..."
+ TEST PASSED: "Checking if the BGP peer has flapped ..."
+ TEST PASSED: "Checking if the BGP peers are in Established state ..."
+ TEST PASSED: "Checking if the BGP RIB name has changed ..."
+ TEST PASSED: "Checking for missing RIBs ..."
+ TEST PASSED: "Checking for new RIBs ..."
- TEST FAILED: "Checking if the number of BGP active prefix has changed more than 20%."
```

ERROR: the number of BGP active prefixes for RIB inet.0 on the BGP peer 192.168.2.1 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

```
- TEST FAILED: "Checking if the number of BGP received prefix has changed more than
```


20%."

ERROR: the number of BGP received prefixes for RIB inet.0 on the BGP peer 192.168.2.1 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

- TEST FAILED: "Checking if the number of BGP accepted prefix has changed more than 20%."

ERROR: the number of BGP accepted prefixes for RIB inet.0 on the BGP peer 192.168.2.1 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

+ TEST PASSED: "Checking if the number of BGP suppressed prefix has changed more than 20%."

CHECKING SECTION: check_show_rsvp_sessions

+ TEST PASSED: "Checking if the number of RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of displayed RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of active (UP) RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of inactive (DOWN) RSVP sessions has changed ..."
+ TEST PASSED: "Checking the source address of the RSVP sessions ..."
+ TEST PASSED: "Checking the destination address of the RSVP sessions ..."
+ TEST PASSED: "Checking the RSVP session names ..."
+ TEST PASSED: "Checking the RSVP session state ..."
+ TEST PASSED: "Checking for missing RSVP sessions ..."
+ TEST PASSED: "Checking for new RSVP sessions ..."

CHECKING SECTION: check_show_rsvp_interface

- TEST FAILED: "Checking the number of active RSVP interfaces ..."

XPath error : Invalid expression

jcs:printf(\$dynpf/pfmt,\$PRE/active-count \$POST/active-count)

^

xmlXPathEval: evaluation failed

dyn:evaluate() : unable to evaluate expression 'jcs:printf(\$dynpf/pfmt,\$PRE/active-count \$POST/active-count)'

+ TEST PASSED: "Checking if the name of the RSVP interface has changed ..."
+ TEST PASSED: "Checking the RSVP status for each interface ..."
+ TEST PASSED: "Checking for missing RSVP interfaces ..."
+ TEST PASSED: "Checking for new RSVP interfaces ..."

CHECKING SECTION: check_show_mpls_interface

+ TEST PASSED: "Checking if there are changes in the name of the MPLS interfaces ..."
+ TEST PASSED: "Checking the MPLS interface state ..."
+ TEST PASSED: "Checking for missing MPLS interfaces ..."
+ TEST PASSED: "Checking for new MPLS interfaces ..."

CHECKING SECTION: check_show_mpls_lsp_extensive

+ TEST PASSED: "Checking if the LSP has changed its destination address ..."
+ TEST PASSED: "Checking if the LSP has changed its source address ..."
+ TEST PASSED: "Checking if the LSP state has changed ..."
+ TEST PASSED: "Checking if the LSP name has changed ..."

```

+ TEST PASSED: "Checking for missing LSPs ..."
+ TEST PASSED: "Checking for new LSPs ..."
+ TEST PASSED: "Checking if the PRIMARY path is the active path for the LSP ..."
+ TEST PASSED: "Checking the number of LSPs ..."
+ TEST PASSED: "Checking the number of LSP in UP state ..."
+ TEST PASSED: "Checking the number of LSP in DOWN state ..."
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: ce20
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-----
CHECKING SECTION: check_show_isis_interface
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS circuit type ..."
+ TEST PASSED: "Checking the Level 1 interface state ..."
+ TEST PASSED: "Checking the Level 2 interface state ..."
+ TEST PASSED: "Checking the interface Level 1 metric ..."
+ TEST PASSED: "Checking the interface Level 2 metric ..."
+ TEST PASSED: "Checking for missing ISIS interfaces ..."
+ TEST PASSED: "Checking for new ISIS interfaces ..."
-----
CHECKING SECTION: check_show_isis_adjacency
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS neighbour ..."
+ TEST PASSED: "Checking the Level of the ISIS adjacency ..."
+ TEST PASSED: "Checking the Subnetwork Point of Attachment (SNPA) ..."
+ TEST PASSED: "Checking for missing ISIS adjacencies ..."
+ TEST PASSED: "Checking for new ISIS adjacencies ..."
-----
CHECKING SECTION: check_show_interfaces_terse
-----
+ TEST PASSED: "Checking PHY operational status of interfaces ..."
+ TEST PASSED: "Checking PHY admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at PHY level ..."
+ TEST PASSED: "Checking for new interfaces at PHY level ..."
+ TEST PASSED: "Checking LOGICAL operational status of interfaces ..."
+ TEST PASSED: "Checking LOGICAL admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking for new interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking the address family configured in the interfaces ..."
+ TEST PASSED: "Checking for missing address family ..."
+ TEST PASSED: "Checking for new family address ..."
+ TEST PASSED: "Checking the interface address configured under the interface ..."
+ TEST PASSED: "Checking for missing interface address ..."
+ TEST PASSED: "Checking for new interface address ..."
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
-----
CHECKING SECTION: check_bgp_summary
-----
+ TEST PASSED: "Checking the number of BGP groups ..."

```

```

+ TEST PASSED: "Checking the number of BGP peers ..."
+ TEST PASSED: "Checking if the BGP configuration has at least 1 BGP peer configured
..."
+ TEST PASSED: "Checking the number of BGP peers down ..."
+ TEST PASSED: "Checking if the BGP peers addresses are still the same ..."
+ TEST PASSED: "Checking if the BGP peers ASNs are still the same ..."
+ TEST PASSED: "Checking if the BGP peer has flapped ..."
+ TEST PASSED: "Checking if the BGP peers are in Established state ..."
+ TEST PASSED: "Checking if the BGP RIB name has changed ..."
+ TEST PASSED: "Checking for missing RIBs ..."
+ TEST PASSED: "Checking for new RIBs ..."
- TEST FAILED: "Checking if the number of BGP active prefix has changed more than 20%."

```

ERROR: the number of BGP active prefixes for RIB inet.0 on the BGP peer 192.168.1.5 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

```

- TEST FAILED: "Checking if the number of BGP received prefix has changed more than 20%."

```

ERROR: the number of BGP received prefixes for RIB inet.0 on the BGP peer 192.168.1.5 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

```

- TEST FAILED: "Checking if the number of BGP accepted prefix has changed more than 20%."

```

ERROR: the number of BGP accepted prefixes for RIB inet.0 on the BGP peer 192.168.1.5 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

```

+ TEST PASSED: "Checking if the number of BGP suppressed prefix has changed more than 20%."

```

```

-----
CHECKING SECTION: check_show_rsvp_sessions

```

```

+ TEST PASSED: "Checking if the number of RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of displayed RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of active (UP) RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of inactive (DOWN) RSVP sessions has changed
..."
+ TEST PASSED: "Checking the source address of the RSVP sessions ..."
+ TEST PASSED: "Checking the destination address of the RSVP sessions ..."
+ TEST PASSED: "Checking the RSVP session names ..."
+ TEST PASSED: "Checking the RSVP session state ..."
+ TEST PASSED: "Checking for missing RSVP sessions ..."
+ TEST PASSED: "Checking for new RSVP sessions ..."

```

```

-----
CHECKING SECTION: check_show_rsvp_interface

```

```

- TEST FAILED: "Checking the number of active RSVP interfaces ..."

```

XPath error : Invalid expression

```

jcs:printf($_dynpf/pfmt,$PRE/active-count $POST/active-count)

```

^

xmlXPathEval: evaluation failed

dyn:evaluate() : unable to evaluate expression 'jcs:printf(\$_dynpf/pfmt,\$PRE/active-count \$POST/active-count)'

```

+ TEST PASSED: "Checking if the name of the RSVP interface has changed ..."

```

```

+ TEST PASSED: "Checking the RSVP status for each interface ..."
+ TEST PASSED: "Checking for missing RSVP interfaces ..."
+ TEST PASSED: "Checking for new RSVP interfaces ..."
-----
CHECKING SECTION: check_show_mpls_interface
-----
+ TEST PASSED: "Checking if there are changes in the name of the MPLS interfaces ..."
+ TEST PASSED: "Checking the MPLS interface state ..."
+ TEST PASSED: "Checking for missing MPLS interfaces ..."
+ TEST PASSED: "Checking for new MPLS interfaces ..."
-----
CHECKING SECTION: check_show_mpls_lsp_extensive
-----
+ TEST PASSED: "Checking if the LSP has changed its destination address ..."
+ TEST PASSED: "Checking if the LSP has changed its source address ..."
+ TEST PASSED: "Checking if the LSP state has changed ..."
+ TEST PASSED: "Checking if the LSP name has changed ..."
+ TEST PASSED: "Checking for missing LSPs ..."
+ TEST PASSED: "Checking for new LSPs ..."
+ TEST PASSED: "Checking if the PRIMARY path is the active path for the LSP ..."
+ TEST PASSED: "Checking the number of LSPs ..."
+ TEST PASSED: "Checking the number of LSP in UP state ..."
+ TEST PASSED: "Checking the number of LSP in DOWN state ..."
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: ce21
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-----
CHECKING SECTION: check_show_isis_interface
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS circuit type ..."
+ TEST PASSED: "Checking the Level 1 interface state ..."
+ TEST PASSED: "Checking the Level 2 interface state ..."
+ TEST PASSED: "Checking the interface Level 1 metric ..."
+ TEST PASSED: "Checking the interface Level 2 metric ..."
+ TEST PASSED: "Checking for missing ISIS interfaces ..."
+ TEST PASSED: "Checking for new ISIS interfaces ..."
-----
CHECKING SECTION: check_show_isis_adjacency
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS neighbour ..."
+ TEST PASSED: "Checking the Level of the ISIS adjacency ..."
+ TEST PASSED: "Checking the Subnetwork Point of Attachment (SNPA) ..."
+ TEST PASSED: "Checking for missing ISIS adjacencies ..."
+ TEST PASSED: "Checking for new ISIS adjacencies ..."
-----
CHECKING SECTION: check_show_interfaces_terse
-----
+ TEST PASSED: "Checking PHY operational status of interfaces ..."
+ TEST PASSED: "Checking PHY admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at PHY level ..."
+ TEST PASSED: "Checking for new interfaces at PHY level ..."

```

```

+ TEST PASSED: "Checking LOGICAL operational status of interfaces ..."
+ TEST PASSED: "Checking LOGICAL admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking for new interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking the address family configured in the interfaces ..."
+ TEST PASSED: "Checking for missing address family ..."
+ TEST PASSED: "Checking for new family address ..."
+ TEST PASSED: "Checking the interface address configured under the interface ..."
+ TEST PASSED: "Checking for missing interface address ..."
+ TEST PASSED: "Checking for new interface address ..."
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
-----

```

CHECKING SECTION: check_bgp_summary

```

-----
+ TEST PASSED: "Checking the number of BGP groups ..."
+ TEST PASSED: "Checking the number of BGP peers ..."
+ TEST PASSED: "Checking if the BGP configuration has at least 1 BGP peer configured ..."
+ TEST PASSED: "Checking the number of BGP peers down ..."
+ TEST PASSED: "Checking if the BGP peers addresses are still the same ..."
+ TEST PASSED: "Checking if the BGP peers ASNs are still the same ..."
+ TEST PASSED: "Checking if the BGP peer has flapped ..."
+ TEST PASSED: "Checking if the BGP peers are in Established state ..."
+ TEST PASSED: "Checking if the BGP RIB name has changed ..."
+ TEST PASSED: "Checking for missing RIBs ..."
+ TEST PASSED: "Checking for new RIBs ..."
- TEST FAILED: "Checking if the number of BGP active prefix has changed more than 20%."

```

ERROR: the number of BGP active prefixes for RIB inet.0 on the BGP peer 192.168.2.5 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

- TEST FAILED: "Checking if the number of BGP received prefix has changed more than 20%."

ERROR: the number of BGP received prefixes for RIB inet.0 on the BGP peer 192.168.2.5 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

- TEST FAILED: "Checking if the number of BGP accepted prefix has changed more than 20%."

ERROR: the number of BGP accepted prefixes for RIB inet.0 on the BGP peer 192.168.2.5 (ASN 65000) has changed more than 20 percent (before = 3 / after = 0).

+ TEST PASSED: "Checking if the number of BGP suppressed prefix has changed more than 20%."

CHECKING SECTION: check_show_rsvp_sessions

```

-----
+ TEST PASSED: "Checking if the number of RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of displayed RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of active (UP) RSVP sessions has changed ..."
+ TEST PASSED: "Checking if the number of inactive (DOWN) RSVP sessions has changed ..."

```

```

+ TEST PASSED: "Checking the source address of the RSVP sessions ..."
+ TEST PASSED: "Checking the destination address of the RSVP sessions ..."
+ TEST PASSED: "Checking the RSVP session names ..."
+ TEST PASSED: "Checking the RSVP session state ..."
+ TEST PASSED: "Checking for missing RSVP sessions ..."
+ TEST PASSED: "Checking for new RSVP sessions ..."
-----
CHECKING SECTION: check_show_rsvp_interface
-----
- TEST FAILED: "Checking the number of active RSVP interfaces ..."
XPath error : Invalid expression
jcs:printf($dynpf/pfmt,$PRE/active-count $POST/active-count)
      ^
xmlXPathEval: evaluation failed
dyn:evaluate() : unable to evaluate expression 'jcs:printf($dynpf/pfmt,$PRE/active-
count $POST/active-count)'
+ TEST PASSED: "Checking if the name of the RSVP interface has changed ..."
+ TEST PASSED: "Checking the RSVP status for each interface ..."
+ TEST PASSED: "Checking for missing RSVP interfaces ..."
+ TEST PASSED: "Checking for new RSVP interfaces ..."
-----
CHECKING SECTION: check_show_mpls_interface
-----
+ TEST PASSED: "Checking if there are changes in the name of the MPLS interfaces ..."
+ TEST PASSED: "Checking the MPLS interface state ..."
+ TEST PASSED: "Checking for missing MPLS interfaces ..."
+ TEST PASSED: "Checking for new MPLS interfaces ..."
-----
CHECKING SECTION: check_show_mpls_lsp_extensive
-----
+ TEST PASSED: "Checking if the LSP has changed its destination address ..."
+ TEST PASSED: "Checking if the LSP has changed its source address ..."
+ TEST PASSED: "Checking if the LSP state has changed ..."
+ TEST PASSED: "Checking if the LSP name has changed ..."
+ TEST PASSED: "Checking for missing LSPs ..."
+ TEST PASSED: "Checking for new LSPs ..."
+ TEST PASSED: "Checking if the PRIMARY path is the active path for the LSP ..."
+ TEST PASSED: "Checking the number of LSPs ..."
+ TEST PASSED: "Checking the number of LSP in UP state ..."
+ TEST PASSED: "Checking the number of LSP in DOWN state ..."
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: pe1
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
-----
CHECKING SECTION: check_show_isis_interface
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS circuit type ..."
+ TEST PASSED: "Checking the Level 1 interface state ..."
+ TEST PASSED: "Checking the Level 2 interface state ..."
+ TEST PASSED: "Checking the interface Level 1 metric ..."
+ TEST PASSED: "Checking the interface Level 2 metric ..."
+ TEST PASSED: "Checking for missing ISIS interfaces ..."

```

+ TEST PASSED: "Checking for new ISIS interfaces ..."

CHECKING SECTION: check_show_isis_adjacency

+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS neighbour ..."
+ TEST PASSED: "Checking the Level of the ISIS adjacency ..."
+ TEST PASSED: "Checking the Subnetwork Point of Attachment (SNPA) ..."
- TEST FAILED: "Checking for missing ISIS adjacencies ..."

ERROR: the ISIS adjacency for interface ge-0/0/3.0 is missing.

+ TEST PASSED: "Checking for new ISIS adjacencies ..."

CHECKING SECTION: check_show_interfaces_terse

+ TEST PASSED: "Checking PHY operational status of interfaces ..."
- TEST FAILED: "Checking PHY admin status of interfaces ..."

ERROR: the admin status for interface ge-0/0/3 has changed from up to down.

+ TEST PASSED: "Checking for missing interfaces at PHY level ..."
+ TEST PASSED: "Checking for new interfaces at PHY level ..."
- TEST FAILED: "Checking LOGICAL operational status of interfaces ..."

ERROR: the operational status for interface ge-0/0/3.0 has changed from up to down.

+ TEST PASSED: "Checking LOGICAL admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking for new interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking the address family configured in the interfaces ..."
+ TEST PASSED: "Checking for missing address family ..."
+ TEST PASSED: "Checking for new family address ..."
+ TEST PASSED: "Checking the interface address configured under the interface ..."
+ TEST PASSED: "Checking for missing interface address ..."
+ TEST PASSED: "Checking for new interface address ..."
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!

CHECKING SECTION: check_bgp_summary

+ TEST PASSED: "Checking the number of BGP groups ..."
+ TEST PASSED: "Checking the number of BGP peers ..."
+ TEST PASSED: "Checking if the BGP configuration has at least 1 BGP peer configured ..."
- TEST FAILED: "Checking the number of BGP peers down ..."

ERROR: the number of BGP peers down has changed from 0 to 1.

+ TEST PASSED: "Checking if the BGP peers addresses are still the same ..."
+ TEST PASSED: "Checking if the BGP peers ASNs are still the same ..."
- TEST FAILED: "Checking if the BGP peer has flapped ..."

ERROR: the BGP peer 10.100.100.2 (ASN 65000) has flapped.

- TEST FAILED: "Checking if the BGP peers are in Established state ..."

ERROR: the BGP peer 10.100.100.2 (ASN 65000) is not in Established state.

+ TEST PASSED: "Checking if the BGP RIB name has changed ..."

- TEST FAILED: "Checking for missing RIBs ..."

ERROR: the RIB inet.0 for the BGP peer 10.100.100.2 (ASN 65000) has gone missing.

ERROR: the RIB bgp.l3vpn.0 for the BGP peer 10.100.100.2 (ASN 65000) has gone missing.

ERROR: the RIB bgp.rtarget.0 for the BGP peer 10.100.100.2 (ASN 65000) has gone missing.

ERROR: the RIB bgp.mvpn.0 for the BGP peer 10.100.100.2 (ASN 65000) has gone missing.

ERROR: the RIB VPNA.inet.0 for the BGP peer 10.100.100.2 (ASN 65000) has gone missing.

ERROR: the RIB VPNB.inet.0 for the BGP peer 10.100.100.2 (ASN 65000) has gone missing.

+ TEST PASSED: "Checking for new RIBs ..."

+ TEST PASSED: "Checking if the number of BGP active prefix has changed more than 20%."

+ TEST PASSED: "Checking if the number of BGP received prefix has changed more than 20%."

+ TEST PASSED: "Checking if the number of BGP accepted prefix has changed more than 20%."

+ TEST PASSED: "Checking if the number of BGP suppressed prefix has changed more than 20%."

CHECKING SECTION: check_show_rsvp_sessions

- TEST FAILED: "Checking if the number of RSVP sessions has changed ..."

ERROR: the number of | Ingress | RSVP sessions has changed from 1 to 0.

ERROR: the number of | Egress | RSVP sessions has changed from 1 to 0.

- TEST FAILED: "Checking if the number of displayed RSVP sessions has changed ..."

ERROR: the number of | Ingress | displayed RSVP sessions has changed from 1 to 0.

ERROR: the number of | Egress | displayed RSVP sessions has changed from 1 to

0.

- TEST FAILED: "Checking if the number of active (UP) RSVP sessions has changed ..."

ERROR: the number of | Ingress | active (UP) RSVP sessions has changed from 1 to 0.

ERROR: the number of | Egress | active (UP) RSVP sessions has changed from 1 to 0.

+ TEST PASSED: "Checking if the number of inactive (DOWN) RSVP sessions has changed ..."

+ TEST PASSED: "Checking the source address of the RSVP sessions ..."

+ TEST PASSED: "Checking the destination address of the RSVP sessions ..."

+ TEST PASSED: "Checking the RSVP session names ..."

+ TEST PASSED: "Checking the RSVP session state ..."

- TEST FAILED: "Checking for missing RSVP sessions ..."

ERROR: the | Ingress | RSVP session to-PE2 has gone missing.

ERROR: the | Egress | RSVP session to-PE1 has gone missing.

+ TEST PASSED: "Checking for new RSVP sessions ..."

CHECKING SECTION: check_show_rsvp_interface

+ TEST PASSED: "Checking the number of active RSVP interfaces ..."

+ TEST PASSED: "Checking if the name of the RSVP interface has changed ..."

- TEST FAILED: "Checking the RSVP status for each interface ..."

ERROR: the status of the RSVP interface ge-0/0/3.0 has changed from Up to Down.

+ TEST PASSED: "Checking for missing RSVP interfaces ..."

+ TEST PASSED: "Checking for new RSVP interfaces ..."

CHECKING SECTION: check_show_mpls_interface

+ TEST PASSED: "Checking if there are changes in the name of the MPLS interfaces ..."

- TEST FAILED: "Checking the MPLS interface state ..."

ERROR: the interface ge-0/0/3.0 has changed its state from Up to Dn.

+ TEST PASSED: "Checking for missing MPLS interfaces ..."

+ TEST PASSED: "Checking for new MPLS interfaces ..."

CHECKING SECTION: check_show_mpls_lsp_extensive

+ TEST PASSED: "Checking if the LSP has changed its destination address ..."

+ TEST PASSED: "Checking if the LSP has changed its source address ..."

- TEST FAILED: "Checking if the LSP state has changed ..."

ERROR: the Ingress LSP to-PE2 has changed its state from Up to Dn.

```
+ TEST PASSED: "Checking if the LSP name has changed ..."
+ TEST PASSED: "Checking for missing LSPs ..."
+ TEST PASSED: "Checking for new LSPs ..."
- TEST FAILED: "Checking if the PRIMARY path is the active path for the LSP ..."
```

ERROR: the Ingress LSP to-PE2 is not running on its PRIMARY path. It is currently running on (none).

```
- TEST FAILED: "Checking the number of LSPs ..."
```

ERROR: the number of Egress LSPs has changed from 1 to 0.

```
- TEST FAILED: "Checking the number of LSP in UP state ..."
```

ERROR: the number of Ingress LSPs in UP state has changed from 1 to 0.

ERROR: the number of Egress LSPs in UP state has changed from 1 to 0.

```
- TEST FAILED: "Checking the number of LSP in DOWN state ..."
```

ERROR: the number of Ingress LSPs in DOWN state has changed from 0 to 1.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: pe2
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
-----
CHECKING SECTION: check_show_isis_interface
```

```
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS circuit type ..."
+ TEST PASSED: "Checking the Level 1 interface state ..."
+ TEST PASSED: "Checking the Level 2 interface state ..."
+ TEST PASSED: "Checking the interface Level 1 metric ..."
+ TEST PASSED: "Checking the interface Level 2 metric ..."
+ TEST PASSED: "Checking for missing ISIS interfaces ..."
+ TEST PASSED: "Checking for new ISIS interfaces ..."
-----
```

```
CHECKING SECTION: check_show_isis_adjacency
```

```
-----
+ TEST PASSED: "Checking the ISIS interface names ..."
+ TEST PASSED: "Checking the ISIS neighbour ..."
+ TEST PASSED: "Checking the Level of the ISIS adjacency ..."
+ TEST PASSED: "Checking the Subnetwork Point of Attachment (SNPA) ..."
- TEST FAILED: "Checking for missing ISIS adjacencies ..."
```

ERROR: the ISIS adjacency for interface ge-0/0/1.0 is missing.

```
+ TEST PASSED: "Checking for new ISIS adjacencies ..."
```

```
-----
CHECKING SECTION: check_show_interfaces_terse
```

```
-----
+ TEST PASSED: "Checking PHY operational status of interfaces ..."
```

```

+ TEST PASSED: "Checking PHY admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at PHY level ..."
+ TEST PASSED: "Checking for new interfaces at PHY level ..."
+ TEST PASSED: "Checking LOGICAL operational status of interfaces ..."
+ TEST PASSED: "Checking LOGICAL admin status of interfaces ..."
+ TEST PASSED: "Checking for missing interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking for new interfaces at LOGICAL level ..."
+ TEST PASSED: "Checking the address family configured in the interfaces ..."
+ TEST PASSED: "Checking for missing address family ..."
+ TEST PASSED: "Checking for new family address ..."
+ TEST PASSED: "Checking the interface address configured under the interface ..."
+ TEST PASSED: "Checking for missing interface address ..."
+ TEST PASSED: "Checking for new interface address ..."
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
---> ERROR: section '' not found ... SKIPPING!
-----
CHECKING SECTION: check_bgp_summary
-----
+ TEST PASSED: "Checking the number of BGP groups ..."
+ TEST PASSED: "Checking the number of BGP peers ..."
+ TEST PASSED: "Checking if the BGP configuration has at least 1 BGP peer configured
..."
- TEST FAILED: "Checking the number of BGP peers down ..."

        ERROR: the number of BGP peers down has changed from 0 to 1.

+ TEST PASSED: "Checking if the BGP peers addresses are still the same ..."
+ TEST PASSED: "Checking if the BGP peers ASNs are still the same ..."
- TEST FAILED: "Checking if the BGP peer has flapped ..."

        ERROR: the BGP peer 10.100.100.1 (ASN 65000) has flapped.

- TEST FAILED: "Checking if the BGP peers are in Established state ..."

        ERROR: the BGP peer 10.100.100.1 (ASN 65000) is not in Established state.

+ TEST PASSED: "Checking if the BGP RIB name has changed ..."
- TEST FAILED: "Checking for missing RIBs ..."

        ERROR: the RIB inet.0 for the BGP peer 10.100.100.1 (ASN 65000) has gone
missing.

        ERROR: the RIB bgp.l3vpn.0 for the BGP peer 10.100.100.1 (ASN 65000) has gone
missing.

        ERROR: the RIB bgp.rtarget.0 for the BGP peer 10.100.100.1 (ASN 65000) has
gone missing.

        ERROR: the RIB bgp.mvpn.0 for the BGP peer 10.100.100.1 (ASN 65000) has gone
missing.

```

ERROR: the RIB VPNA.inet.0 for the BGP peer 10.100.100.1 (ASN 65000) has gone missing.

ERROR: the RIB VPNB.inet.0 for the BGP peer 10.100.100.1 (ASN 65000) has gone missing.

+ TEST PASSED: "Checking for new RIBs ..."
 + TEST PASSED: "Checking if the number of BGP active prefix has changed more than 20%."
 + TEST PASSED: "Checking if the number of BGP received prefix has changed more than 20%."
 + TEST PASSED: "Checking if the number of BGP accepted prefix has changed more than 20%."
 + TEST PASSED: "Checking if the number of BGP suppressed prefix has changed more than 20%."

 CHECKING SECTION: check_show_rsvp_sessions

- TEST FAILED: "Checking if the number of RSVP sessions has changed ..."

ERROR: the number of | Ingress | RSVP sessions has changed from 1 to 0.

ERROR: the number of | Egress | RSVP sessions has changed from 1 to 0.

- TEST FAILED: "Checking if the number of displayed RSVP sessions has changed ..."

ERROR: the number of | Ingress | displayed RSVP sessions has changed from 1 to 0.

ERROR: the number of | Egress | displayed RSVP sessions has changed from 1 to 0.

- TEST FAILED: "Checking if the number of active (UP) RSVP sessions has changed ..."

ERROR: the number of | Ingress | active (UP) RSVP sessions has changed from 1 to 0.

ERROR: the number of | Egress | active (UP) RSVP sessions has changed from 1 to 0.

+ TEST PASSED: "Checking if the number of inactive (DOWN) RSVP sessions has changed ..."
 + TEST PASSED: "Checking the source address of the RSVP sessions ..."
 + TEST PASSED: "Checking the destination address of the RSVP sessions ..."
 + TEST PASSED: "Checking the RSVP session names ..."
 + TEST PASSED: "Checking the RSVP session state ..."
 - TEST FAILED: "Checking for missing RSVP sessions ..."

ERROR: the | Ingress | RSVP session to-PE1 has gone missing.

ERROR: the | Egress | RSVP session to-PE2 has gone missing.

+ TEST PASSED: "Checking for new RSVP sessions ..."

CHECKING SECTION: check_show_rsvp_interface

+ TEST PASSED: "Checking the number of active RSVP interfaces ..."

+ TEST PASSED: "Checking if the name of the RSVP interface has changed ..."

+ TEST PASSED: "Checking the RSVP status for each interface ..."

+ TEST PASSED: "Checking for missing RSVP interfaces ..."

+ TEST PASSED: "Checking for new RSVP interfaces ..."

CHECKING SECTION: check_show_mpls_interface

+ TEST PASSED: "Checking if there are changes in the name of the MPLS interfaces ..."

+ TEST PASSED: "Checking the MPLS interface state ..."

+ TEST PASSED: "Checking for missing MPLS interfaces ..."

+ TEST PASSED: "Checking for new MPLS interfaces ..."

CHECKING SECTION: check_show_mpls_lsp_extensive

+ TEST PASSED: "Checking if the LSP has changed its destination address ..."

+ TEST PASSED: "Checking if the LSP has changed its source address ..."

- TEST FAILED: "Checking if the LSP state has changed ..."

ERROR: the Ingress LSP to-PE1 has changed its state from Up to Dn.

+ TEST PASSED: "Checking if the LSP name has changed ..."

+ TEST PASSED: "Checking for missing LSPs ..."

+ TEST PASSED: "Checking for new LSPs ..."

- TEST FAILED: "Checking if the PRIMARY path is the active path for the LSP ..."

ERROR: the Ingress LSP to-PE1 is not running on its PRIMARY path. It is currently running on (none).

- TEST FAILED: "Checking the number of LSPs ..."

ERROR: the number of Egress LSPs has changed from 1 to 0.

- TEST FAILED: "Checking the number of LSP in UP state ..."

ERROR: the number of Ingress LSPs in UP state has changed from 1 to 0.

ERROR: the number of Egress LSPs in UP state has changed from 1 to 0.

- TEST FAILED: "Checking the number of LSP in DOWN state ..."

ERROR: the number of Ingress LSPs in DOWN state has changed from 0 to 1.

dmontagner@querencia:~/jsnap\$

Analyzing the results of the network verification presented here, there are some observations that can be made:

- A simple configuration mistake (or problem) can be devastating in some network designs.
- A simple problem or mistake can lead to other cascading problems.
- JSNAP can be very assertive by automatically identifying each one of the network problems.
- JSNAP can execute a very large number of verifications across a very large number of devices taking much less time than any other method.
- In some network scenarios, it is better to have different JSNAP files to be used with different groups of routers. For instance, we could have a JSNAP configuration file for CE routers, another for PE routers, and a third one for P routers.
- Some tests failed reporting what appears to be a software problem in JSNAP (RSVP tests). Those errors are caused because under the failure condition of the topology of this book, few of the XPath tests by the RSVP tests will not be present in the post- snapshot. Because the JSNAP test operators in use on those tests require both pre- and post-snapshots to contain the XPath being tested, JSNAP will fail the tests whenever one or both snapshots don't contain the XPath being tested.
- Chassis verifications were disabled (skipped) in these tests because the network topology was running in Junosphere. Junosphere uses virtual-routers that may or may not have the complete hardware information needed for the Chassis verifications developed in this book.

Summary

The development of a JSNAP configuration file may be a bit difficult when you first learn about JSNAP, but this book was developed with the idea of helping network engineers to automate network verification tests, as well as helping them to accelerate the JSNAP learning process. Moreover, the book focuses on developing the most used set of network verifications that can be applied to a wide range of networks. Complex networks will require further development of the JSNAP configuration file presented in this book, but for sure, this configuration file can be used as a starting point.

I hope you have enjoyed your journey through this book.

– Diogo Montagner

Appendix

Answer to the Chapter 3 Challenge

Here is the JSNAP Configuration file for CPU, the answer to Chapter 3's JSNAP Challenge:

```
do {
    check_cpu;
}

check_cpu {
    command show chassis routing-engine;
    iterate route-engine {
        id slot;
        in-range cpu-idle, 20, 99 {
            info "Checking if CPU Idle is within 20% ~ 99%.";
            err "The CPU utilisation of RE %s is too high!", $ID.1;
        }
        no-diff status {
            info "Checking the REs have changed its status...";
            err " ERROR: the RE %s has changed its status from %s to %s.", $ID.1,
$PRE/status, $POST/status;
        }
        is-lt temperature/@junos:celsius, 55 {
            info "Checking if the Routing-Engine temperature is below 55 degrees
Celsius ...";
            err " ERROR: the Routing-Engine temperature is higher than 55 degrees
Celsius (current = %s degrees Celsius).", $POST/temperature/@junos:celsius;
        }
    }
    item route-engine[slot = '0'] {
        is-equal mastership-state, "master" {
            info "Checking if RE0 is the Master RE ...";
            err " ERROR: RE0 is not the Master RE. Its current state is %s",
$POST/mastership-state;
        }
    }
}
```

What To Do, Where To Go Next

<http://www.juniper.net/dayone>

Free access to all *Day One* and *This Week* Juniper books in PDF format.

www.juniper.net/junos

Everything you need for Junos adoption and education.

<http://forums.juniper.net/jnet>

The Juniper-sponsored J-Net Communities forum is dedicated to sharing information, best practices, and questions about Juniper products, technologies, and solutions. Register to participate at this free forum.

http://www.juniper.net/techpubs/en_US/junos-snapshot1.0/information-products/pathway-pages/junos-snapshot.html#overview

The Junos Snapshot Administrator technical documentation.

<http://www.juniper.net/support/downloads/?p=jsnap#sw>

Download the Junos Snapshot Administrator software.

<https://github.com/Juniper/junos-snapshot-administrator>

The Junos Snapshot Administrator repository on GitHub.