JUNIPER NETWORKS | Engineering Simplicity

Junos Snapshot Administrator Guide

Published
2021-04-22

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos Snapshot Administrator Guide

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide

Use this guide to capture and audit runtime environment snapshots of your networked devices running Junos OS using configuration files written in SLAX.

RELATED DOCUMENTATION

- Junos Snapshot Administrator Software

CHAPTER 1

Junos Snapshot Administrator Overview

Junos Snapshot Administrator Overview

Junos® Snapshot enables you to capture and audit runtime environment snapshots of your networked devices running the Junos operating system (Junos OS). You can capture and validate the operational status of a device and review operational changes to a device. You create configuration files that define the scope of snapshots and customize the evaluation criteria for the snapshot data.

Junos Snapshot Administrator can perform the following functions on either a single device or list of devices running Junos OS:

- Take a snapshot of the runtime environment on a device.
- Compare two snapshots.
- Audit a device's runtime environment against pre-defined criteria.

For example, prior to a software or hardware upgrade on a device, you can take a pre-install and post-install snapshot of the device and then compare the two snapshots. You can then review the operational changes on the device and validate these changes from a list of expected changes.

To be successful in using Junos Snapshot Administrator, you need a basic understanding of Junos OS *operational mode command* output in an XML format. You also need familiarity with XPath expressions. For more information about the Junos XML structure and XPath expressions, see the Juniper Networks Day One book *Navigating the Junos XML Hierarchy* at https://www.juniper.net/dayone . For additional information about XPath, review the W3Schools XPath website at https://www.w3schools.com/xml/xpath_intro.asp.

RELATED DOCUMENTATION

Installing Junos Snapshot Administrator | 4

Junos Snapshot Administrator Configuration File | 8

CHAPTER 2

Installing Junos Snapshot Administrator

Installing Junos Snapshot Administrator

IN THIS SECTION

You install Junos Snapshot Administrator on a remote server within the networking environment. Prior to installing Junos Snapshot Administrator, ensure that the remote server meets the following minimum requirements:

- 32-bit or 64-bit CentOS 5.5 or later operating system

  For more information about installing the CentOS software on your server, see the CentOS website at http://www.centos.org/ .

- **getopt** (GNU enhanced) program is installed

  The **getopt** (GNU enhanced) utility is a common utility that is usually (but not always) installed as part of the basic CentOS installation process. For more information about installing this utility, see the installation instructions for the CentOS operating system.

Prior to using Junos Snapshot Administrator, ensure that you satisfy the requirements for SSHv2 connections with devices running Junos OS.

The procedure for installing Junos Snapshot Administrator and the requirements for SSHv2 connections with devices running Junos OS are outlined in the following sections.

Installing Junos Snapshot Administrator

Before you begin, ensure that you have the following on the remote server:

- Administrator privileges
- Internet access

To install Junos Snapshot Administrator:

1. Access the download page at https://www.juniper.net/support/downloads/?p=jsnap.

2. Select the software tab.
3. Download the Junos Snapshot Administrator .rpm file to the remote server.
4. Install the package on the remote server.

```
$ rpm -i package-name
```

5. Validate the Junos Snapshot Administrator installation.

   You might need to restart the shell in order for the system to recognize the new command.

```
$ jsnap
```

   Junos Snapshot Administrator should output information about the tool syntax and options. If you did not get the expected output, correct the Junos Snapshot Administrator installation before going any further.

Satisfying Requirements for SSHv2 Connections

The NETCONF server communicates with client applications within the context of a NETCONF session. The server and client explicitly establish a connection and session before exchanging data, and close the session and connection when they are finished.

Junos Snapshot Administrator accesses the NETCONF server using the SSH protocol and uses the standard SSH authentication mechanism. To establish an SSHv2 connection with a device running Junos OS, you must ensure that the following requirements are met:

- The client application has a user account and can log in to each device where a NETCONF session will be established.
- The login account used by the client application has an SSH public/private key pair or a text-based password.
- The client application can access the public/private keys or text-based password.
- The NETCONF service over SSH is enabled on each device where a NETCONF session will be established.

For information about enabling NETCONF on a device running Junos OS and satisfying the requirements for establishing an SSH session, see "Controlling the NETCONF Session" in the NETCONF XML Management Protocol Developer Guide.

For information about NETCONF over SSH, see RFC 4742, *Using the NETCONF Configuration Protocol over Secure SHell (SSH)*, which is available at http://www.ietf.org/rfc/rfc4742.txt .

RELATED DOCUMENTATION

Junos Snapshot Administrator Overview | 2

3

CHAPTER

Junos Snapshot Administrator Configuration File

Junos Snapshot Administrator Configuration File

IN THIS SECTION

Understanding the Junos Snapshot Administrator Configuration File

IN THIS SECTION

The Junos Snapshot Administrator configuration file defines the scope of a snapshot and the evaluation criteria for either a single snapshot or a comparison of two snapshots. You provide the location of the Junos Snapshot Administrator configuration file as an argument to the **jsnap** command.

In the Junos Snapshot Administrator configuration file, curly braces delimit code blocks, and semicolons signal the end of a statement or command. You can insert comments into the configuration file by using either a hash (#) or a semicolon (;) at the start of a line.

The Junos Snapshot Administrator configuration file consists of a required **do** code block placed at the beginning of the configuration file followed by any number of user-defined test sections. Details of the individual configuration file components are described in the following sections:

do Section

The **do** code block lists the name of each test section that will be used in the snapshot. This section is mandatory and must be placed at the start of the configuration file. You can define as many test sections

as desired, but you must include the test section name in the **do** code block in order to execute that test. The syntax of the **do** code block is:

```
do {
     test-section-name1;
     test-section-name2;
     test-section-name3;
}
```

For example, the following **do** code block lists five test sections. When Junos Snapshot Administrator references the configuration file, although the configuration file might define more than five test sections, the tool only executes the five test sections listed in the **do** code block.

```
do {
     re0-master;
     ospf-checks;
     l2vpn-list;
     vpls-list;
     bgp-checks;
}
```

Test Sections

Within the configuration file, you define one or more test sections. The test sections, which are placed in any order after the **do** code block, define the scope of a snapshot and the evaluation criteria for a single snapshot or a comparison of two snapshots.

Each test section consists of a unique configuration stanza, which contains:

- The section name—Unique, user-defined string that should describe the check being performed.
- One **command** statement—Specifies the Junos OS *operational mode command* that is executed to collect the data.
- One or more **item** or **iterate** content section declarations—Defines the test cases used to evaluate the data.

The general syntax of the test section code block is:

```
test-section-name {
     command command-syntax;
     (item | iterate) xpath-expression {
```

```
            [id xpath-expression;]
            #   test cases
      }
}
```

For example:

```
re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
          is-equal mastership-state, "master" {
               info re0 is always master;
               err "  re0 is not master, rather %s", mastership-state;
               err "     Correct so that re0 is the master!";
          }
     }
}
```

The test section components are described in detail in the following sections:

Test Section Name

Test section names are unique, user-defined strings that should describe the check being performed. You add the names of the test sections that you want to execute to the **do** code block at the start of the configuration file. When Junos Snapshot Administrator references the configuration file, the tool executes any test sections that are listed in the **do** code block.

Ideally, you should create descriptive test section names so that you or anyone else working with the configuration file can quickly discern the scope and purpose of each test section long after the configuration file is created. For example, you might use **active-chassis-alarm-check** for a test section that checks whether there are any active chassis alarms, as shown here:

```
active-chassis-alarm-check {
   command show chassis alarms;
   item alarm-summary {
      exists no-active-alarm {
         info No chassis alarms;
         err "There are %s chassis alarms", active-alarm-count;
      }
   }
}
```

Test Section Command Statement

Each test section consists of a single **command** identifier followed by the Junos OS operational mode command that is executed to collect the desired data for that check. For example, if you want to collect and evaluate data about the OSPF neighbors for an interface, you would include the **command** identifier followed by the **show ospf neighbor** operational mode command, as shown here:

```
ospf-neighbor-check {
    command show ospf neighbor;
    iterate ospf-neighbor {
        id interface-name;
        #
        #  test cases
        #
    }
}
```

Provided the **ospf-neighbor-check** section name is included in the **do** code block, the resulting snapshot will include the XML data from the **show ospf neighbor** command output. For more information about Junos OS operational mode commands, see the *Junos XML API Operational Reference*. A sample of the **show ospf neighbor** output is shown here:

```
<ospf-neighbor-information>
        <ospf-neighbor>
            <neighbor-address>10.1.12.2</neighbor-address>
            <interface-name>ae18.0</interface-name>
            <ospf-neighbor-state>Full</ospf-neighbor-state>
            <neighbor-id>10.0.0.2</neighbor-id>
            <neighbor-priority>128</neighbor-priority>
            <activity-timer>3</activity-timer>
        </ospf-neighbor>
        <ospf-neighbor>
            <neighbor-address>10.1.15.2</neighbor-address>
            <interface-name>ae19.0</interface-name>
            <ospf-neighbor-state>Full</ospf-neighbor-state>
            <neighbor-id>10.0.0.5</neighbor-id>
            <neighbor-priority>128</neighbor-priority>
            <activity-timer>3</activity-timer>
        </ospf-neighbor>
    </ospf-neighbor-information>
```

You can evaluate specific elements in the snapshot output or compare elements across multiple snapshots by defining test cases.

Test Section Evaluation Criteria

The test section **command** statement can be followed by either an **item** or **iterate** content section declaration, within which you define one or more test cases for evaluating the captured snapshot data. You use the **item** statement to uniquely identify a specific element, and the **iterate** statement to iterate over multiple elements.

The general syntax for the content section declaration is:

```
(item | iterate) xpath-expression {
     test-operator operator-params {
     #specify an ID for test operators that compare two collections
     [id xpath-expression;]
          info string;
          err "string";
     }
}
```

Each test case is defined by a test operator followed by any required parameters. For more information about available test operators see "Understanding Junos Snapshot Administrator Test Operators" on page 22 and "Junos Snapshot Administrator Test Operators Summary" on page 25. Within the test-case code block you define an **info** statement to provide information about the test case and expected operating conditions. You also define one or more **err** statements, which are generated when the content fails the specific test case.

The following example has a test case that checks the XML output of the **show chassis routing-engine** command to determine if the Routing Engine in slot 0 is the primary Routing Engine. If the **mastership-state** of the Routing Engine in slot 0 is not equal to "master", the code generates two **err** statements.

```
re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
          is-equal mastership-state, "master" {
               info RE-0 is always master;
               err "  RE-0 is not master, rather %s", mastership-state;
               err "     Correct error so that RE-0 is the master";
          }
```

```
        }
}
```

When using certain test operators to compare element values across two snapshots, in order to map the first snapshot data item to a second snapshot data item, you must select elements of the data that create a unique ID. To create a unique ID for a test case, you include the **id** statement followed by an XPath expression that references the unique element. To create a unique ID based on multiple element values, you define multiple **id** statements. You can also construct ID values relative to the content value. For detailed information about creating ID values, see "Understanding Junos Snapshot Administrator Test Operators" on page 22.

Example: Creating the Junos Snapshot Administrator Configuration File

IN THIS SECTION

This example creates a basic sample Junos Snapshot Administrator configuration file.

Requirements

- Junos Snapshot Administrator Release 1.0 is installed on the server.

Overview

This example creates a Junos Snapshot Administrator configuration file with one test section named **re0-master**. The **re0-master** test section retrieves and parses the XML output from the Junos OS operational mode command **show chassis routing-engine**. A sample of the XML output from a dual Routing Engine device is shown here:

```
<route-engine-information xmlns="http://xml.juniper.net/junos/11.4R1/junos-
chassis">
  <route-engine>
      <slot>0</slot>
      <mastership-state>master</mastership-state>
```

```
      <mastership-priority>master (default)</mastership-priority>
      <status>OK</status>
      <temperature junos:celsius="30">30 degrees C / 86 degrees F</temperature>
      <cpu-temperature junos:celsius="27">
         27 degrees C / 80 degrees F
      </cpu-temperature>
      <memory-dram-size>768</memory-dram-size>
      <memory-buffer-utilization>48</memory-buffer-utilization>
      <cpu-user>1</cpu-user>
      <cpu-background>0</cpu-background>
      <cpu-system>5</cpu-system>
      <cpu-interrupt>1</cpu-interrupt>
      <cpu-idle>94</cpu-idle>
      <model>RE-5.0</model>
      <serial-number>19995858810</serial-number>
      <start-time junos:seconds="1337708989">
         2012-05-22 10:49:49 PDT
      </start-time>
      <up-time junos:seconds="8735869">
         101 days, 2 hours, 37 minutes, 49 seconds
      </up-time>
      <last-reboot-reason>
         Router rebooted after a normal shutdown.
      </last-reboot-reason>
      <load-average-one>0.00</load-average-one>
      <load-average-five>0.00</load-average-five>
      <load-average-fifteen>0.00</load-average-fifteen>
   </route-engine>
</route-engine-information>
```

For a dual Routing Engine device, the test case checks the XML output to determine if the Routing Engine in slot 0 is the primary Routing Engine. The test section uses the **item route-engine[slot = '0']** expression to select the **route-engine** element that has the child element **slot** value of "0". The test case uses the **is-equal** test operator to compare the value of the **mastership-state** child element with the string value "master". If the test case returns true, the Routing Engine in slot 0 is the primary Routing Engine. If the test case returns false, the code reports two error statements.

The mandatory **do** section includes the names of all test sections that should be executed when Junos Snapshot Administrator references this configuration file. For this example, the **do** section contains only **re0-master**.

Configuration

IN THIS SECTION

The Junos Snapshot Administrator configuration consists of a mandatory **do** section and one or more test sections.

How to Configure the Test Sections

Step-by-Step Procedure

The Junos Snapshot Administrator configuration consists of test sections that define the commands and evaluation criteria that are used in a snapshot or snapshot comparison. To configure a test section:

1. Name the test section using a unique and descriptive string.

```
re0-master {
    ...
}
```

2. Add the **command** statement, and specify the Junos OS operational mode command that the code executes to retrieve the desired XML data.

```
re0-master {
     command show chassis routing-engine;
}
```

3. Add the **iterate** or **item** statement followed by the XPath expression that selects the desired elements.

```
re0-master {
     command show chassis routing-engine;
```

```
     item route-engine[slot = '0'] {

     }
}
```

4. If the test section compares elements from two snapshots, add the **id** statement specifying a unique ID to map the first snapshot data item to the second snapshot data element.

   This example does not require an **id** statement.

5. Create the test case condition used during the check.

```
re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
          is-equal mastership-state, "master" {

          }
     }
}
```

6. Within the test case code block, add the **info** statement describing the test case or normal operating conditions.

```
re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
          is-equal mastership-state, "master" {
               info re0 is always master;

          }
     }
}
```

7. Within the test case code block, add one or more **err** statements, which are executed if the test case fails.

```
re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
```

```
          is-equal mastership-state, "master" {
               info re0 is always master;
               err "  re0 is not master, rather %s", mastership-state;
               err "     Correct so that re0 is the master!";
          }
     }
}
```

How to Configure the "do" Section

Step-by-Step Procedure

The Junos Snapshot Administrator configuration file must begin with a **do** code block that defines the test sections to be used in the snapshot.

1. At the beginning of the configuration file, add the **do** code block.

```
do {

}
```

2. Add the name of each test section that will be used in the snapshot.

```
do {
     re0-master;
}
```

Results

```
do {
    re0-master;
}

re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
          is-equal mastership-state, "master" {
               info re0 is always master;
               err "  re0 is not master, rather %s", mastership-state;
```

```
                err "     Correct so that re0 is the master!";
            }
        }
}
```

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Reference Variables | 18
- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25

Understanding Junos Snapshot Administrator Reference Variables

Junos Snapshot Administrator provides special variables that can be used within test case **err** statements to reference a specific data element as defined by an **id** statement XPath expression or to reference a data element in a specific snapshot. Available variables include:

- $ID.*num*—Variable mapped to a specific **id** statement that can be used in place of XPath expressions in the test case.
- $PRE—Anchor variable that is prepended to an XPath expression and that references data elements from the first snapshot.
- $POST—Anchor variable that is prepended to an XPath expression and that references data elements from the second snapshot.

In the following example, the **err** statements use **$ID.1** to reference the **connection-id** element, **$ID.2** to reference the **../local-site-id** element, and **$ID.3** to reference the **ancestor::instance/instance-name** element.

```
vpls-list {
     command show vpls connections up;
     iterate instance//connection {
          id connection-id;
          id ../local-site-id;
          id ancestor::instance/instance-name;
```

```
          list-not-less {
               info Check VPLS connections - all back up;
               err "  VPLS connection missing on service %s", $ID.3;
               err "    Site: %s, connection-id: %s", $ID.2, $ID.1;
               err "    Remote-PE: %s", remote-pe;
               err "    local-interface: %s", local-interface/interface-name;
          }
     }
}
```

The following example prepends the **$PRE** and **$POST** variables to **neighbor-address** in the test case **err** statements to reference the value of the OSPF neighbor address from the first and second snapshots, respectively.

```
ospf-check {
     command show ospf neighbor;
     iterate ospf-neighbor {
          id interface-name;
          no-diff neighbor-address {
               info OSPF neighbor change check;
               err "OSPF interface %s neighbor changed!", interface-name;
               err "   was going to %s, now going to %s", $PRE/neighbor-address,
$POST/neighbor-address;
          }
     }
}
```

RELATED DOCUMENTATION

Junos Snapshot Administrator Configuration File | 8

Understanding Junos Snapshot Administrator Test Operators | 22

4

CHAPTER

Junos Snapshot Administrator Test Operators

Understanding Junos Snapshot Administrator Test Operators

Junos Snapshot Administrator enables you to capture and audit runtime environment snapshots of your networked devices running Junos OS. The Junos Snapshot Administrator configuration file defines the scope of a snapshot and the evaluation criteria for either a single snapshot or a comparison of two snapshots. The Junos Snapshot Administrator **--snapcheck** option takes a single snapshot and evaluates the results, and the **--check** option compares the results of two separate snapshots. Within the configuration file, you can create test cases that evaluate or compare content from specific Junos OS operational mode commands. The test cases use test operators to either evaluate data elements in a single snapshot or compare data elements in two separate snapshots.

Junos Snapshot Administrator provides numerous relational operators that test for existence, equality, inequality, size, and inclusion in or exclusion from a specific range or list. Specific operators work with different operand types including strings, numbers, and XML elements. You should construct test cases using test operators that pertain to the type of check being performed. For a list of available operators, see "Junos Snapshot Administrator Test Operators Summary" on page 25.

Junos Snapshot Administrator uses a few test operators, **delta**, **list-no-less**, **list-no-more**, and **no-diff**, to compare elements or element values in two separate snapshots. Test cases using these test operators are executed when you use the **--check** option. When you use the **--snapcheck** option, which is specific to a single collection, test cases using these test operators are effectively ignored. Junos Snapshot Administrator outputs a message when a test case is ignored, as shown in the following sample output:

```
------------------------------------------------------------------------
CHECKING SECTION: ospf-checks
------------------------------------------------------------------------
INFO: snapcheck mode: skipping test: list-not-less
INFO: snapcheck mode: skipping test: no-diff
+ TEST PASSED: All OSPF neighbors are up
+ TEST PASSED: OSPF neighbors must have the same priority value
INFO: snapcheck mode: skipping test: no-diff
```

When comparing element values across two snapshots, in order to map the first snapshot data item to the second snapshot data item, you must select elements of the data that create a unique ID. To create a unique ID for a test case, you include the **id** statement followed by an XPath expression that references the unique element. To create a unique ID based on multiple element values, you define multiple **id** statements. You can also construct ID values relative to the content value.

Consider the following XML data for the **show ospf neighbor** *operational mode command*.

```
<ospf-neighbor-information>
     <ospf-neighbor>
          <neighbor-address>10.10.115.169</neighbor-address>
          <interface-name>ae18.0</interface-name>
          <ospf-neighbor-state>Full</ospf-neighbor-state>
          <neighbor-id>10.10.20.170</neighbor-id>
          <neighbor-priority>128</neighbor-priority>
          <activity-timer>3</activity-timer>
     </ospf-neighbor>
     <ospf-neighbor>
          <neighbor-address>10.10.115.174</neighbor-address>
          <interface-name>ae19.0</interface-name>
          <ospf-neighbor-state>Full</ospf-neighbor-state>
          <neighbor-id>10.10.20.168</neighbor-id>
          <neighbor-priority>128</neighbor-priority>
          <activity-timer>3</activity-timer>
     </ospf-neighbor>
</ospf-neighbor-information>
```

For this case, you can create a unique ID using the value of the **interface-name** element, which is a child element of **ospf-neighbor**. You specify the ID using the **id** statement followed by the **interface-name** XPath expression.

```
ospf-neighbor-check {
     command show ospf neighbor;
     iterate ospf-neighbor {
          id interface-name;
          #
          #  test-cases
          #
     }
}
```

To create a unique ID based on multiple element values, you define multiple **id** statements. The following example uniquely identifies an OSPF neighbor by the values of both the **interface-name** and the **neighbor-id** elements:

```
ospf-neighbor-check {
     command show ospf neighbor;
```

```
      iterate ospf-neighbor {
           id interface-name;
           id neighbor-id;
           #
           #  test-cases
           #
      }
}
```

You can also construct ID values relative to the content value. The following example creates a unique ID for a VPLS connection using three values: the VPLS **instance-name**, the **local-site-id** within the VPLS service, and the **connection-id** of the connection.

```
vpls-list {
     command show vpls connections up;
     iterate instance//connection {
          id connection-id;
          id ../local-site-id;
          id ancestor::instance/instance-name;
          list-not-less {
               info Check VPLS connections - all back up;
               err "  VPLS connection missing on service %s", $ID.3;
               err "    Site: %s, connection-id: %s", $ID.2, $ID.1;
               err "    Remote-PE: %s", remote-pe;
               err "    local-interface: %s", local-interface/interface-name;
          }
     }
}
```

The previous code defines the content data as the set of connection items, which are selected by the XPath expression **instance//connection**. Each connection has a **connection-id**, which defines the first **id** value. Each connection has a parent element **local-site**, which has a **local-site-id** child element. Therefore, the second **id** value is **../local-site-id**. The third **id** value, **ancestor::instance/instance-name**, is an XPath expression that means: starting from the connection, go up through the XML parent hierarchy (ancestor) until you find the **instance** element, and then select the **instance-name** value. The result is the VPLS service name.

Junos Snapshot Administrator uses special **$ID.*num*** variables that map to the **id** statements in a test section. When you define one or more **id** statements in a test section, you can use these variables to refer to the XPath expression throughout that test case. This is useful when the XPath expression gets too long. Note the use of the **$ID** variables in the **err** statements in the previous example. For more

information about Junos Snapshot Administrator variables, see "Understanding Junos Snapshot Administrator Reference Variables" on page 18.

RELATED DOCUMENTATION

Junos Snapshot Administrator Test Operators Summary | 25

Junos Snapshot Administrator Configuration File | 8

Understanding Junos Snapshot Administrator Reference Variables | 18

Junos Snapshot Administrator Test Operators Summary

Junos Snapshot Administrator enables you to capture and audit runtime environment snapshots of your networked devices running Junos OS. The Junos Snapshot Administrator configuration file defines the scope of a snapshot and the evaluation criteria for either a single snapshot or a comparison of two snapshots. Within the configuration file, you can create test cases that evaluate or compare content from specific Junos OS operational mode commands. The test cases use test operators to either evaluate data elements in a single snapshot or compare data elements in two separate snapshots.

Table 1 on page 25 lists the Junos Snapshot Administrator test operators along with a brief description of each operator. Operators are grouped by operand type.

Table 1: Junos Snapshot Administrator Test Operators

Operator	Description
Compare Elements or Element Values in Two Snapshots	
`delta`	Compare the change in value of a specified data element, which must be present in both snapshots, to a specified delta. You can specify the delta as an absolute, positive, or negative percentage, or an absolute, positive, or negative fixed value.
`list-not-less`	Determine if the specified items are present in the first snapshot but are not present in the second snapshot.

Table 1: Junos Snapshot Administrator Test Operators *(Continued)*

Operator	Description
`list-not-more`	Determine if the specified items are present in the second snapshot but are not present in the first snapshot.
`no-diff`	Compare specified data elements that are present in both snapshots, and verify that the value is the same.
Operate on Elements with Numeric or String Values	
`all-same`	Check if all content values for the specified elements are the same. Optionally, you can check if all content values for the specified elements are the same as the content value of a reference item.
`is-equal`	Test if an XML element string or integer value matches a given value.
`not-equal`	Test if an XML element string or integer value does not match a given value.
Operate on Elements with Numeric Values	
`in-range`	Test if the XML element value is within a given numeric range.
`is-gt`	Test if the XML element value is greater than a given numeric value.
`is-lt`	Test if the XML element value is less than a given numeric value.
`not-range`	Test if the XML element value is outside a given numeric range.
Operate on Elements with String Values	
`contains`	Determine if an XML element string value contains the provided string value.

Table 1: Junos Snapshot Administrator Test Operators *(Continued)*

Operator	Description
`is-in`	Determine if an XML element string value is included in a specified list of string values.
`not-in`	Determine if an XML element string value is excluded from a specified list of string values.
Operate on XML Elements	
`exists`	Verify the existence of an XML element in the snapshot.
`not-exists`	Verify the lack of existence of an XML element in the snapshot.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Overview | 2
- Junos Snapshot Administrator Configuration File | 8

all-same operator

IN THIS SECTION

Syntax

```
all-same xpath-expression, [ [reference-item] ] {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that checks if all content values for the specified elements are the same. If you include the optional reference item, the operator checks if all content values for the specified elements are the same as the content value of the reference item.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
reference-item	(Optional) XPath expression specifying the reference element against which value all other element values are compared.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following example code checks that all OSPF neighbors have the same priority value. If any priority values are different, the code generates the error message.

```
ospf-check {
   command show ospf neighbor;
   iterate ospf-neighbor {
      all-same neighbor-priority {
```

```
          info OSPF neighbors must have the same priority value;
          err "OSPF interface %s has mismatch priority %s", interface-name,
neighbor-priority;
       }
    }
}
```

The following example code checks that all OSPF neighbors have the same priority value as that of interface ae19.0. If any priority values are different from that of the reference interface, the code generates the error message.

```
ospf-check {
   command show ospf neighbor;
   iterate ospf-neighbor {
      all-same neighbor-priority, [interface-name = 'ae19.0']  {
         info OSPF neighbors must have the same priority value as ae19;
         err "OSPF interface %s has mismatch priority %s", interface-name,
neighbor-priority;
      }
   }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- is-equal operator | 39
- not-equal operator | 53

contains operator

IN THIS SECTION

Syntax

```
contains xpath-expression, test-string {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that determines if an XML element string value contains the provided *test-string* value.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
test-string	String searched for in the XML element value. Enclose the string in quotation marks.

xpath-expression XPath expression selecting the elements to evaluate.

Usage Examples

Given the following partial XML output from the **show version** operational mode command:

```
<multi-routing-engine-results>
     <multi-routing-engine-item>
          <re-name>re0</re-name>
          <software-information>
               <host-name>R1</host-name>
               <product-model>mx960</product-model>
               <product-name>mx960</product-name>
               <package-information>
                    <name>junos</name>
                    <comment>JUNOS Base OS boot [10.4R7.5]</comment>
               </package-information>
               <package-information>
                    <name>jbase</name>
                    <comment>JUNOS Base OS Software Suite [10.4R7.5]</comment>
               </package-information>
...
```

The following test case selects the first **package-information** node set, and checks the child **comment** element for a Junos OS release number of 10.4. If the 10.4 release string is not found, the code generates the error message. The error string includes the comment value containing the release number found on that Routing Engine.

```
version-check {
   command show version invoke-on all-routing-engines;
   iterate //software-information {
      contains package-information[1]/comment, "10.4R" {
         info Checking Junos version for 10.4 release;
         err "Found %s on RE %s", package-information[1]/comment, ../re-name;
      }
   }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- is-in operator | 43
- not-in operator | 57

delta operator

IN THIS SECTION

Syntax

```
id id;
delta xpath-expression, delta-value {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that compares the change in value of a specified data element, which must be present in both snapshots, to a specified delta. You can specify the delta as an absolute, positive, or negative percentage, or as an absolute, positive, or negative fixed value.

Parameters

delta-value Delta value expressed as a percentage or fixed value and against which the change in the element value is compared.

Expressed as a percentage:

- Absolute percentage—Indicates that either an increase or decrease in the value greater than the absolute percentage is an error, for example, 10%.
- Positive percentage—Indicates that only an increase in the value greater than the absolute percentage is an error, for example, +10%.
- Negative percentage—Indicates that only a decrease in the value greater than the absolute percentage is an error, for example, -10%.

Expressed as a fixed value:

- Absolute fixed value—Indicates that either an increase or decrease in the value greater than the absolute value is an error, for example, 500.
- Positive fixed value—Indicates that only an increase in the value greater than the absolute value is an error, for example, +500.
- Negative fixed value—Indicates that only a decrease in the value greater than the absolute value is an error, for example, -500.

err ***string*** Statement generated when the test case returns false.

id ***id*** XPath expression relative to the data content that specifies a unique data element that maps the first snapshot data item to the second snapshot data item. To create a unique ID based on multiple element values, define multiple **id** statements.

info ***string*** Description of the test case.

xpath-expression XPath expression selecting the elements to evaluate.

Usage Examples

The following test cases check the BGP route prefix count after a maintenance window. To ensure that the BGP peers were restored, the code checks the prefix counts for a delta change of -10%. That is, if the new prefix count is less than 90% of the original prefix count, the test reports an error.

```
bgp-checks {
   command show bgp summary;
   iterate bgp-rib {
      id name;
      delta total-prefix-count, -10% {
         info BGP total prefix count should not change by more than -10%;
         err "  BGP rib: %s total prefix count has exceeded threshold", name;
         err "     pre-check: %s, post-check: %s", $PRE/total-prefix-count,
$POST/total-prefix-count;
      }
      delta active-prefix-count, -10% {
         info BGP active prefix count should not change by more than -10%;
         err "  BGP rib: %s total prefix count has exceeded threshold", name;
         err "     pre-check: %s, post-check: %s", $PRE/active-prefix-count,
$POST/active-prefix-count;
      }
   }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

Understanding Junos Snapshot Administrator Test Operators | 22

Junos Snapshot Administrator Test Operators Summary | 25

list-not-less operator | 47

list-not more operator | 49

no-diff operator | 51

exists operator

IN THIS SECTION

Syntax

```
exists xpath-expression {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that verifies the existence of an XML element in the snapshot.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
xpath-expression	XPath expression selecting the elements to test.

Usage Examples

The following test case checks for active chassis alarms. If the **no-active-alarm** element exists, there are no active chassis alarms. Otherwise, the code reports an error indicating that there are active alarms.

```
alarm-checks {
   command show chassis alarms;
   item alarm-summary {
      exists no-active-alarm {
         info No chassis alarms;
         err "There are %s chassis alarms", active-alarm-count;
      }
   }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- not-exists operator | 55

in-range operator

IN THIS SECTION

Syntax

```
in-range xpath-expression, integer-start, integer-end {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that tests if an XML element value is within a given numeric range.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
integer-start	Numeric value defining the start of the range.
integer-end	Numeric value defining the end of the range.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following test case checks the OSPF database and tests if each router has between 5 and 10 links. If the **link-count** value falls outside the specified range, the code reports an error.

```
ospf-db-checks {
     command show ospf database detail;
     iterate //ospf-router-lsa {
          in-range link-count, 5, 10 {
               info OSPF router links [5 - 10];
              err "Router %s has %s links", ../advertising-router, link-count;
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- is-gt operator | 41
- is-lt operator | 45
- not-range operator | 59

is-equal operator

IN THIS SECTION

Syntax

```
is-equal xpath-expression, value {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that tests if the XML element string or integer value matches a given value.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.

value	String or integer value against which the element value is compared. Enclose string values in quotation marks.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following test case prints an error if the Routing Engine in slot 0 (re0) is not the primary Routing Engine.

```
re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
          is-equal mastership-state, "master" {
               info re0 is always master;
               err "  re0 is not master, rather %s", mastership-state;
               err "     Correct so that re0 is the master Routing Engine!";
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

Understanding Junos Snapshot Administrator Test Operators | 22

Junos Snapshot Administrator Test Operators Summary | 25

all-same operator | 27

not-equal operator | 53

is-gt operator

IN THIS SECTION

Syntax

```
is-gt xpath-expression, integer-value {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that tests if an XML element value is greater than a given numeric value.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
integer-value	Numeric value against which to compare the XML element value.

xpath-expression XPath expression selecting the elements to evaluate.

Usage Examples

The following test case checks that each OSPF interface has at least 1 neighbor by verifying that the **neighbor-count** is greater than 0. If the neighbor count is 0, the code reports an error.

```
ospf-int-checks {
     command show ospf interface;
     iterate ospf-interface {
          is-gt neighbor-count, 0 {
               info OSPF interfaces must have at least 1 neighbor;
               err "OSPF interface %s does not have any neighbors", interface-
name;
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

Understanding Junos Snapshot Administrator Test Operators | 22

Junos Snapshot Administrator Test Operators Summary | 25

in-range operator | 36

is-lt operator | 45

not-range operator | 59

is-in operator

IN THIS SECTION

Syntax

```
is-in xpath-expression, string-list {
     info string;
     err "string";
    [err "string"];
}
```

Description

Junos Snapshot Administrator test operator that determines if an XML element string value is included in the specified list of string values.

Parameters

err *string* Statement generated when the test case returns false.

info *string* Description of the test case.

string-list	Comma-separated list of strings against which to compare the XML element value for inclusion. Enclose each string in quotation marks.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

Given the following XML output from the **show rsvp session** operational mode command:

```
<rsvp-session-data>
     <session-type>Ingress</session-type>
     <count>3</count>
     <rsvp-session junos:style="brief">
          <destination-address>10.255.20.137</destination-address>
          <source-address>10.255.20.167</source-address>
          <lsp-state>Dn</lsp-state>
          <route-count>0</route-count>
          <rsb-count>0</rsb-count>
          <resv-style>-</resv-style>
          <label-in>-</label-in>
          <label-out>-</label-out>
          <name>test</name>
     </rsvp-session>
...
```

The following test case checks if the RSVP session **lsp-state** has a value of either **Up** or **NotInService**. If the **lsp-state** value is not in the specified string list, the code reports an error.

```
rsvp-checks {
     command show rsvp session;
     iterate rsvp-session-data/rsvp-session {
          is-in lsp-state, "Up", "NotInService" {
               info RSVP LSP state is [Up | NotInService];
               err " RSVP session to %s has LSP state %s.", destination-address,
lsp-state;
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- contains operator | 30
- not-in operator | 57

is-lt operator

IN THIS SECTION

Syntax

```
is-lt xpath-expression, integer-value {
     info string;
     err "string";
    [err "string"];
}
```

Description

Junos Snapshot Administrator test operator that tests if an XML element value is less than a given numeric value.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
integer-value	Numeric value against which to compare the XML element value.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following test case checks whether a BGP peer flaps by testing if the **flap-count** is less than 5. If the **flap-count** is greater than 5, the code reports an error.

```
bgp-peer-checks {
     command show bgp neighbor;
     iterate bgp-peer {
          is-lt flap-count, 5 {
               info BGP peer flap-count < 5;
               err "BGP peer %s has %s flaps", peer-address, flap-count;
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- in-range operator | 36
- is-gt operator | 41
- not-range operator | 59

list-not-less operator

IN THIS SECTION

Syntax

```
id id;
list-not-less {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that determines if the specified XML elements are present in the first snapshot but are not present in the second snapshot. The **list-not-less** test-operator validates the existence in the second snapshot of the elements defined by the **id** statement.

Parameters

err *string* Statement generated when the test case returns false.

id *id* XPath expression relative to the data content that specifies a unique data element that maps the first snapshot data item to the second snapshot data item. To create a unique ID based on multiple element values, define multiple **id** statements.

info *string* Description of the test case.

Usage Examples

The following test case checks if the OSPF neighbors that existed before the device maintenance are present after the device maintenance. If any OSPF neighbors present in the first snapshot are missing from the second snapshot, the code reports an error.

```
ospf-check {
     command show ospf neighbor;
     iterate ospf-neighbor {
          id interface-name;
          list-not-less {
               info OSPF interface list check;
               err "OSPF interface gone missing: %s going to %s", interface-
name, neighbor-address;
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

Understanding Junos Snapshot Administrator Test Operators | 22
Junos Snapshot Administrator Test Operators Summary | 25
delta operator | 32
list-not more operator | 49
no-diff operator | 51

list-not more operator

IN THIS SECTION

Syntax

```
id id;
list-not-more {
     info string;
     err "string";
    [err "string";]
```

```
}
```

Description

Junos Snapshot Administrator test operator that determines if the specified XML elements are present in the second snapshot but are not present in the first snapshot. The **list-not-more** test-operator validates the existence in the first snapshot of the elements defined by the **id** statement.

Parameters

err *string* Statement generated when the test case returns false.

id *id* XPath expression relative to the data content that specifies a unique data element that maps the first snapshot data item to the second snapshot data item. To create a unique ID based on multiple element values, define multiple **id** statements.

info *string* Description of the test case.

Usage Examples

The following example check chassis alarms. This configuration uses both **list-not-less** and **list-not-more** to check for any changes in the alarms. The data elements referenced in the **list-not-more err** section are elements from the second snapshot that were not present in the first snapshot.

```
alarm-checks {
     command show chassis alarms;
     item alarm-summary {
          not-exists active-alarm-count {
               info No chassis alarms;
               err "There are %s chassis alarms", active-alarm-count;
          }
     }
     iterate alarm-detail {
          id alarm-description;
```

```
            list-not-less {
                 info Alarm Gone Missing;
                 err "-Alarm: %s", alarm-description;
      }
            list-not-more {
                 info Alarm Got More;
                 err "+Alarm: %s", alarm-description;
            }
      }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- delta operator | 32
- list-not-less operator | 47
- no-diff operator | 51

no-diff operator

IN THIS SECTION

Syntax

```
id id;
no-diff xpath-expression {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that compares specified data elements that are present in both the first and second snapshot collections and verifies that the value is the same.

Parameters

err *string*	Statement generated when the test case returns false.
id *id*	XPath expression relative to the data content that specifies a unique data element that maps the first snapshot data item to the second snapshot data item. To create a unique ID based on multiple element values, define multiple **id** statements.
info *string*	Description of the test case.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following test case reports an error if the OSPF neighbor address has changed. The **$PRE** and **$POST** variables in the second **err** statement reference the first and second data collections, respectively.

```
ospf-check {
     command show ospf neighbor;
```

```
      iterate ospf-neighbor {
           id interface-name;
           no-diff neighbor-address {
                info OSPF neighbor change check;
                err "OSPF interface %s neighbor changed!", interface-name;
                err "   was going to %s, now going to %s", $PRE/neighbor-address,
$POST/neighbor-address;
           }
      }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- delta operator | 32
- list-not-less operator | 47
- list-not more operator | 49

not-equal operator

IN THIS SECTION

Syntax

```
not-equal xpath-expression, value {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that tests if the XML element string or integer value does not match a given value.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
value	String or integer value against which the element value is compared. Enclose string values in quotation marks.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following test case prints an error if the Routing Engine in slot 0 (re0) is the primary Routing Engine.

```
re0-master {
     command show chassis routing-engine;
     item route-engine[slot = '0'] {
          not-equal mastership-state, "master" {
               info re0 must not be master;
               err "     Correct so that re0 is not  the master Routing Engine!";
```

```
            }
        }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- all-same operator | 27
- is-equal operator | 39

not-exists operator

IN THIS SECTION

Syntax

```
not-exists xpath-expression {
     info string;
```

```
      err "string";
     [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that verifies the lack of existence of an XML element in the snapshot.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
xpath-expression	XPath expression specifying the elements to test.

Usage Examples

The following test case checks for active chassis alarms. If the **active-alarm-count** element does not exist, there are no active chassis alarms. Otherwise, the code reports an error indicating that there are active alarms.

```
alarm-checks {
   command show chassis alarms;
   item alarm-summary {
      not-exists active-alarm-count {
         info No chassis alarms;
         err "There are %s chassis alarms", active-alarm-count;
      }
   }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

Understanding Junos Snapshot Administrator Test Operators | 22

Junos Snapshot Administrator Test Operators Summary | 25

exists operator | 35

not-in operator

IN THIS SECTION

Syntax

```
not-in xpath-expression, string-list {
     info string;
     err "string";
    [err "string";]

}
```

Description

Junos Snapshot Administrator test operator that determines if an XML element string value is excluded from the specified list of string values.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
string-list	Comma-separated list of strings against which to compare the XML element value for exclusion. Enclose each string in quotation marks.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following test case checks that the RSVP session **lsp-state** element does not have a value of **Dn** or **Failed**. If the **lsp-state** value is in the specified string list, the code reports an error.

```
rsvp-checks {
     command show rsvp session;
     iterate rsvp-session-data/rsvp-session {
          not-in lsp-state, "Dn", "Failed" {
               info RSVP LSP state is not [Dn | Failed];
               err " RSVP session to %s has LSP state %s.", destination-address,
lsp-state;
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

Understanding Junos Snapshot Administrator Test Operators | 22
Junos Snapshot Administrator Test Operators Summary | 25
contains operator | 30
is-in operator | 43

not-range operator

IN THIS SECTION

Syntax

```
not-range xpath-expression, integer-start, integer-end {
     info string;
     err "string";
    [err "string";]
}
```

Description

Junos Snapshot Administrator test operator that tests if an XML element value is outside of a given numeric range.

Parameters

err *string*	Statement generated when the test case returns false.
info *string*	Description of the test case.
integer-start	Numeric value defining the start of the range.
integer-end	Numeric value defining the end of the range.
xpath-expression	XPath expression selecting the elements to evaluate.

Usage Examples

The following example checks the OSPF database and tests if each router has either less than 5 links or greater than 10 links. If the **link-count** value falls inside the specified range, the code reports an error.

```
ospf-db-checks {
     command show ospf database detail;
     iterate //ospf-router-lsa {
          not-range link-count, 5, 10 {
               info OSPF router links < 5 or > 10;
               err "Router %s has %s links", ../advertising-router, link-count;
          }
     }
}
```

Release Information

Operator introduced in Junos Snapshot Administrator Release 1.0.

RELATED DOCUMENTATION

- Understanding Junos Snapshot Administrator Test Operators | 22
- Junos Snapshot Administrator Test Operators Summary | 25
- in-range operator | 36
- is-gt operator | 41
- is-lt operator | 45

5

CHAPTER

Using Junos Snapshot Administrator

Using Junos Snapshot Administrator

IN THIS SECTION

You can use Junos Snapshot Administrator on a device running Junos OS to capture and save a runtime environment snapshot, compare two snapshots, or capture a snapshot and immediately evaluate it.

When you take a snapshot, you provide a snapshot name. Junos Snapshot Administrator uses the snapshot name, target device name, and configuration file test section strings to generate output filenames that uniquely identify that data. For example, say you are collecting data from device ABC, you define the snapshot name **s1**, and your configuration file has three test sections named **ospf-checks**, **bgp-checks**, and **version-check**. When you take a snapshot, Junos Snapshot Administrator creates the following output files:

- **ABC__ospf-checks__s1.xml**
- **ABC__bgp-checks__s1.xml**
- **ABC__version-check__s1.xml**

The following sections outline the Junos Snapshot Administrator commands.

How to Take a Snapshot

To take a snapshot of a device, enter the following on the remote server's command line:

```
user@server$ jsnap --snap snapshot-name -l username -t device-name configuration-filename
```

The command parameters are:

- ***snapshot-name***—String used in the output filenames to uniquely identify that snapshot.
- ***username***—Username used to access the device running Junos OS.

- ***device-name***—Name or IP address of the device you are accessing.
- ***configuration-filename***—Snapshot configuration filename.

For example, the user bsmith has an login account on router ABC. Prior to a maintenance upgrade, user bsmith takes a snapshot of the device. The snapshot name is **preupgrade** and the configuration filename is **ABCsnapshot.conf**. Upon connecting, the device prompts for the user's password.

```
bsmith@server$ jsnap --snap preupgrade -l bsmith -t ABC ABCsnapshot.conf
```

```
bsmith password:
Connecting to bsmith@ABC ...
CONNECTED.
EXEC: 'show chassis alarms' ...
SAVE: 'ABC__alarm-checks__preugrade.xml' ...
EXEC: 'show ospf interface' ...
SAVE: 'ABC__ospf-int-checks__preugrade.xml' ...
```

Additionally, you can take a snapshot using only a single test section from the configuration file. To take a snapshot of a device using only a single test section, include the **-s *section-name*** argument, and specify the test section name.

```
user@server$ jsnap --snap snapshot-name -l username -t device-name -s section-name configuration-filename
```

How to Compare Two Snapshots

To compare two existing snapshot collections, enter the following on the remote server's command line:

```
user@server$ jsnap --check snapshot1,snapshot2 -t device-name configuration-filename
```

The command parameters are:

- ***snapshot1***–String used in the output filename to uniquely identify the first snapshot.
- ***snapshot2***—String used in the output filename to uniquely identify the second snapshot.
- ***device-name***—Name or IP address of the device.

- ***configuration-filename***–Snapshot configuration filename.

For example, the user bsmith has an login account on router ABC. Prior to and immediately following a maintenance upgrade, user bsmith takes a snapshot of the device. The snapshot names are **preupgrade** and **postupgrade**. To compare these two snapshots using the criteria defined in the configuration file **ABCsnapshot.conf**, bsmith issues the following command:

```
bsmith@server$ jsnap --check preupgrade,postupgrade -t ABC ABCsnapshot.conf
```

The Junos Snapshot Administrator output displays the target router and the test results for each of the active test sections in the configuration file. Sample output is shown here:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
>>>
>>> TARGET: ABC
>>>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
------------------------------------------------------------------------------
CHECKING SECTION: alarm-checks
------------------------------------------------------------------------------
- TEST FAILED: No chassis alarms

There are 1 chassis alarms

- TEST FAILED: 1

Alarm Description: Management Ethernet Link Down

+ TEST PASSED: Alarm Gone Missing
- TEST FAILED: Alarm Got More

+Alarm: Management Ethernet Link Down

------------------------------------------------------------------------------
CHECKING SECTION: ospf-int-checks
------------------------------------------------------------------------------
+ TEST PASSED: OSPF interfaces must have at least 1 neighbor
```

Additionally, you can compare two snapshots using only a single test section from the configuration file. To compare two snapshots using only a single test section, include the **-s *section-name*** argument, and specify the test section name.

```
user@server$ jsnap --check snapshot1,snapshot2 -t device-name -s section-name configuration-filename
```

How to Take and Evaluate a Snapshot

To take a snapshot and immediately evaluate it based on a pre-defined set of criteria, enter the following on the remote server's command line:

```
user@server$ jsnap --snapcheck snapshot-name -l username -t device-name configuration-filename
```

The command parameters are:

- ***snapshot-name***—String used in the output filenames to uniquely identify that snapshot.
- ***username***—Username used to access the device running Junos OS.
- ***device-name***—Name or IP address of the device you are accessing.
- ***configuration-filename***—Snapshot configuration filename.

RELATED DOCUMENTATION

Junos Snapshot Administrator Configuration File | 8