

Paragon Insights User Guide

Published
2024-08-21

RELEASE
4.3.0

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

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, 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.

Paragon Insights User Guide

4.3.0

Copyright © 2024 Juniper Networks, Inc. All rights reserved.

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 | viii

1

Introduction to Paragon Insights

Paragon Insights Overview | 2

Paragon Insights Concepts | 6

Paragon Insights Data Collection Methods | 7

Paragon Insights Topics | 9

Paragon Insights Rules - Basics | 10

Paragon Insights Rules - Deep Dive | 12

Paragon Insights Playbooks | 31

Paragon Insights Tagging | 33

Overview | 33

Types of Tagging | 40

Add a Tagging Profile | 48

Apply a Tagging Profile | 53

Delete a Tagging Profile | 55

Paragon Insights Time Series Database (TSDB) | 57

Paragon Insights Machine Learning (ML) | 65

Paragon Insights Machine Learning Overview | 65

Understanding Paragon Insights Anomaly Detection | 66

Understanding Paragon Insights Outlier Detection | 68

Understanding Paragon Insights Predict | 72

Paragon Insights Rule Examples | 73

Frequency Profiles and Offset Time | 87

Frequency Profiles | 87

Offset Time Unit | 94

Paragon Insights Management and Monitoring

Manage Paragon Insights Users and Groups | 107

Manage Devices, Device Groups, and Network Groups | 121

Adding a Device | 122

Editing a Device | 129

Adding a Device Group | 129

Editing a Device Group | 136

Configuring a Retention Policy for the Time Series Database | 136

Adding a Network Group | 137

Editing a Network Group | 140

Paragon Insights Rules and Playbooks | 141

Add a Pre-Defined Rule | 141

Create a New Rule Using the Paragon Insights GUI | 142

Edit a Rule | 159

Add a Pre-Defined Playbook | 159

Create a New Playbook Using the Paragon Insights GUI | 160

Edit a Playbook | 161

Clone a Playbook | 162

Manage Playbook Instances | 163

Monitor Device and Network Health | 172

Dashboard | 172

Health | 183

Network Health | 192

Graphs Page | 192

Understand Resources and Dependencies | 206

About the Resources Page | 209

Add Resources for Root Cause Analysis | 212

Configure Dependency in Resources | 215

Example Configuration: OSPF Resource and Dependency | 221

Edit Resources and Dependencies | 232

 Edit a Resource | 232

 Edit Resource Dependency | 233

Filter Resources | 234

Upload Resources | 235

Download Resources | 236

Clone Resources | 236

Delete Resources and Dependencies | 237

 Delete a Resource | 238

 Delete Resource Dependency | 239

Monitor Network Device Health Using Grafana | 239

 Grafana Overview | 239

 Access the Grafana UI | 240

 Run a Query | 240

 View Prepopulated Graphs | 242

 Back Up and Restore Grafana Data | 243

Understanding Action Engine Workflows | 244

Manage Action Engine Workflows | 244

Alerts and Notifications | 252

 Generate Alert Notifications | 252

 Configure a Notification Profile | 253

 Enable Alert Notifications for a Device Group or Network Group | 259

Manage Alerts Using Alert Manager | 260

Viewing Alerts | 260

Manage Individual Alerts | 262

Configure Alert Blackouts | 263

Stream Sensor and Field Data from Paragon Insights | 264

Configure the Notification Type for Publishing | 264

Publish Data for a Device Group or Network Group | 267

Generate Reports | 268

Use Exim4 for E-Mails | 282

Configure the Exim4 Agent to Send E-mail | 283

Manage Audit Logs | 284

Filter Audit Logs | 284

Export Audit Logs | 285

Paragon Insights Commands and Audit Logs | 286

Configure a Secure Data Connection for Paragon Insights Devices | 286

Configure Data Summarization | 290

Modify the UDA, UDF, and Workflow Engines | 299

Commit or Roll Back Configuration Changes in Paragon Insights | 307

Logs for Paragon Insights Services | 309

Troubleshooting | 312

Paragon Insights Configuration – Backup and Restore | 321

Back Up the Configuration | 322

Restore the Configuration | 322

Backup or Restore the Time Series Database (TSDB) | 323

License Management

Paragon Insights Licensing Overview | 326

View, Add, or Delete Paragon Insights Licenses | 326

Add a Paragon Insights License | 327

Delete a Paragon Insights License | 328

View Paragon Insights Licensing Features | 328

View Status and Details of Paragon Insights License | 329

About This Guide

Use this guide to understand the features you can configure and the tasks you can perform from the Paragon Insights (formerly HealthBot) GUI.

1

CHAPTER

Introduction to Paragon Insights

[Paragon Insights Overview](#) | 2

[Paragon Insights Concepts](#) | 6

[Paragon Insights Tagging](#) | 33

[Paragon Insights Time Series Database \(TSDB\)](#) | 57

[Paragon Insights Machine Learning \(ML\)](#) | 65

[Frequency Profiles and Offset Time](#) | 87

Paragon Insights Overview

IN THIS SECTION

- [Main Components of Paragon Insights | 2](#)
- [Closed-Loop Automation | 4](#)
- [Benefits of Paragon Insights | 6](#)

Paragon Insights (formerly HealthBot) is a highly automated and programmable device-level diagnostics and network analytics tool that provides consistent and coherent operational intelligence across network deployments. Paragon Insights integrates multiple data collection methods (such as Junos Telemetry Interface (JTI), NETCONF, syslog, and SNMP) to aggregate and correlate large volumes of time-sensitive telemetry data, thereby providing a multidimensional and predictive view of the network. Additionally, Paragon Insights translates troubleshooting, maintenance, and real-time analytics into an intuitive user experience to give network operators actionable insights into the health of individual devices and of the overall network.

Main Components of Paragon Insights

Paragon Insights consists of two main components:

- Health Monitoring, to view an abstracted, hierarchical representation of device and network-level health, and define the health parameters of key network elements through customizable key performance indicators (KPIs), rules, and playbooks. A playbook is a collection of rules. You can create a playbook and apply the playbook to a device group or a network group. For more information on rules and playbooks, see ["Paragon Insights Rules and Playbooks" on page 141](#).
- Root Cause Analysis, which helps you find the root cause of a device or network-level issue when Paragon Insights detects a problem with a network element.

Paragon Insights Health Monitoring

The Challenge

With increasing data traffic generated by cloud-native applications and emerging technologies, service providers and enterprises need a network analytics solution to analyze volumes of telemetry data, offer insights into overall network health, and produce actionable intelligence. While although telemetry-based techniques have existed for years, the growing number of protocols, data formats, and KPIs from diverse networking devices has made data analysis complex and costly. Traditional CLI-based interfaces require specialized skills to extract business value from telemetry data, creating a barrier to entry for network analytics

How Paragon Insights Health Monitoring Helps

By aggregating and correlating raw telemetry data from multiple sources, the Paragon Insights Health Monitoring component provides a multidimensional view of network health that reports current status, as well as projected threats to the network infrastructure and its workloads.

Health status determination is tightly integrated with the Paragon Insights RCA component, which can make use of system log data received from the network and its devices. Paragon Insights Health Monitoring provides status indicators that alert you when a network resource is currently operating outside a user-defined performance policy. Paragon Insights Health Monitoring does a risk analysis using historical trends and predicts whether a resource may be unhealthy in the future. Paragon Insights Health Monitoring not only offers a fully customizable view of the current health of network elements, but also automatically initiates remedial actions based on predefined service level agreements (SLAs).

Defining the health of a network element, such as broadband network gateway (BNG), provider edge (PE), core, and leaf-spine, is highly contextual. Each element plays a different role in a network, with unique KPIs to monitor. Given that there is no single definition for network health across all use cases, Paragon Insights provides a highly customizable framework to allow you to define your own health profiles.

Paragon Insights Root Cause Analysis

The Challenge

For some network issues, it can be challenging for network operators to figure out what caused a networking device to stop working properly. In such cases, an operator must call on a specialist (with knowledge built from years of experience) to troubleshoot the problem and find the root cause.

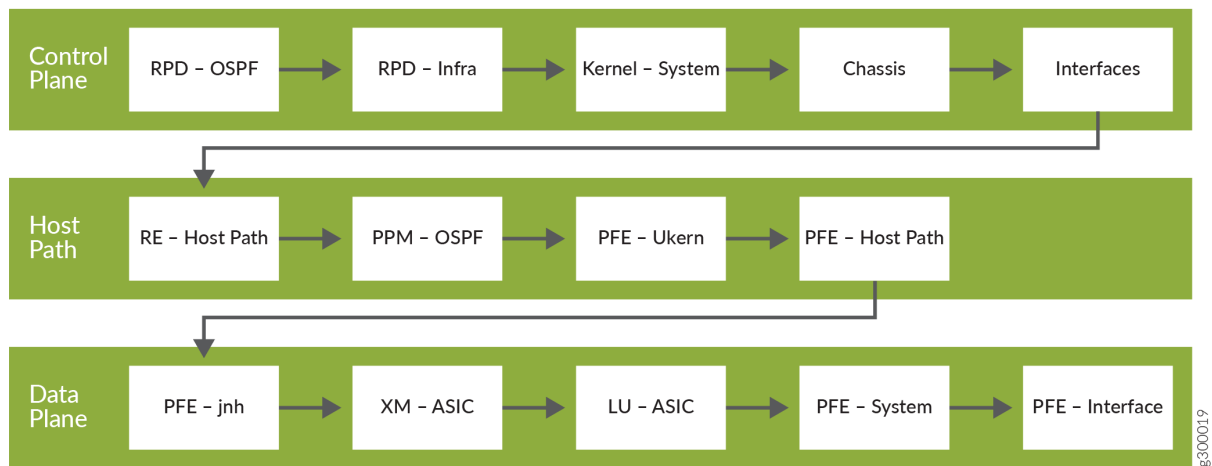
How Paragon Insights RCA Helps

The Paragon Insights RCA component simplifies the process of finding the root cause of a network issue. Paragon Insights's RCA captures the troubleshooting knowledge of specialists and has a knowledge base in the form of Paragon Insights rules. These rules are evaluated either on demand by a specific trigger or periodically in the background to ascertain the health of a networking component, such as routing protocol, system, interface, or chassis, on the device.

To illustrate the benefits of Paragon Insights RCA, let us consider the problem of OSPF flapping. [Figure 1 on page 4](#) highlights the workflow sequence involved in debugging OSPF flapping. A network

operator troubleshooting this issue would need to perform manual debugging steps for each tile (step) of the workflow sequence in order to find the root cause of the OSPF flapping. On the other hand, the Paragon Insights RCA application troubleshoots the issue automatically by using an RCA bot. The RCA bot tracks all of the telemetry data collected by the Paragon Insights and translates the information into graphical status indicators (displayed in the Paragon Insights web GUI) that correlate to different parts of the workflow sequence shown in [Figure 1 on page 4](#).

Figure 1: High-level workflow to debug OSPF-flapping



When you configure Paragon Insights, each tile of the workflow sequence (shown in [Figure 1 on page 4](#)) can be defined by one or more rules. For example, the RPD-OSPF tile could be defined as two rule conditions: one to check if "hello-transmitted" counters are incrementing and the other to check if "hello-received" counters are incrementing. Based on these user-defined rules, Paragon Insights provides status indicators, alarm notifications, and an alarm management tool to inform and alert you of specific network conditions that could lead to OSPF flapping.

By isolating a problem area in the workflow, Paragon Insights RCA proactively guides you in determining the appropriate corrective action to take to fix a pending issue or avoid a potential one.

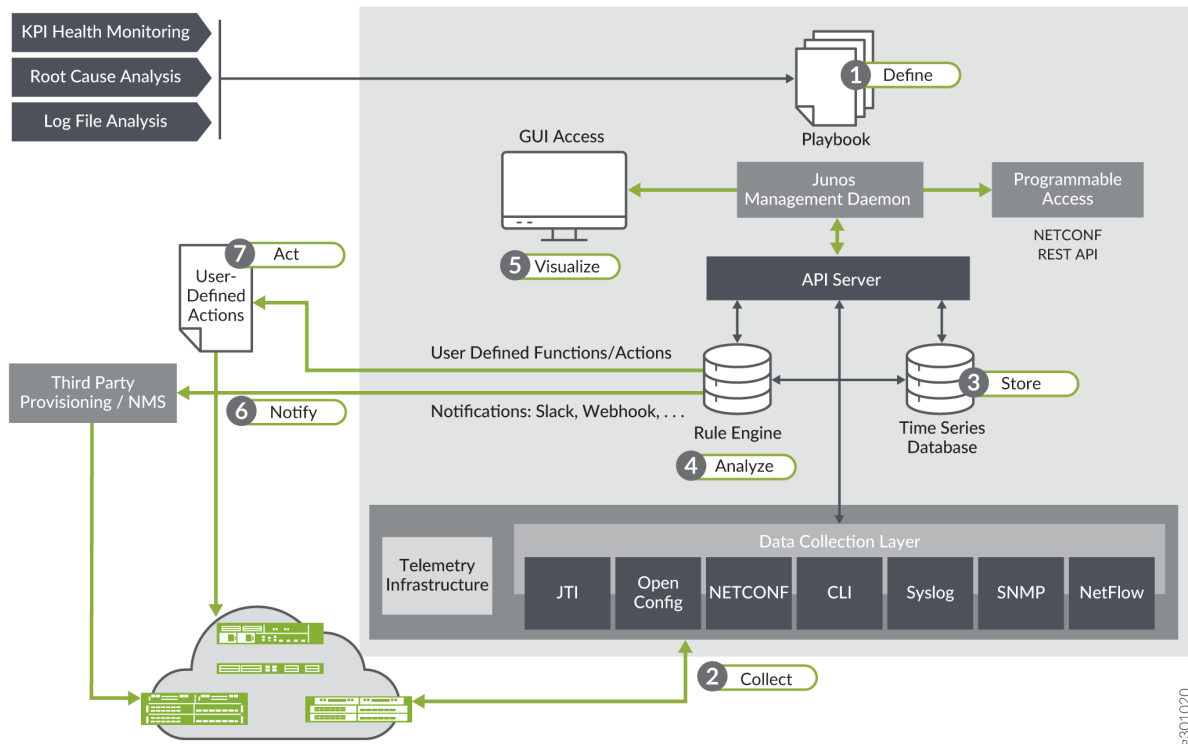
Closed-Loop Automation

Paragon Insights offers closed-loop automation. The automation workflow can be divided into seven main steps (see [Figure 2 on page 5](#)):

1. **Define**—The user defines the health parameters of key network elements through customizable key performance indicators (KPIs), rules, and playbooks, by using the tools provided by Paragon Insights.

2. **Collect**—Paragon Insights collects rule-based telemetry data from multiple devices using the collection methods specified for the different network devices.
3. **Store**—Paragon Insights stores time-sensitive telemetry data in a time-series database (TSDB). This allows users to query, perform operations on, and write new data back to the database, days, or even weeks after the initial storage.
4. **Analyze**—Paragon Insights analyzes telemetry data based on the specified KPIs, rules, and playbooks.
5. **Visualize**—Paragon Insights provides multiple ways for you to visualize the aggregated telemetry data through its web-based GUI to gain actionable and predictive insights into the health of your devices and the overall network.
6. **Notify**—Paragon Insights notifies you through the GUI and notification alarms when problems in individual devices or in the network are detected.
7. **Act**—Paragon Insights performs user-defined actions to help resolve and proactively prevent network problems.

Figure 2: Paragon Insights Closed-Loop Automation Workflow



Benefits of Paragon Insights

- Customization—Provides a framework to define and customize health profiles, allowing truly actionable insights for the specific device or network being monitored.
- Automation—Automates root cause analysis and log file analysis, streamlines diagnostic workflows, and provides self-healing and remediation capabilities.
- Greater network visibility—Provides advanced multidimensional analytics across network elements, giving you a clearer understanding of network behavior to establish operational benchmarks, improve resource planning, and minimize service downtime.
- Intuitive GUI—Offers an intuitive web-based GUI for policy management and easy data consumption.
- Open integration—Lowers the barrier of entry for telemetry and analytics by providing open source data pipelines, notification capabilities, and third-party device support.
- Multiple data collection methods—Includes support for JTI, OpenConfig, NETCONF, CLI, Syslog, NetFlow, and SNMP.

RELATED DOCUMENTATION

[Paragon Insights Getting Started Guide](#)

Paragon Insights Concepts

IN THIS SECTION

- [Paragon Insights Data Collection Methods | 7](#)
- [Paragon Insights Topics | 9](#)
- [Paragon Insights Rules - Basics | 10](#)
- [Paragon Insights Rules - Deep Dive | 12](#)
- [Paragon Insights Playbooks | 31](#)

Paragon Insights (formerly HealthBot) is a highly programmable telemetry-based analytics application. With it, you can diagnose and root cause network issues, detect network anomalies, predict potential network issues, and create real-time remedies for any issues that come up.

To accomplish this, network devices and Paragon Insights have to be configured to send and receive large amounts of data, respectively. Device configuration is covered throughout this and other sections of the guide.

Configuring Paragon Insights, or any application, to read and react to incoming telemetry data requires a language that describes several elements that are specific to the systems and data under analysis. This type of language is called a Domain Specific Language (DSL), i.e., a language that is specific to one domain. Any DSL is built to help answer questions. For Paragon Insights, these questions are:

- Q: What components make up the systems that are sending data?

A: Network devices are made up of memory, cpu, interfaces, protocols and so on. In Paragon Insights, these are called "[Paragon Insights Topics](#)" on page 9.

- Q: How do we gather, filter, process, and analyze all of this incoming telemetry data?

A: Paragon Insights uses "[Paragon Insights Rules - Basics](#)" on page 10 that consist of information blocks called sensors, fields, variables, triggers, and more.

- Q: How do we determine what to look for?

A: It depends on the problem you want to solve or the question you want to answer. Paragon Insights uses "[Paragon Insights Playbooks](#)" on page 31 to create collections of specific rules and apply them to specific groups of devices in order accomplish specific goals. For example, part of the system-kpis-playbook can alert a user when system memory usage crosses a user-defined threshold.

This section covers these key concepts and more, which you need to understand before using Paragon Insights.

Paragon Insights Data Collection Methods

IN THIS SECTION

- [Data Collection - 'Push' Model](#) | 8
- [Data Collection - 'Pull' Model](#) | 8

In order to provide visibility into the state of your network devices, Paragon Insights first needs to collect their telemetry data and other status information. It does this using sensors.

Paragon Insights supports sensors that “push” data from the device to Paragon Insights and sensors that require Paragon Insights to “pull” data from the device using periodic polling.

Data Collection - 'Push' Model

As the number of objects in the network, and the metrics they generate, have grown, gathering operational statistics for monitoring the health of a network has become an ever-increasing challenge. Traditional 'pull' data-gathering models, like SNMP and the CLI, require additional processing to periodically poll the network element, and can directly limit scaling.

The 'push' model overcomes these limits by delivering data asynchronously, which eliminates polling. With this model, the Paragon Insights server can make a single request to a network device to stream periodic updates. As a result, the 'push' model is highly scalable and can support the monitoring of thousands of objects in a network. Junos devices support this model in the form of the [Junos Telemetry Interface \(JTI\)](#).

Paragon Insights currently supports five 'push' ingest types.

- Native GPB
- NetFlow
- sFlow
- OpenConfig
- Syslog

These push-model data collection—or *ingest*—methods are explained in detail in the [Paragon Insights Data Ingest Guide](#).

Data Collection - 'Pull' Model

While the 'push' model is the preferred approach for its efficiency and scalability, there are still cases where the 'pull' data collection model is appropriate. With the 'pull' model, Paragon Insights requests data from network devices at periodic intervals.

Paragon Insights currently supports two 'pull' ingest types.

- iAgent (CLI/NETCONF)
- SNMP

These pull-model data collection—or *ingest*—methods are explained in detail in the [Paragon Insights Data Ingest Guide](#).

Paragon Insights Topics

Network devices are made up of a number of components and systems from CPUs and memory to interfaces and protocol stacks and more. In Paragon Insights, a topic is the construct used to address those different device components. The **Topic** block is used to create name spaces that define what needs to be modeled. Each **Topic** block is made up of one or more **Rule** blocks which, in turn, consist of the **Field** blocks, **Function** blocks, **Trigger** blocks, etc. See "[Paragon Insights Rules - Deep Dive](#)" on page 12 for details. Each rule created in Paragon Insights must be part of a topic. Juniper has curated a number of these system components into a list of **Topics** such as:

- chassis
- class-of-service
- external
- firewall
- interfaces
- kernel
- linecard
- logical-systems
- protocol
- routing-options
- security
- service
- system

You can create sub-topics underneath any of the Juniper topic names by appending `.<sub-topic>` to the topic name. For example, `kernel.tcpip` or `system.cpu`.

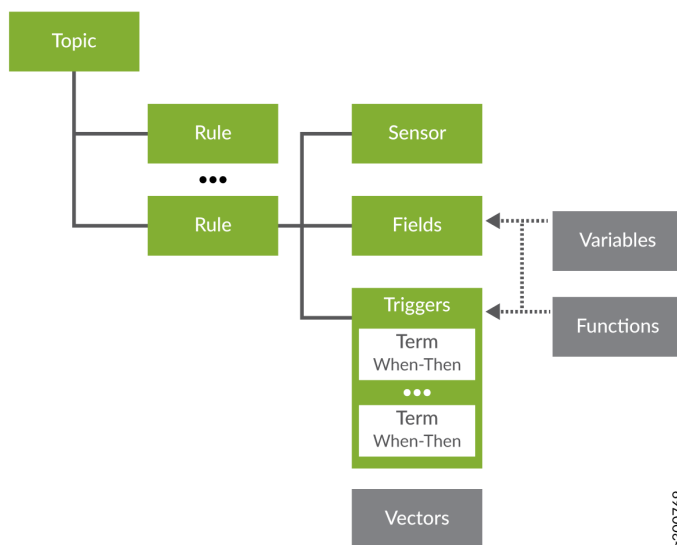
Any pre-defined rules provided by Juniper fit within one of the Juniper topics with the exception of *external*. The *external* topic is reserved for user-created rules. In the Paragon Insights web GUI, when you create a new rule, the **Topics** field is automatically populated with the *external* topic name.

Paragon Insights Rules - Basics

Paragon Insights' primary function is collecting and reacting to telemetry data from network devices. Defining how to collect the data, and how to react to it, is the role of a *rule*.

Paragon Insights ships with a set of default rules, which can be seen on the **Configuration > Rules** page of the Paragon Insights GUI, as well as in GitHub in the [healthbot-rules](#) repository. You can also create your own rules.

The structure of a Paragon Insights rule looks like this:



To keep rules organized, Paragon Insights organizes them into *topics*. Topics can be very general, like **system**, or they can be more granular, like **protocol.bgp**. Each topic contains one or more rules.

As described above, a *rule* contains all the details and instructions to define how to collect and handle the data. Each rule contains the following required elements:

- The *sensor* defines the parameters for collecting the data. This typically includes which data collection method to use (as discussed above in "[Paragon Insights Data Collection Methods](#)" on page 7), some guidance on which data to ingest, and how often to push or pull the data. In any given rule, a sensor can be defined directly within the rule or it can be referenced from another rule.
- Example: Using the SNMP sensor, poll the network device every 60 seconds to collect all the device data in the Juniper SNMP MIB table [jnxOperatingTable](#).
- The sensor typically ingests a large set of data, so *fields* provide a way to filter or manipulate that data, allowing you to identify and isolate the specific pieces of information you care about. Fields can also act as placeholder values, like a static threshold value, to help the system perform data analysis.

- Example: Extract, isolate, and store the **jnxOperating15MinLoadAvg** (CPU 15-minute average utilization) value from the SNMP table specified above in the sensor.
- *Triggers* periodically bring together the fields with other elements to compare data and determine current device status. A trigger includes one or more 'when-then' statements, which include the parameters that define how device status is visualized on the health pages.
 - Example: Every 90 seconds, check the CPU 15min average utilization value, and if it goes above a defined threshold, set the device's status to red on the device health page and display a message showing the current value.

The rule can also contain the following optional elements:

- *Vectors* allow you to leverage existing elements to avoid the need to repeatedly configure the same elements across multiple rules.
 - Examples: A rule with a configured sensor, plus a vector to a second sensor from another rule; a rule with no sensors, and vectors to fields from other rules
- *Variables* can be used to provide additional supporting parameters needed by the required elements above.
 - Examples: The string "ge-0/0/0", used within a field collecting status for all interfaces, to filter the data down to just the one interface; an integer, such as "80", referenced in a field to use as a static threshold value
- *Functions* allow you to provide instructions (in the form of a Python script) on how to further interact with data, and how to react to certain events.
 - Examples: A rule that monitors input and output packet counts, using a function to compare the count values; a rule that monitors system storage, invoking a function to cleanup temp and log files if storage utilization goes above a defined threshold

NOTE: Rules, on their own, don't actually do anything. To make use of rules you need to add them to ["Paragon Insights Playbooks" on page 31](#).

Paragon Insights Rules - Deep Dive

IN THIS SECTION

- [Rules | 13](#)
- [Sensors | 16](#)
- [Fields | 16](#)
- [Vectors | 18](#)
- [Variables | 19](#)
- [Functions | 19](#)
- [Triggers | 20](#)
- [Tagging | 24](#)
- [Rule Properties | 24](#)
- [Pre/Post Action | 24](#)
- [Multiple Sensors per Device | 25](#)
- [Sensor Precedence | 28](#)

A rule is a package of components, or blocks, needed to extract specific information from the network or from a Junos device. **Rules** conform to a specifically tailored domain specific language (DSL) for analytics applications. The DSL is designed to allow rules to capture:

- The minimum set of input data that the rule needs to be able to operate
- The minimum set of telemetry sensors that need to be configured on the device(s)
- The fields of interest from the configured sensors
- The reporting or polling frequency
- The set of triggers that operate on the collected data
- The conditions or evaluations needed for triggers to kick in
- The actions or notifications that need to be performed when a trigger kicks in

The details around rules, topics and playbooks are presented in the following sections.

Rules

Rules are meant to be free of any hard coding. Think of threshold values; If a threshold is hard coded, there is no easy way to customize it for a different customer or device that has different requirements. Therefore, rules are defined using parameterization to set the default values. This allows the parameters to be left at default or be customized by the operator at the time of deployment. Customization can be done at the device group or individual device level while applying the ["Paragon Insights Playbooks" on page 31](#) in which the individual rules are contained.

Rules that are device-centric are called device rules. Device components such as chassis, system, linecards, and interfaces are all addressed as ["Paragon Insights Topics" on page 9](#) in the rule definition. Generally, device rules make use of sensors on the devices.

Rules that span multiple devices are called network rules. Network rules:

- must have a rule-frequency configured
- must not contain sensors
- cannot be mixed with device rules in a playbook

To deploy either type of rule, include the rule in a playbook and then apply the playbook to a device group or network group.

NOTE: Paragon Insights comes with a set of pre-defined rules.

Not all of the blocks that make up a rule are required for every rule. Whether or not a specific block is required in a rule definition depends on what sort of information you are trying to get to. Additionally, some rule components are not valid for network rules. [Table 1 on page 14](#) lists the components of a rule and provides a brief description of each one.

Table 1: Rule Components

Block	What it Does	Required in Device Rules?	Valid for Network Rules?
"Sensors" on page 16	<p>The Sensors block is like the access method for getting at the data. There are multiple types of sensors available in Paragon Insights: OpenConfig, Native GPB, iAgent, SNMP, and syslog.</p> <p>It defines what sensors need to be active on the device in order to get to the data fields on which the triggers eventually operate. Sensor names are referenced by the Fields.</p> <p>OpenConfig and iAgent sensors require that a frequency be set for push interval or polling interval respectively. SNMP sensors also require you to set a frequency.</p>	No—Rules can be created that only use a field reference from another rule or a vector with references from another rule. In these cases, rule-frequency must be explicitly defined.	No
"Fields" on page 16	The source for the Fields block can be a pointer to a sensor, a reference to a field defined in another rule, a constant, or a formula. The field can be a string, integer or floating point. The default field type is string.	Yes— Fields contain the data on which the triggers operate. Starting in HealthBot Release 3.1.0, regular fields and key-fields can be added to rules based on conditional tagging profiles. See the "Tagging" on page 24 section below.	Yes
"Vectors" on page 18	The Vectors block allows handling of lists, creating sets, and comparing elements amongst different sets. A vector is used to hold multiple values from one or more fields.	No	Yes

Table 1: Rule Components *(Continued)*

Block	What it Does	Required in Device Rules?	Valid for Network Rules?
"Variables" on page 19	The Variables block allows you to pass values into rules. Invariant rule definitions are achieved through mustache-style templating like {{<placeholder-variable> }}. The placeholder-variable value is set in the rule by default or can be user-defined at deployment time.	No	No
"Functions" on page 19	The Functions block allows you to extend fields, triggers, and actions by creating prototype methods in external files written in languages like python. The functions block includes details on the file path, method to be accessed, and any arguments, including argument description and whether it is mandatory.	No	No
"Triggers" on page 20	<p>The Triggers block operates on fields and are defined by one or more <i>Terms</i>. When the conditions of a Term are met, then the action defined in the Term is taken.</p> <p>By default, triggers are evaluated every 10 seconds, unless explicitly configured for a different frequency.</p> <p>By default, all triggers defined in a rule are evaluated in parallel.</p>	Yes—Triggers enable rules to take action.	Yes
"Rule Properties" on page 24	The Rule Properties block allows you to specify metadata for a Paragon Insights rule, such as hardware dependencies, software dependencies, and version history.	No	Yes
"Pre/Post Action" on page 24	The Pre/Post Action block allows you to automate Action Engine Workflows when you add them in rule configurations. Paragon Insights executes the pre-action tasks at the start of playbook instantiation and the post-action tasks after you stop playbook instances.	No	Yes

Sensors

When defining a sensor, you must specify information such as sensor name, sensor type and data collection frequency. As mentioned in [Table 1 on page 14](#), sensors can be one of the following:

- **OpenConfig** For information on OpenConfig JTI sensors, see the [Junos Telemetry Interface User Guide](#).
- **Native GPB** For information on Native GPB JTI sensors, see the [Junos Telemetry Interface User Guide](#).
- **iAgent** The iAgent sensors use NETCONF and YAML-based PyEZ tables and views to fetch the necessary data. Both structured (XML) and unstructured (VTY commands and CLI output) data are supported. For information on Junos PyEZ, see the [Junos PyEZ Documentation](#).
- **SNMP** Simple Network Management Protocol.
- **syslog** system log
- **BYOI** Bring your own ingest – Allows you to define your own ingest types.
- **Flow** NetFlow traffic flow analysis protocol
- **sFlow** sFlow packet sampling protocol

When different rules have the same sensor defined, only one subscription is made per sensor. A key, consisting of *sensor-path* for OpenConfig and Native GPB sensors, and the tuple of *file* and *table* for iAgent sensors is used to identify the associated rule.

When multiple sensors with the same *sensor-path* key have different frequencies defined, the lowest frequency is chosen for the sensor subscription.

Fields

There are four types of field sources, as listed in [Table 1 on page 14](#). [Table 2 on page 17](#) describes the four field ingest types in more detail.

Table 2: Field Ingest Type Details

Field Type	Details
Sensor	<p>Subscribing to a sensor typically provides access to multiple columns of data. For instance, subscribing to the OpenConfig interface sensor provides access to a bunch of information including counter related information such as:</p> <p>/interfaces/counters/tx-bytes,</p> <p>/interfaces/counters/rx-bytes,</p> <p>/interfaces/counters/tx-packets,</p> <p>/interfaces/counters/rx-packets,</p> <p>/interfaces/counters/oper-state, etc.</p> <p>Given the rather long names of paths in OpenConfig sensors, the Sensor definition within Fields allows for aliasing, and filtering. For single-sensor rules, the required set of Sensors for the Fields table are programmatically auto-imported from the raw table based on the triggers defined in the rule.</p>
Reference	<p>Triggers can only operate on Fields defined within that rule. In some cases, a Field might need to reference another Field or Trigger output defined in another Rule. This is achieved by referencing the other field or trigger and applying additional filters. The referenced field or trigger is treated as a stream notification to the referencing field. References aren't supported within the same rule.</p> <p>References can also take a time-range option which picks the value, if available, from the time-range provided. Field references must always be unambiguous, so proper attention must be given to filtering the result to get just one value. If a reference receives multiple data points, or values, only the latest one is used. For example, if you are referencing a the values contained in a field over the last 3 minutes, you might end up with 6 values in that field over that time-range. Paragon Insights only uses the latest value in a situation like this.</p>
Constant	<p>A field defined as a constant is a fixed value which cannot be altered during the course of execution. Paragon Insights Constant types can be strings, integers, and doubles.</p>

Table 2: Field Ingest Type Details (*Continued*)

Field Type	Details
Formula	<p>Raw sensor fields are the starting point for defining triggers. However, Triggers often work on derived fields defined through formulas by applying mathematical transformations.</p> <p>Formulas can be pre-defined or user-defined (UDF). Pre-defined formulas include: Min, Max, Mean, Sum, Count, Rate of Change, Elapsed Time, Eval, Standard Deviation, Microburst, Dynamic Threshold, Anomaly Detection, Outlier Detection, and Predict.</p> <p>Rate of Change refers to the difference between current and previous values over their points of time. Packet transfer is an example use case where the Rate of Change formula can be used. The Hold Time field takes a threshold of time interval. The time interval between current and previous values cannot exceed the specified Hold Time value. The Multiplication Factor field is used to convert the unit of the field value. If the field value is calculated in Bytes, specifying 1024 as Multiplication Factor would convert the result into Kilobytes. Hold Time and Multiplication Factor are not mandatory fields when you apply Rate of Change formula.</p> <p>In Paragon Insights 4.0.0, you can get the current point time in Elapsed Time formula by using \$time.</p> <p>Some pre-defined formulas can operate on time ranges in order to work with historical data. If a time range is not specified, then the formula works on current data, specified as <i>now</i>.</p>

Vectors

Vectors are useful in helping to gather multiple elements into a single rule. For example, using a vector you could gather all of the interface error fields. The syntax for Vector is:

```
vector <vector-name>{
    path [$field-1 $field-2 .. $field-n];
    filter <list of specific element(s) to filter out from vector>;
    append <list of specific element(s) to be added to vector>;
}
```

\$field-n can be field of type reference.

The fields used in defining vectors can be direct references to fields defined in other rules:

```
vector <vector-name>{
    path [/device-group[device-group-name=<device-group>]\
```

```

/device[device-name=<device>]/topic[topic-name=<topic>]\
/rule[rule-name=<rule>]/field[<field-name>=<field-value>\
AND|OR ...]/<field-name> ...];
    filter <list of specific element(s) to filter out from vector>;
    append <list of specific element(s) to be added to vector>;
}

```

This syntax allows for optional filtering through the <field-name>=<field-value> portion of the construct. Vectors can also take a time-range option that picks the values from the time-range provided. When multiple values are returned over the given time-range, they are all selected as an array.

The following pre-defined formulas are supported on vectors:

- `unique @vector1`—Returns the unique set of elements from vector1
- `@vector1 and @vector2`—Returns the intersection of unique elements in vector1 and vector2.
- `@vector1 or @vector2`—Returns the total set of unique elements in the two vectors.
- `@vector1 unless @vector2`—Returns the unique set of elements in vector-1, but not in vector-2

Variables

Variables are defined during rule creation on the **Variables** page. This part of variable definition creates the default value that gets used if no specific value is set in the device group or on the device during deployment. For example, the `check-interface-status` rule has one variable called `interface_name`. The value set on the **Variables** page is a regular expression (regex), `.*`, that means all interfaces.

If applied as-is, the `check-interface-status` rule would provide interface status information about all the interfaces on all of the devices in the device group. While applying a playbook that contains this rule, you could override the default value at the device group or device level. This allows you flexibility when applying rules. The order of precedence is device value overrides device group value and device group value overrides the default value set in the rule.

BEST PRACTICE: It is highly recommended to supply default values for variables defined in device rules. All Juniper-supplied rules follow this recommendation. Default values must not be set for variables defined in network rules.

Functions

Functions are defined during rule creation on the **Functions** tab. Defining a function here allows it to be used in **Formulas** associated with **Fields** and in the **When** and **Then** sections of **Triggers**. Functions used

in the *when* clause of a trigger are known as user-defined functions. These must return true or false. Functions used in the *then* clause of a trigger are known as user-defined actions.

Triggers

Triggers play a pivotal role in Paragon Insights rule definitions. They are the part of the rule that determines if and when any action is taken based on changes in available sensor data. Triggers are constructed in a *when-this, then-that* manner. As mentioned earlier, trigger actions are based on **Terms**. A **Term** is built with *when* clauses that watch for updates in field values and *then* clauses that initiate some action based on what changed. Multiple **Terms** can be created within a single trigger.

Evaluation of the *when* clauses in the **Terms** starts at the top of the list of terms and proceeds to the bottom. If a *term* is evaluated and no match is made, then the next *term* is evaluated. By default, evaluation proceeds in this manner until either a match is made or the bottom of the list is reached without a match.

Pre-defined operators that can be used in the *when* clause include:

NOTE: For evaluated equations, the left-hand side and right-hand side of the equation are shortened to LHS and RHS, respectively in this document.

- *greater-than*—Used for checking if one value is greater than another.
 - Returns: True or False
 - Syntax: `greater-than <LHS> <RHS> [time-range <range>]`
 - Example: `//Memory > 3000 MB in the last 5 minutes`
`when greater-than $memory 3000 time-range 5m;`
- *greater-than-or-equal-to*—Same as *greater-than* but checks for greater than or equal to (\geq)
- *less-than*
 - Returns: True or False
 - Syntax: `less-than <LHS> <RHS> [time-range <range>]`
 - Example: `//Memory < 6000 MB in the last 5 minutes`
`when less-than $memory 6000 time-range 5m;`
- *less-than-or-equal-to*—Same as *less-than* but checks for less than or equal to (\leq)
- *equal-to*—Used for checking that one value is equal to another value.

- Returns: True or False
- Syntax: `equal-to <LHS> <RHS> [time-range <range>]`
- Example: `//Queue's buffer utilization % == 0`
`when equal-to $buffer-utilization 0;`
- *not-equal-to*—Same as *equal-to* but checks for negative condition (`!=`)
- *exists*—Used to check if some value exists without caring about the value itself. Meaning that some value should have been sent from the device.
 - Returns: True or False
 - Syntax: `exists <$var> [time-range <range>]`
 - Example: `//Has the device configuration changed?`
`when exists $netconf-data-change`
- *matches-with (for strings & regex)*—Used to check for matches on strings using Python regex operations. See [Syntax](#) for more information.

NOTE: LHS, or left hand side, is the string in which we are searching; RHS, or right hand side, is the match expression. Regular expressions can only be used in RHS.

- Returns: True or False
- Syntax: `matches-with <LHS> <RHS> [time-range <range>]`
- Example: `//Checks that ospf-neighbor-state has been UP for the past 10 minutes`
`when matches-with $ospf-neighbor-state “^UP$” time-range 10m;`

Example 1: escape backslash `'\'` with one more backslash `'\'`

escape `\` with one more `\`

Expression: `^\\S+-\\d+\\/\\d+\\/\\d+$`

Values that will match:

`xe-1/0/0`

`et-1/0/1`

`fe-2/0/0`

Values that will not match:

```
xe-1/0/0.1
fxp0
```

Example 2: escape \ with one more \

Expression: `\\(\\S+\\)\\s\\(\\S+\\)`

Values that will match:

(root) (mgd)

(user1) (ssh)

Values that will not match:

root mgd

(root) mgd

- *does-not-match-with (for strings & regex)*– Same as *matches-with* but checks for negative condition
- *range*– Checks whether a value, X, falls within a given range such as minimum and maximum (min <= X <= max)
 - Returns: True or False
 - Syntax: `range <$var> min <minimum value> max <maximum value> [time-range <range>]`
 - Example: //Checks whether memory usage has been between 3000 MB and 6000 MB in the last 5 minutes


```
when range $mem min 3000 max 6000 time-range 5m;
```
- *increasing-at-least-by-value*– Used to check whether values are increasing by at least the minimum acceptable rate compared to the previous value. An optional parameter that defines the minimum acceptable rate of increase can be provided. The minimum acceptable rate of increase defaults to 1 if not specified.
 - Returns: True or False
 - Syntax:


```
increasing-at-least-by-value <$var> [increment <minimum value of increase between successive points>]
```

```
increasing-at-least-by-value <$var> [increment <minimum value of increase between successive points>] time-range <range>
```
 - Example: Checks that the ospf-tx-hello has been increasing steadily over the past 5 minutes.


```
when increasing-at-least-by-value $ospf-tx-hello increment 10 time-range 5m;
```

- *increasing-at-most-by-value*—Used to check whether values are increasing by no more than the maximum acceptable rate compared to the previous value. An optional parameter that defines the maximum acceptable rate of increase can be provided. The maximum acceptable rate of increase defaults to 1 if not specified.

- Returns: True or False

- Syntax:

increasing-at-most-by-value <\$var> [increment <maximum value of increase between successive points>]

increasing-at-most-by-value <\$var> [increment <maximum value of increase between successive points>] time-range <range>

- Example: Checks that the error rate has not increased by more than 5 in the past 5 minutes.

```
when increasing-at-most-by-value $error-count increment 5 time-range 5m;
```

- *increasing-at-least-by-rate*—Used for checking that rate of increase between successive values is at least given rate. Mandatory parameters include the value and time-unit, which together signify the minimum acceptable rate of increase.

- Returns: True or False

- Syntax:

This syntax compares current value against previous value ensuring that it increases at least by value rate.

increasing-at-least-by-rate <\$var> value <minimum value of increase between successive points> per <second|minute|hour|day|week|month|year> [time-range <range>]

This syntax compares current value against previous value ensuring that it increases at least by percentage rate

increasing-at-least-by-rate <\$var> percentage <percentage> per <second|minute|hour|day|week|month|year> [time-range <range>]

- Example: Checks that the ospf-tx-hello has been increasing strictly over the past five minutes.

```
when increasing-at-least-by-rate $ospf-tx-hello value 1 per second time-range 5m;
```

- *increasing-at-most-by-rate*—Similar to increasing-at-least-by-rate, except that this checks for decreasing rates.

Using these operators in the *when* clause, creates a function known as a user-defined condition. These functions should always return true or false.

If evaluation of a *term* results in a match, then the action specified in the *Then* clause is taken. By default, processing of terms stops at this point. You can alter this flow by enabling the **Evaluate next term** button at the bottom of the *Then* clause. This causes Paragon Insights to continue *term* processing to create more complex decision-making capabilities like when-this and this, then that.

The following is a list of pre-defined actions available for use in the *Then* section:

- next
- status

Tagging

Starting with Release 3.1.0, HealthBot supports tagging. Tagging allows you to insert fields, values, and keys into a Paragon Insights rule when certain conditions are met. See ["Paragon Insights Tagging" on page 33](#) for details.

Rule Properties

The **Rule Properties** block allows you to specify metadata for a Paragon Insights rule, such as hardware dependencies, software dependencies, and version history. This data can be used for informational purposes or to verify whether or not a device is compatible with a Paragon Insights rule.

Pre/Post Action

(Optional) Starting with Paragon Insights Release 4.3.0, you can use pre-action and post-action tab in rules to automate tasks that are pre-configured in action engine workflows. Action engine workflows are used to perform tasks that you can execute as command line commands, NETCONF commands, or as commands in executable files. See ["Understanding Action Engine Workflows" on page 244](#) for more information.

When you run playbook instances on device groups, Paragon Insights executes pre-action tasks at the start of playbook instantiation. Pre-action tasks are useful to execute device configurations in a device group or to issue a notification about device status to another application. There is no dependency between multiple pre-action tasks and between the execution of pre-action tasks and rules. If you configure more than one pre-action task in a rule, Paragon Insights executes all pre-action tasks simultaneously.

Paragon Insights executes post-action tasks when you stop a playbook instance. Even though an optional configuration, post-action tasks are useful to remove any additional configurations added to devices through the pre-action tasks.

Both pre- and post-action tasks have the execute-once option. By default, execute-once is disabled. If you enable execute-once, then Paragon Insights executes the tasks only once on a device in a device group. Execute-once is applicable in the following cases:

- When you run multiple instances of a playbook on the same device group.
- When you include a rule with a set of pre-action or post-action tasks in different playbooks, and run the playbook instances on the same device group.

Paragon Insights checks and resolves duplication of pre-action and post-action tasks before executing them on the devices in a device group. Duplication occurs when you configure a specific pre-action or post-action task in many rules that are included in a playbook.

NOTE: When you upgrade Paragon Insights, the application does not execute pre-action and post-action tasks that are deployed before the upgrade.

To configure pre-action tasks and post-action tasks in rules, see ["Paragon Insights Rules and Playbooks" on page 141](#).

Multiple Sensors per Device

Paragon Insights Release 4.0.0 allows you to add multiple sensors per rule that can be applied to all the devices in a device group. In earlier releases, you could add only one sensor per rule. Each sensor per rule generates data in a field table. If you added the different sensors in multiple rules, it results in as many field tables as the number of rules. When you add multiple types of sensors (such as OpenConfig or Native GPB) in a single rule in Paragon Insights, data from the sensors is consolidated in a single field table that is simpler to export or to visualize. The GUI support for multiple sensor configurations will be implemented in subsequent releases.

Guidelines

Sp-admins or users with create access privilege must note the following guidelines when configuring multiple sensors.

- When adding multiple sensors to rules, you must ensure that there are no overlapping data or keysets received from the sensors applied to a device. Overlapping keysets can result in duplicate data points, overwriting of data points, and inaccurate evaluation of data. To avoid this, you can use filter expressions such as the `where` statement in **Fields**.
- When you add multiple sensors in a rule in Paragon Insights GUI, you must set a common value for all sensors in the following fields:

- *Frequency* field in **Sensors** tab (sensor frequency)
- *Field aggregation time-range* field in Rules page.
- *Frequency* field in **Triggers** tab.
- *Time range* field used in triggers, formula, and reference.

For example, when multiple sensors are added in a rule, all sensors applied to a device must have the same value for sensor frequency, irrespective of the type of sensors. If a rule has iAgent and OpenConfig sensors, the *Frequency* value in both sensors must be the same. This holds true for all the fields listed above.

NOTE: We recommend you use offset values if you cannot match the time range values on different sensors. For more information, see [Frequency Profiles and Offset Time](#).

However, the frequency you set in a frequency profile will override the frequency values set in multiple sensors in a rule.

- A Rule with multiple sensors is applied on all devices added in a particular device group. It is assumed that the devices in a device group support the types of sensors used in Paragon Insights rules.

Sometimes, not all devices in a device group can support the same type of sensor. For example, the device group DG1 has an MX2020 router with OpenConfig package installed and another MX2020 router configured without the OpenConfig package. The first MX2020 router would support OpenConfig sensor whereas, the second MX2020 router would not support the same sensor.

To avoid such scenarios, you must ensure that all devices in the same device group unanimously support the types of sensors used to collect information.

Configure Multiple Sensors in Paragon Insights GUI

To add multiple sensors in a rule:

1. Navigate to **Configuration > Rules > Add Rule**.

Rules page appears.

2. Enter the Paragon Insights topic and rule name, rule description, synopsis, and optionally, the field aggregation time-range and rule frequency. For more information on these fields in Rules page, see ["Paragon Insights Rules and Playbooks" on page 141](#).
3. Click **Add Sensor** button in **Sensors** tab and fill in the necessary details based on the type of sensor you choose.

Repeat step 3 to add as many sensors as you need for your use case.

4. Configure fields for the sensors in the **Fields** tab.
5. Configure sensors in Rule Properties tab to set multiple sensors active.

You must enter all the sensors, earlier configured in the rule, under the supported-devices hierarchy in Rule Properties. For example, if you configured sensors s1, s2, and s3 in a rule, the Rule Properties configuration must also include the same sensors:

```
rule-properties {
    version 1;
    contributor juniper;
    supported-healthbot-version 4.0.0;
    supported-devices {
        sensors [s1 s2 s3];
    }
}
```

You can also write and upload a new rule in the Paragon Insights GUI.

The rule must follow the curly brackets format ({ }) and indentation for hierarchical structure. Terminating or leaf statements in the configuration hierarchy are displayed with a trailing semicolon (;) to define configuration details, such as supported version, sensors, and other configuration statements.

6. Click **Save & Deploy** to apply the new sensors in your network or click **Save** to save the configurations of the new sensors and deploy them later.

Use Cases

The following scenarios illustrate use cases for multiple sensors in a rule:

- In Pathfinder Controller, there can be different native sensors that provide non-overlapping counter details for Segment Routing (SR) and Resource Reservation Protocol (RSVP) label switched paths (LSP). If the field table needs to be combined for the data collected from the LSPs, multiple sensors can be made active for a device in the same rule.
- If you want to get data for *ge* interface using iAgent sensor and for *fe* interface using Native GPB sensor, then you could use multiple active sensor for a device. You need to ensure non-overlapping data in this case by using Field filtering expression to filter by the interface name. Instead of interfaces, an sp-admin or a user with create access privilege can consider any other key performance indicators too.

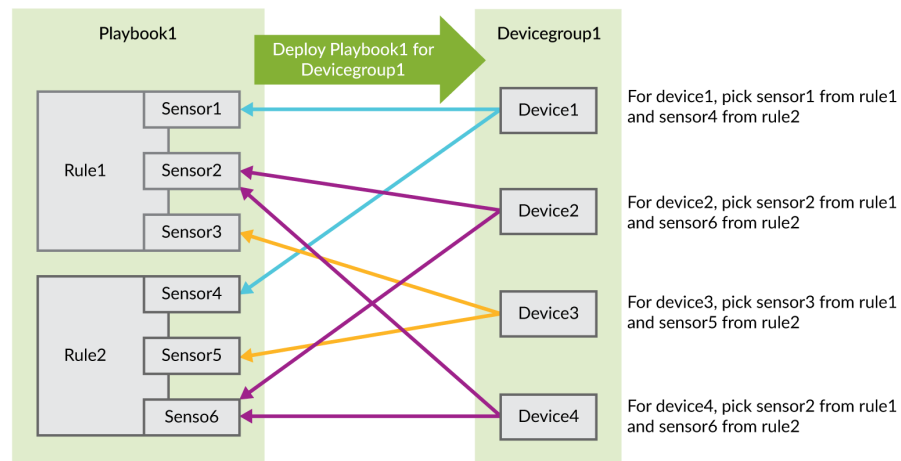
Sensor Precedence

To make data collection from a sensor effective, devices in a device group must support a particular sensor as an ingest method. Select devices in a device group running an older version of operating system, devices from different vendors in a device group, or different products from the same vendor (such as EX, MX and PTX routers from Juniper) are all scenarios that can cause challenges to applying a sensor to a device group. In such cases, you need to set a different sensor that is compatible with specific devices in a device group.

Paragon Insights Release 4.0.0 enables you to set sensor precedence so that, you can configure different sensors in each hierarchy of Rule Properties such as vendor name, operating system, product name, platform, and release version. This makes it possible to apply suitable sensors on multi-vendor devices in a heterogeneous device group. In Release 4.0.0, you can configure sensor precedence only through Paragon Insights CLI. Starting in Paragon Insights Release 4.2.0, you can configure more than one sensor for a rule. For more information, see ["Paragon Insights Rules and Playbooks" on page 141](#).

[Figure 3 on page 28](#) illustrates two rules each with multiple sensors. It is assumed that Rule Properties is configured for sensor precedence.

Figure 3: Representation of Sensor Precedence in Rules



Let us suppose Sensor1 in Rule 1 is OpenConfig and Sensor4 in Rule 2 is iAgent and Device1 runs Junos operating system (OS). If OpenConfig and iAgent were set as default sensors for Junos OS hierarchy in Rule Properties then, Device1 would receive data from Sensor1 and Sensor4 when the Playbook is deployed for the device group.

Before You Begin

In standalone deployment of Paragon Insights, before you configure sensor precedence, the devices in a device group must be configured with the following fields:

- *Vendor name*: Paragon Insights Release 4.0.0 supports multiple vendors that include Juniper Networks, Cisco Systems, Arista Networks, and PaloAlto Networks.
- *Operating system*: Name of the operating system supported by vendors such as Junos, IOS XR, and so on.
- *Product*: Name of the family of products (devices) offered by a vendor. For example, MX routers, ACX routers, PTX routers from Juniper Networks.
- *Platform*: Particular member device in a series of products. For example, MX2020, ACX5400 and so on.
- *Release or version*: release version of the platform selected.

The configuration of above fields in the device hierarchy must match the sensor precedence specified in the Rule Properties. For example, if you include platform MX2020 in device configuration, the sensor precedence hierarchy must also include MX2020.

Configuration for Sensor Precedence

The following is a sample configuration of setting sensor precedence in Rule Properties.

```
rule-properties {
    version 1;
    contributor juniper;
    supported-healthbot-version 4.0.0;
    supported-devices {
        sensors interfaces-iagent;
        juniper {
            sensors interfaces-iagent;
            operating-system junos {
                products EX {
                    sensors interfaces-iagent;
                    platforms EX9200 {
                        sensors interfaces-iagent;
                        releases 17.3R1 {
                            sensors interfaces-iagent;
                            release-support min-supported-release;
                        }
                    }
                }
            }
        }
    }
}
```

```

    }
    platforms EX9100 {
        sensors interfaces-oc;
    }
}
products MX {
    sensors interfaces-oc;
}
products PTX {
    sensors interfaces-oc;
}
products QFX {
    sensors interfaces-oc;
}
}
}
other-vendor cisco {
    vendor-name cisco;
    sensors interfaces-oc;
}
}
}

```

Sensor precedence mandates changes to the current hierarchy of configuration in Rule Properties. The hierarchy of the following elements have changed in Paragon Insights Release 4.0.0. The old hierarchy of the listed elements in Rule Properties is deprecated.

- *Releases*: The old statement hierarchy defined Releases under Product, and Platform was listed under Releases. This hierarchy is deprecated.

```

products MX {
    releases 15.2R1 {
        release-support min-supported-release;
        platform All;
    }
}

```

Deprecated
Deprecated
Deprecated

- Operating system: The old statement hierarchy defined operating system as a leaf element for other vendors. This order is deprecated.

```
other-vendor cisco {
    vendor-name cisco;
    operating-system nexus;    ### Deprecated
    sensors [ s1 s2 ]; // Default sensors for cisco vendor
    operating-systems nxos {
        sensors [ s1 s2 ]; // Default sensors for cisco vendor with NX OS
        products NEXUS {
            sensors [ s1 s2 ]; // Default sensors for cisco vendor with NX OS and for
the specified product
            platforms 7000 {
                sensors [ s1 s2 ]; // Default sensors for cisco vendor with NX OS and
for the specified product and platform
                releases 15.8 {
                    release-support only-on-this-release;
                    sensors [ s1 s2 ]; // Sensors for cisco vendor with NX OS and for
the product, platform and version
                }
            }
        }
    }
}
```

Paragon Insights Playbooks

In order to fully understand any given problem or situation on a network, it is often necessary to look at a number of different system components, topics, or key performance indicators (KPIs). Paragon Insights operates on playbooks, which are collections of rules for addressing a specific use case. **Playbooks** are the Paragon Insights element that gets applied, or run, on your device groups or network groups.

Paragon Insights comes with a set of pre-defined **Playbooks**. For example, the system-KPI playbook monitors the health of system parameters such as system-cpu-load-average, storage, system-memory, process-memory, etc. It then notifies the operator or takes corrective action in case any of the KPIs cross pre-set thresholds. Following is a list of Juniper-supplied Playbooks.

- bgp-session-stats
- route-summary-playbook
- lldp-playbook

- interface-kpis-playbook
- system-kpis-playbook
- linecard-kpis-playbook
- chassis-kpis-playbook

You can create a playbook and include any rules want in it. You apply these playbooks to device groups. By default, all rules contained in a Playbook are applied to all of the devices in the device group. There is currently no way to change this behavior.

If your playbook definition includes network rules, then the playbook becomes a network playbook and can only be applied to network groups.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.2.0	Starting in Paragon Insights Release 4.2.0, you can configure more than one sensor for a rule.
4.0.0	Paragon Insights Release 4.0.0 allows you to add multiple sensors per rule that can be applied to all the devices in a device group.
4.0.0	Paragon Insights Release 4.0.0 enables you to set sensor precedence so that, you can configure different sensors in each hierarchy of Rule Properties such as vendor name, operating system, product name, platform, and release version.
3.1.0	Starting in HealthBot Release 3.1.0, regular fields and key-fields can be added to rules based on conditional tagging profiles.
3.1.0	Starting with Release 3.1.0, HealthBot supports tagging.

RELATED DOCUMENTATION

| [Paragon Insights Rules and Playbooks](#) | 141

Paragon Insights Tagging

IN THIS SECTION

- [Overview | 33](#)
- [Types of Tagging | 40](#)
- [Add a Tagging Profile | 48](#)
- [Apply a Tagging Profile | 53](#)
- [Delete a Tagging Profile | 55](#)

You can use the Paragon Insights (formerly HealthBot) graphical user interface (GUI) to create tagging profiles. You can configure a tagging profile to insert fields, values, and keys into a Paragon Insights rule. You can also set conditions that are checked against values stored in the times series database (TSDB) or Redis database.

Overview

IN THIS SECTION

- [Tagging Profile Terminology | 34](#)
- [How do Tagging Profiles Work? | 38](#)
- [Caveats | 40](#)

Tagging allows you to insert fields, values, and keys into a Paragon Insights rule when certain conditions are met.

Paragon Insights supports the following types of tagging:

- **Static Tagging**

In static tagging, the tagging profile is applied to values stored in the time series data base (TSDB). These values do not vary a lot with time. In static tagging, you can avoid using *When* statements, and

you can add *Then* statements individually to a row of the TSDB. You can add tags to all rows since no conditions are present.

- **Dynamic Tagging**

Paragon Insights Release 4.0.0 supports dynamic tagging where conditions used in Paragon Insights tagging are checked against values that are stored in Redis database. This database acts like a cache memory that stores dynamic data. Dynamic data is real-time data that is stored in Redis database.

Tagging Profile Terminology

The following list describes the tagging profile terminologies:

Policy

A policy is the top-level element in a tagging profile. You can add multiple policies within a single tagging profile. Multiple policies that exist within a tagging profile can have their own rules and terms.

Usage Notes:

- Defining multiple policies within a single profile allows you to define terms for each rule in one profile rather than having to create one profile for each rule.

Rules

A rule is any defined Paragon Insights rule. The rule element type in a tagging profile is a list element. You can apply a specific policy profile to the rule(s) (*[rule1, rule2]*) included within the tagging profile.

Usage Notes:

You can describe the topic-name/rule-name requirement for the rules element in the following ways:

- To name specific rules within a tagging profile, use the form: *topic-name/rule-name*.

For example, *protocol.bgp/check-bgp-advertised-routes*. Navigate to **Configuration>Rules** to view configured rules.

- Use an asterisk (*) with no other value or brackets to match all rules.
- Use python-based fnmatch patterns to select all rules within a specific topic. For example, *line-cards/**.

For more information, see [fnmatch — Unix filename pattern matching](#).

Terms

The term section of the tagging profile is where the match conditions are set and examined, and actions based on those matches are set and carried out. Set the conditions for a match in a *when statement*. Set the actions to be carried out upon completing a match in one or more *then statements*.

Usage Notes:

- Each term can contain a *when* statement but it is not mandatory.
- Each term must contain at least one *then* statement.
- Multiple terms can be set within a single policy.
- Terms are processed sequentially from top to bottom until a match is found. If a match is found, processing stops after completing the statements found in the *then* section. Other terms, if present, are not processed unless the *next* flag is enabled within the matched term. If the matched term has the next flag enabled, then subsequent terms are processed in order.

When Statements

When statements define the match conditions that you specify. *When* statements ultimately resolve to be true or false. You can define a *term* without a *when* statement. This equates to a default *term* wherein the match is assumed true and the subsequent *then* statement is carried out. Conversely, multiple conditions can be checked within one *when* statement.

If one or more of the conditions set forth in a *when* statement are not met, the statement is false and the *term* has failed to match; processing moves to the next *term*, if present.

Usage Notes:

When statements perform boolean operations on the received data to determine if it matches the criteria you set. The supported operations are:

- Numeric Operations:
 - equal-to
 - not-equal-to
 - greater-than
 - greater-than-or-equal-to
 - less-than
 - less-than-or-equal-to

- String Operations:
 - matches-with
 - does-not-match-with
- Time Operations:
 - matches-with-scheduler

NOTE: The `matches-with-scheduler` option requires that a discreet scheduler be configured in the **Administration > Ingest Settings > Scheduler** page. The name of the scheduler can then be used in the `matches-with-scheduler` *when* statement

- Go Language Expressions:
 - `eval <simple-go-expression>`

For example: `eval a + b <= c.`

Then Statements

Then statements implement the tagging instructions that you provide. This is done only after there is a complete match of the conditions set forth in a *when* statement contained in the same *term*. Each *term* defined must have at least one *then* statement. Each *then* statement must have one or more than one action(s) defined; the actions available in *then* statements are:

add-field Adds a normal field to the rule(s) listed in the rule section.

Multiple fields can be added within a *then* statement. The add-field action requires that you also define the kind of field you are adding with the *field-type* parameter:

- string
- integer
- float
- unsigned integer

Starting in Paragon Insights Release 4.2.0, you can also select unsigned integer as a name field data type.

NOTE: If you do not define a field type, the new field gets added with the default field-type of string.

add-key Adds a key field with string data type to the rule(s). Added key fields are indexed and can be searched for just like any other key field.

Usage Notes:

- You can set the *next* flag to true within a then statement. When this flag is set to true, the next term in the policy gets evaluated if all of the conditions of the current term match.

Example Configuration: Elements of a Tagging Profile

Paragon Insights configuration elements are displayed as pseudo-config. This configuration resembles the hierarchical method used by Junos OS.

["Elements of a Tagging Profile" on page 37](#) shows how tagging profile elements are named and how they are related to each other.

Elements of a Tagging Profile

```
healthbot {
  ingest-settings {
    data-enrichment {
      tagging-profile <tagging-profile-name> {
        policy <policy-name> {
          rules [ List of Rules ];
          term <term-name1>
          {
            when {
              <condition1>
              <condition2>
            }
            then {
add-field <field-name1>
              {
                value <field-value1>;
                type <field-type>;
              }
add-field <field-name2>
              {
```


Table 3: Fields in NetFlow Stream (Continued)

source-port	destination-port	protocol	derived-application
Any	2541	6 (TCP)	
1755	Any	17 (UDP)	MS-streaming
Any	830	6 (TCP)	netconf-ssh
7802	Any	17 (UDP)	vns-tp

In [Table 3 on page 38](#), you use three existing fields in a NetFlow stream to guess the application traffic in the stream. You then create a new field called *derived-application* and populate it based on the values seen in the traffic.

You can apply tagging profiles at the device group level. See ["Example pseudo-configuration" on page 39](#).

- When a device in a device group has a tagging profile applied to it, and the device group has another tagging profile applied to the whole group of devices, the tagging profile of the device group is merged with the existing tagging profile of the device.

For example, D-A-Net is a device that is part of a device group called Group-D1. D-A-Net has a tagging profile applied to it. There is another tagging profile applied on the device group, Group-D1, as well. In such a scenario, the tagging profile applied to the device group is merged with the tagging profile of the device, D-A-Net.

- When the tagging profile applied to the device group and the tagging profile applied to the device in the group renders the same output, the tagging profile of the device is preserved.

Example pseudo-configuration shown below

```

device r0 {
    host r0;
    tagging-profile [ profile1 ]
}
device r1 {
    host r1
}
device-group core {
    devices [ r0 r1 ];

```

```
tagging-profile [ profile2 ]
}
```

In this example, device *r0* has tagging profile, *profile1*, assigned at the device level and tagging profile, *profile2*, assigned by its membership in the device- group (*core*).

Device *r1* has tagging profile, *profile2*, assigned by its membership in device group, *core*.

In this scenario, *profile1* and *profile2* are merged on device *r0*. However, if *profile1* and *profile2* both define the same fields but the fields contain different values, the value from *profile1* takes precedence because it is assigned directly to the device.

Device *r1* only gets tagging profile *profile2*.

Caveats

- Fields and keys added using tagging profiles cannot be used within periodic aggregation fields. This is because periodic aggregation must take place before any UDFCode function (reference, vector, UDF, ML) is applied.
- Tagging profiles can consist of only fields in add-key or add-field. Vectors cannot be added to a rule by a tagging profile.
- Vector comparison operations cannot be used within tagging profile terms. Only field Boolean operations are permitted.
- For tagging profile conditional operations within a *when* statement, the used field must be of type sensor, constant, or reference.

This is applicable only in static tagging.

- If the field used within tagging profile Boolean operation is of type reference, then this reference field must not depend on any user-defined-function or formula defined within the same rule.

Types of Tagging

IN THIS SECTION

- [Static Tagging | 41](#)
- [Dynamic Tagging | 43](#)

Paragon Insights supports static tagging and dynamic tagging.

Static Tagging

In static tagging, the tagging profile is applied to values stored in the time series data base (TSDB). These values do not vary a lot with time. In static tagging, you can avoid using *When* statements, and you can add *Then* statements to a tagging profile.

Sample Static Tagging Configuration

```
healthbot {
  ingest-settings {
    data-enrichment {
      tagging-profile profile {
        policy policy1 {
          rules *;
          term term1 {
            then {
              add-key "tenant-id" {
                value tenant1;
              }
            }
          }
        }
      }
    }
  }
}
```

In this sample static tagging configuration, the lack of a *when* statement means that any device that this tagging profile is applied to will have the field *tenant-id* assigned with the value *tenant1*. The fields and values defined in this profile are assigned to all rules that are applied to a device or device-group because of the *** in the rules parameter.

You can also create a static tagging profile from the Paragon Insights graphical user interface (GUI). Navigate to **Configuration > Sensor > Settings > Tagging Profile** page to create a tagging profile.

Application Identification

[Table 3 on page 38](#) shows an example application identification scenario based on source-port, destination-port, and protocol of traffic seen in a NetFlow stream.

To create the derived-application field as given in [Table 3 on page 38](#) from the received data (data under source-port, destination port, and protocol), you must use a tagging profile definition that looks like this:

```
healthbot {
  ingest-settings {
    data-enrichment {
      tagging-profile profile1 {
        policy policy1 {
          rules *;
          term term1 {
            when {
              matches-with "$source-port" "$netchat-source-port";
              matches-with "$protocol" "6 (TCP)";
            }
            then {
              add-key "application" {
                value netchat;
              }
            }
          }
          term term2 {
            when {
              matches-with "$protocol" "6 (TCP)";
              matches-with "$destination-port" "$netchat-dest-port";
            }
            then {
              add-key "application" {
                value netchat;
              }
            }
          }
          term term3 {
            when {
              matches-with "$source-port" "$ms-streaming-source-port";
              matches-with "$protocol" "17 (UDP)";
            }
            then {
              add-key "application" {
                value ms-streaming;
              }
            }
          }
        }
      }
    }
  }
}
```


Understanding Redis Database and Dynamic Tagging Configurations

Understanding Redis Database and Dynamic Tagging Configurations

- Key structure is <Device-group-name>:<device-id>::key-name__network:<network-group-name>::key-name , where :: is the key separator.

Example key structures:

- Device Group

Syntax:

<Device-group-name>:<device-id>::key-name

Example:

Device GroupCore:r1::/components/

- Network Group

Syntax:

__network:<network-group-name>::key-name

Example:

network::net_check::topic/rule

- Values are stored in JSON string format <json dump as string> in Redis. However, values are provided in string, integer, and float formats.

Example value formats:

- Core:r1::/components/= value1
- Core:r1::/components/='{{"key1": value1, "key2": value2}'
- Core:r1::/components/='{{"key1": {"key2": value1, "key3": value2}'
- Core:r1::/components/='{{"key1": {"key2": "[list of values]", "key3": value1}'
- Sample tagging-profile configurations using when statement.

```
"when" : {
    "matches-with" : [
        {
            "left-operand" : "$field1",
            "right-operand" : "/interfaces/.key1",
```

```

        "in-memory": true
      }
    ]
  }
}

```

- Use a . operator between interfaces.

In the following example, key3 interface is nested within key2 interface in the right operand.

```

"when" : {
  "matches-with" : [
    {
      "left-operand" : "$field1",
      "right-operand" : "/interfaces/.key2.key3",
    }
  ]
}

```

- • Sample tagging-profile configurations using then statement.

```

"then" : {
  "add-field" : [
    {
      "name" : "field1",
      "value" : "redis-key",
      "type" : "integer",
      "in-memory": true
    }
  ]
}

```

- Use a . operator between interfaces.

In the following example, key3 interface is nested within key2 interface in the right operand.

```

"then" : {
  "add-field" : [
    {
      "name" : "field2",

```

```

        "value" : "redis-key1.redis-key2.redis-key3",
        "type" : "integer",
        "in-memory": true
    }
]
}

```

- Using exist operator in configurations.
- Using exist as key.

Redis Data Structure

```

"Core:r1::/interfaces/" = '{"ge-1/0/2": {"key1": value1, "key2": value2}}'

```

tagging-profile Using when Statement

```

"when": {
    "exists": {
        "field": "$interface-name",
        "path": "/interfaces/",
        "in-memory": true
    }
    "then": { do-something.. }
}

```

- Using exist as value in list.

Redis Data Structure

```

"Core:r1::/interfaces/" = '{"key1": {"key2": ['ge-1/0/2', 'ge-1/0/3'], "key3": value1}}'

```

tagging-profile Using when Statement

```

"when": {
    "exists": {
        "field": "$interface-name",
        "path": "/interfaces/.key1.key2" ,
        "in-memory": true
    }
},

```

```

    "then": {
      "add-field": [
        "name": "field1",
        "value": "/interfaces/.key1.key3",
        "in-memory": true
      ]
    }
  }

```

- Using \$ in **then** statements.

When you use \$<field-name> within a Redis key, \$<field-name> is replaced with a value from the already processed database value.

As an example, consider that ge-1/0/2 is present within Redis key.

Redis Data Structure

```

"Core:r1::/interfaces/" = '{"ge-1/0/2": {"key1": value1, "key2": value2},
  "ge-1/0/3": {"key1": value1, "key2": value2}}'

```

Example tagging -profile

```

"when": {
  "exists": [
    {
      "field": "$interface-name",
      "path": "/interfaces/",
      "in-memory": true
    }
  ],
  "greater-than": [
    {
      "left-operand": "30",
      "right-operand": "/interfaces/.$interface-name.key1",
      "in-memory": true
    }
  ]
},
"then": {
  "add-field": [
    "name": "interface-meta-data",

```

```

        "value": "/interfaces/.$interface-name.key2",
        "in-memory": true
    ]
}

```

In this scenario, the tagging-profile checks if `$interface-name` is present in the Redis database, and if `key1` value for the given interface name is greater than 30. If the statement is true, the tagging-profile fetches `key2` value from `name` field. In this example tagging profile, the `name` value is `interface-meta-data`.

- To enable dynamic tagging, set `in-memory` value to `true`.

By default `in-memory` value is set to `false`.

```

"when": {
  "exists": {
    "field": "$interface-name",
    "path": "/interfaces/.key1.key2" ,
    "in-memory": true
  }
}
"then": {
  "add-field": [
    "name": "interface-meta-data",
    "value": "/interfaces/.$interface-name.key2",
    "in-memory": true
  ]
}

```

Add a Tagging Profile

You can use the Paragon Insights graphical user interface (GUI) to add static tagging and dynamic tagging profiles.

Adding a Static Tagging Profile

To add a static tagging profile:

1. Navigate to **Settings > Ingest**.

The **Ingest Settings** page is displayed.

2. Click the **Tagging Profile** tab and then click the plus (+) icon to add a tagging profile.

The **Create Tagging Profile** page is displayed.

3. Enter the following information in the **Create Tagging Profile** page:

- a. Enter a name for the tagging profile in the **Profile Name** text box.

The maximum length is 64 characters.

Regex pattern: "[a-zA-Z][a-zA-Z0-9_-]*"

- b. Click the plus (+) icon under Policies to define a policy for this tagging profile.

You can define one or more policies.

The **Policies** section is displayed.

- i. Enter a name for the new policy in the **Policy Name** text box.

The maximum length is 64 characters.

Regex pattern: "[a-zA-Z][a-zA-Z0-9_-]*"

- ii. Enter a rule that you want to apply to this tagging profile. The rule can contain an *fnmatch* expression.

You can apply one or more than one rule to a profile. A rule is any defined Paragon Insights rule.

- c. Click the add (+) icon under **Terms** to define a list of conditions.

- i. Enter a name for the match condition in the **Term Name** text box.

- ii. Configure When and Then statements.

You can define tagging instructions in a Then statement. After the conditions that you set in a When statement are met, the Then statement is implemented. Starting in Paragon Insights 4.2.0, When statements are mandatory in static tagging.

To configure a **Then** statement:

1. Click the plus (+) icon to add a key to the rules listed.

The **Key Name** and **Value** text boxes are displayed.

- Enter a name for the key in the **Key Name** text box.

The maximum length is 64 characters.

Regex pattern: "[a-zA-Z][a-zA-Z0-9_-]*"

This name is added as the key field for all rules configured within the tagging profile rules section.

- Enter a value that you want to associate to the key, in the **Value** text box.

2. Click the plus (+) icon to add a field to the rules listed.

The **Field Name** and **Value** text boxes, and the **Type** drop-down list are displayed.

- Enter the name in the **Field Name** text box.
- Enter a value in the Value text box.
- Select the field type from the **Type** drop-down list.

String type is selected by default.

Starting in Paragon Insights Release 4.2.0, you can also select unsigned integer as a name field data type. An unsigned integer is a data type that can contain values from 0 through 4,294,967,295.

- Click **OK**.

3. Set the **Evaluate next term** flag to **True** to evaluate conditions in the next term. Evaluate next term only if the first condition is satisfied.

By default, the **Evaluate next term** flag is set to **False**.

4. Click **Save** to only save the configuration.

Click **Save & Deploy** to save and deploy the configuration immediately.

Adding a Dynamic Tagging Profile

To configure a dynamic tagging profile with Redis:

1. Navigate to **Settings > Ingest**.

The **Ingest Settings** page is displayed.

2. Click the **Tagging Profile** tab and then click the plus (+) icon to add a tagging profile.

The **Create Tagging Profile** page is displayed.

3. Enter the following information in the **Create Tagging Profile** page.

- a. Enter a name for the tagging profile in the **Profile Name** text box.

The maximum length is 64 characters.

Regex pattern: "[a-zA-Z][a-zA-Z0-9_-]*"

- b. Click the plus (+) icon under **Policies** to define a policy for this tagging profile.

You can define one or more policies.

The **Policies** section is displayed.

- i. Enter a name for the new policy in the **Policy Name** text box.

The maximum length is 64 characters.

Regex pattern: “[a-zA-Z][a-zA-Z0-9_-]*”

- ii. Enter a rule that you want to apply to this tagging profile. The rule can contain an *fnmatch* expression.

You can apply one or more rules to a profile. A rule is any defined Paragon Insights rule.

- c. Click the plus (+) icon under **Terms** to define a list of conditions.

- i. Enter a name for the match condition in the **Term Name** text box.

- ii. Configure When and Then statements:

You set conditions for a match in a when statement. To configure **When** statement,

1. Click the **Edit (pencil)** icon.

The **When Condition** page is displayed.

2. Click **+ Add another when** to view the **Operator** drop-down list.

3. Select a boolean operation that you want to apply to incoming data from the **Operator** drop-down list.

The **Left Operand** and **Right Operand** text boxes are displayed.

NOTE: **+ Add another when** is automatically renamed to the operator condition that you selected.

- Enter the value of the left operand of assignment that you selected, in the **Left Operand** text box.

You can use \$ as prefix to populate database values. For example, \$memory. However, using \$ as prefix is not mandatory.

- Enter the value of the right operand of assignment that you selected, in the **Right Operand** text box.

This value is populated from the Redis database.

- Set the **Evaluate in Memory** flag to **True** to populate data from the Redis database.

By default, the **Evaluate in Memory** flag is set to **False**. When the flag is set to false, data is populated from the TSDB.

- Click **OK**.
- Set the **Evaluate next term** flag to **True** to evaluate conditions in the next term. After the first condition is satisfied, the conditions in the next term are evaluated.

By default, the **Evaluate next term** flag is set to **False**.

You can define tagging instructions in a **Then** statement. After the conditions that you set in a **When** statement are met, the **Then** statement is implemented. Starting in Paragon Insights Release 4.2.0, **When** statements are mandatory in tagging profile.

To configure a **Then** statement:

1. Click the plus (+) icon to add a key to the rules listed.

The **Key Name** and **Value** text boxes are displayed.

- Enter a name for the key in the **Key Name** text box.

The maximum length is 64 characters.

Regex pattern: "[a-zA-Z][a-zA-Z0-9_-]*"

This name will be added as key field for all rules configured within the tagging profile rules section.

- Enter a value that you want to associate with the key, in the **Value** text box.

2. Click the plus (+) to add a text box to the rules listed.

The **Field Name** and **Value** text boxes, and the **Type** drop-down list are displayed.

- Enter the name in the **Field Name** text box.
- Enter a value in the **Value** text box.
- Select the field type from the **Type** drop-down list.

String type is selected by default.

Starting in Paragon Insights Release 4.2.0, you can also select unsigned integer as a name field data type. An unsigned integer is a data type that can contain values from 0 through 4,294,967,295.

3. Set the **Evaluate in Memory** flag to **True** to populate data from the Redis database.

By default, the **Evaluate in Memory** flag is set to **False**.

4. Click **OK**.

5. Set the **Evaluate next term** flag to **True** to evaluate conditions in the next term. The next term is evaluated only if the first condition is satisfied.

By default, the **Evaluate next term** flag is set to **False**.

4. Click **Save** to only save the configuration.

Click **Save & Deploy** to save and deploy the configuration immediately.

Apply a Tagging Profile

You can configure a tagging profile to insert fields, values, and keys into a Paragon Insights rule. You can also set conditions that are checked against values stored in the times series database (TSDB) or Redis database.

After you have created a tagging profile from the Paragon Insights graphical user interface (GUI), you can apply a tagging profile to:

- a new device
- to an existing device
- to a new device group
- to an existing device group

Follow these steps to apply a tagging profile.

To apply a tagging profile to a new device:

1. Navigate to **Configuration > Device**.

The **Device Configuration** page is displayed.

2. Click (+) icon to add a new device.

The **Add Device(s)** page is displayed.

3. After you have entered the necessary information to add a device, click the **Tagging Profiles** section.
4. Select the tagging profile you want to apply to the device, from the **Tagging Profiles** drop-down list.
5. Click **Save** to only save the configuration.

Click **Save & Deploy** to save and immediately deploy the new configuration.

To apply a tagging profile to an existing device:

1. Navigate to **Configuration > Device**.

The **Device Configuration** page is displayed.

2. Select the check box next to the name of the device and click **Edit device**.

The Edit “*device*” page is displayed.

3. Click the **Tagging Profiles** section to view the **Tagging Profiles** drop-down list.
4. Select the tagging profile you want to apply to the device, from the **Tagging Profiles** drop-down list.
5. Click **Save** to only save the configuration.

Click **Save & Deploy** to save and immediately deploy the new configuration.

To apply a tagging profile to a new device group:

1. Navigate to **Configuration > Device Group**.

The **Device Group Configuration** page is displayed.

2. Click the add (+) icon to add a new device group.

The **Add Device Group** page is displayed.

3. After you have entered the necessary information to add a device group, click the **Tagging Profiles** section.
4. Select the tagging profile you want to apply to the device, from the **Tagging Profiles** drop-down list.
5. Click **Save** to only save the configuration.

Click **Save & Deploy** to save and immediately deploy the new configuration.

To apply a tagging profile to an existing device group:

1. Navigate to **Configuration > Device Group**.

The **Device Group Configuration** page is displayed.

2. Select the check box next to the name of the device group and click the **Edit device group** icon.

The Edit “*device*” page is displayed.

3. Click the **Tagging Profiles** section to view the **Tagging Profiles** drop-down list.
4. Select the tagging profile you want to apply to the device group, from the **Tagging Profiles** drop-down list.
5. Click **Save** to only save the configuration.

Click **Save & Deploy** to save and immediately deploy the new configuration.

NOTE:

- When a device in a device group has a tagging profile applied to it, and the device group has another tagging profile applied to the whole group of devices, the tagging profile of the device group is merged with the existing tagging profile of the device.

For example, D-A-Net is a device that is part of a device group called Group-D1. D-A-Net has a tagging profile applied to it. There is another tagging profile applied on the device group, Group-D1, as well. In such a scenario, the tagging profile applied to the device group is merged with the tagging profile of the device, D-A-Net.

- When the tagging profile applied to the device group and the tagging profile applied to the device in the group renders the same output, the tagging profile of the device is preserved.

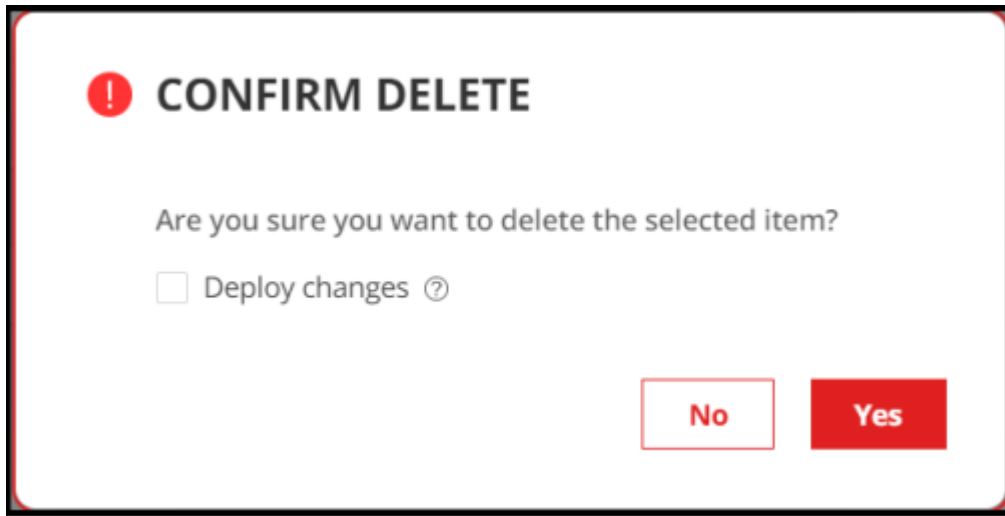
Delete a Tagging Profile

To delete a tagging profile:

1. Click **Settings > Ingest** from the left-nav bar.
The **Ingest Settings** page is displayed.
2. Click the **Tagging Profile** tab to view the **Tagging Profile** page.
3. Select the tagging profile that you want to delete, and click the **delete (trash can)** icon.

The **CONFIRM DELETE** pop-up appears.

Figure 4: Confirm Delete Pop-up



4. Do any one of the following:

- Click **Yes** to delete the tagging profile from the database. However, the changes are not applied to the ingest service.

NOTE:

- We recommended that you do not delete a tagging profile that is currently in use.
 - After you delete a tagging profile from the database, you cannot associate that tagging profile with another device or device group even if you have not deployed changes.
 - You can also deploy changes to the ingest service or roll back the changes that you have already deleted, from the **PENDING CONFIGURATION** page. For more information, see "[Commit or Roll Back Configuration Changes in Paragon Insights](#)" on [page 307](#).
- Select the **Deploy changes** check box and then click **Yes** to delete the tagging profile from the database, and to apply the changes to the ingest service.
 - (Optional) Click **No** to cancel this operation.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.0.0	Paragon Insights Release 4.0.0 supports dynamic tagging where conditions used in Paragon Insights tagging are checked against values that are stored in Redis database.

Paragon Insights Time Series Database (TSDB)

IN THIS SECTION

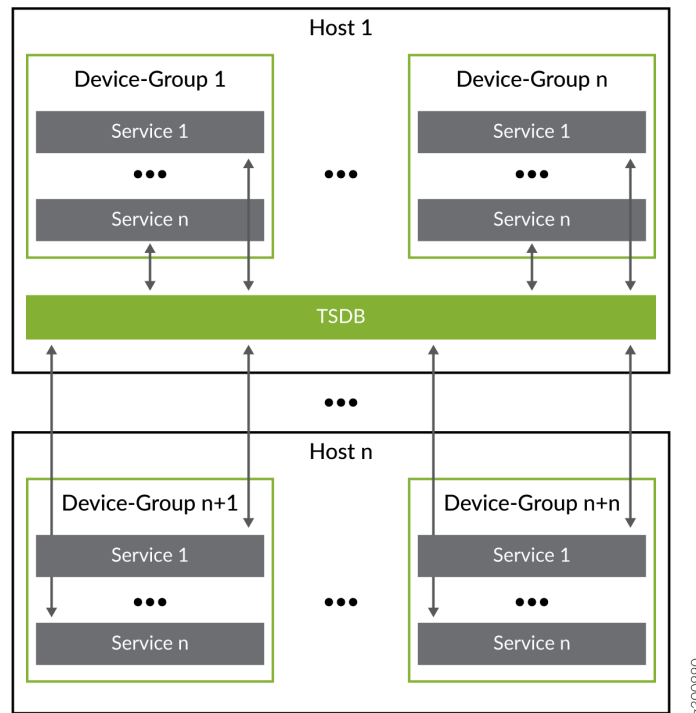
- [Historical Context | 57](#)
- [TSDB Improvements | 58](#)
- [Database Sharding | 59](#)
- [Database Replication | 60](#)
- [Database Reads and Writes | 61](#)
- [Manage TSDB Settings in the Paragon Insights GUI | 62](#)
- [Paragon Insights CLI Configuration Options | 64](#)

Paragon Insights (formerly HealthBot) collects a lot of data through its various ingest methods. All of that data is time sensitive in some context. This is why Paragon Insights uses a time series database (TSDB) to store and manage all of the information received from the various network devices. This topic provides an overview of the TSDB as well some guidance on managing it.

Historical Context

In releases earlier than HealthBot Release 3.0.0, there was one TSDB instance regardless of whether you ran Paragon Insights as a single node or as a multi-node (Docker Ccompose) installation. [Figure 5 on page 58](#) shows a high-level view of what this looked like.

Figure 5: Single TSDB Instance - Releases earlier than HealthBot Release 3.0.0



This arrangement left no room for scaling or redundancy of the TSDB. Without redundancy, there is no high availability (HA); A failure left you with no way to continue operation or restore missing data. Adding more Docker Compose nodes to this topology would only provide more Paragon Insights processing capability at the expense of TSDB performance.

TSDB Improvements

To address these issues and provide TSDB high availability (HA), three new TSDB elements are introduced in HealthBot Release 3.0.0, along with clusters of Paragon Insights nodes* for the other Paragon Insights microservices:

- – How many servers, or nodes, are available to store TSDB data and scale Paragon Insights?
- – How many copies of your data do you want to keep?
- – How is data written and read back from the TSDB? What happens when something goes wrong?

NOTE: *Paragon Insights uses Kubernetes for clustering its Docker-based microservices across multiple physical or virtual servers (nodes). Kubernetes clusters consist of a primary node and multiple worker nodes. During the healthbot setup portion of Paragon Insights multinode installations, the installer asks for the IP addresses (or hostnames) of the Kubernetes primary node and worker nodes. You can add as many worker nodes to your setup as you need, based on the required replication factor for the TSDB databases. The number of nodes you deploy should be at least the same as the replication factor. (See the following sections for details).

For the purposes of this discussion, we refer to the Kubernetes worker nodes as Paragon Insights nodes. The primary node is not considered in this discussion.

Database Sharding

Database sharding refers to selectively storing data on certain nodes. This method of writing data to selected nodes distributes the data among available TSDB nodes and permits greater scaling since each TSDB instance then handles only a portion of the time series data from the devices.

To achieve sharding, Paragon Insights creates one database per device group/device pair and writes the resulting database to a specific (system determined) instance of TSDB hosted on one (or more) of the Paragon Insights nodes.

For example, say we have two devices, D1 and D2 and two device groups, G1 and G2. If D1 resides in groups G1 and G2, and D2 resides only in group G2, then we end up with 3 databases: G1:D1, G2:D1, and G2:D2. Each database is stored on its own TSDB instance on a separate Paragon Insights node as shown in [Figure 6 on page 60](#) below. When a new device is on-boarded and placed within a device group, Paragon Insights chooses a TSDB database instance on which to store that device/device-group data.

Figure 6: Distributed TSDB

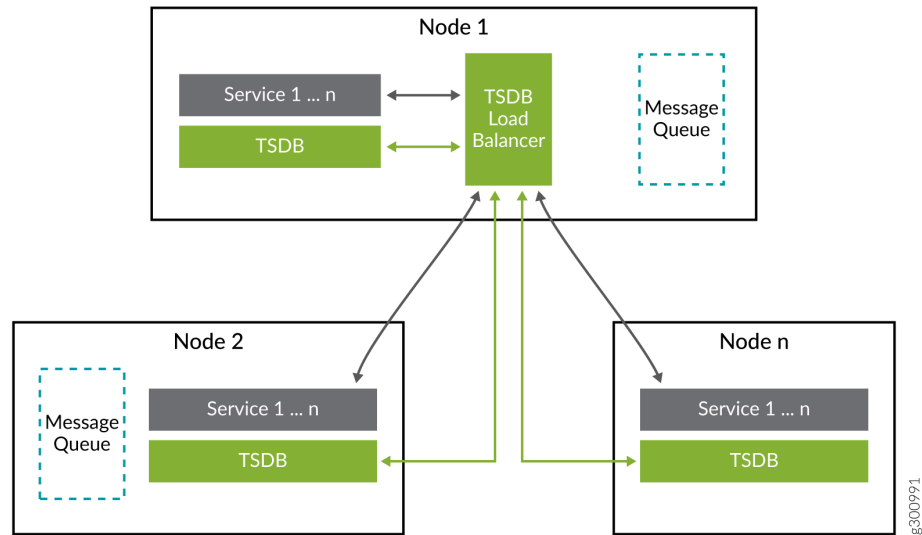


Figure 6 on page 60, above, shows 3 Paragon Insights nodes, each with a TSDB instance and other Paragon Insights services running.

NOTE:

- A maximum of 1 TSDB instance is allowed on any given Paragon Insights node. Therefore, a Paragon Insights node can have 0 or 1 TSDB instances at any time.
- A Paragon Insights node can be dedicated to running only TSDB functions. When this is done, no other Paragon Insights functions run on that node. This prevents other Paragon Insights functions from starving the TSDB instance of resources.
- We recommend that you dedicate nodes to TSDB to provide the best performance.
- Paragon Insights and TSDB nodes can be added to a running system using the Paragon Insights CLI.

Database Replication

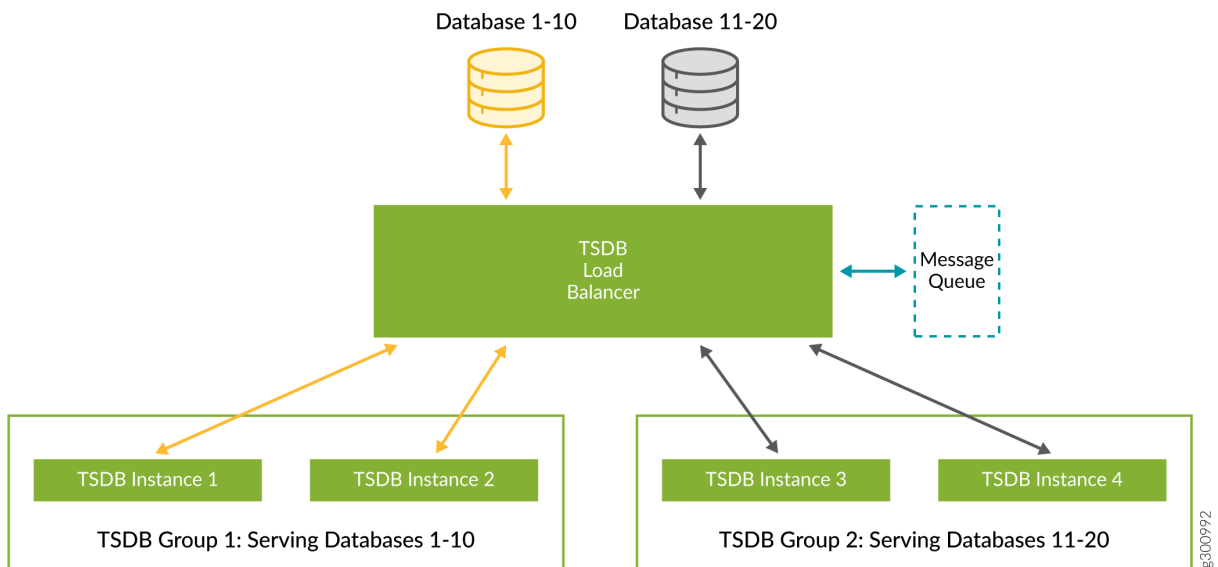
As with any other database system, replication refers to storing the data in multiple instances on multiple nodes. In Paragon Insights, we establish a replication factor to determine how many copies of the database are needed.

A replication factor of 1 only creates one copy of data, and therefore, provides no HA. When multiple Paragon Insights nodes are available and replication factor is set to 1, then only sharding is achieved.

The replication factor determines the minimum number of Paragon Insights nodes needed. A replication factor of 3 creates three copies of data, requires at least 3 Paragon Insights nodes, and provides HA. The higher the replication factor, the stronger the HA and the higher the resource requirements in terms of Paragon Insights nodes. If you want to scale your system further, you should add Paragon Insights nodes in exact multiples of the replication factor, or 3, 6, 9, etc.

Consider an example where, based on device/device-group pairing mentioned earlier, Paragon Insights has created 20 databases. The Paragon Insights system in question has a replication factor of 2 and has 4 nodes running TSDB. Based on this, two TSDB replication groups are created; in our example they are *TSDB Group 1* and *TSDB Group 2*. In [Figure 7 on page 61](#) below, the data from databases 1-10 is being written to TSDB instances 1 and 2 in TSDB group 1. Data from databases 11-20 is written to TSDB instances 3 and 4 in TSDB group 2. The outline around the TSDB instances represents a TSDB replication group. The size of the replication group is determined by the replication factor.

Figure 7: TSDB Databases



Database Reads and Writes

As shown in [Figure 6 on page 60](#), Paragon Insights can make use of a distributed messaging queue. In cases of performance problems or errors within a given TSDB instance, this allows for writes to the

database to be performed in a sequential manner ensuring that all data is written in proper time sequence.

All Paragon Insights microservices use standardized database query (read) and write functions that can be used even if the underlying database system is changed at some point in the future. This allows for flexibility in growth and future changes. Other read and write features of the database system include:

- In normal operation, database writes are sent to all TSDB instances within a TSDB group.
- Database writes can be buffered up to 1GB per TSDB instance so that failed writes can be retried until successful.
- If problems persist and the buffer fills up, the oldest data is dropped in favor of new data.
- When buffering is active, database writes are performed sequentially so that new data cannot be written until the previous write attempts are successful.
- Database queries (reads) are sent to the TSDB instance which has reported the fewest write errors in the last 5 minutes. If all instances are performing equally, then the query is sent to a random TSDB instance in the required group.

Manage TSDB Settings in the Paragon Insights GUI

You can use the Paragon Insights GUI to configure the time series database (TSDB) settings.

To configure TSDB settings:



WARNING: Selecting, deleting, or dedicating TSDB nodes must be done during a maintenance window because some services will be restarted and the Paragon Insights GUI will likely be unresponsive.

1. Select **Settings > System**.

The System Settings page appears.

2. Click **Time Series Database**.

The **TSDB Settings** page appears.

3. From the **TSDB Settings** page, you can:

- a. Select one or more nodes (from the **TSDB Nodes** list) to be used as TSDB nodes.

(The **TSDB Nodes** list displays the available nodes in the Paragon Insights installation that you can select as TSDB nodes. By default, Paragon Insights automatically selects one node as a TSDB node.)

- b. Set the replication factor by typing a value (or by using the arrows to specify a value) in the **Replication Factor** text box.

(The replication factor determines how many copies of the database are needed. The replication factor is set to 1 by default.)

- c. Dedicate nodes as TSDB nodes by clicking the **Dedicate** toggle to turn it on.

A TSDB node might have more than one microservice running. However, when you dedicate a node as TSDB node, it runs only the TSDB microservice, and stops running all other microservices.

NOTE:

- If the node is associated to a persistent volume (storage in a cluster), then you cannot use that node as a dedicated TSDB node.
- A fail-safe mechanism ensures that you cannot dedicate all Paragon Insights nodes as TSDB nodes.

- d. Ignore system errors (when you remove or replace a failed TSDB node from Paragon Insights) by clicking the **Force** toggle to turn it on.

For example, when a TSDB node fails and the replication factor for that node is set to one, the TSDB data for that node is lost. In this scenario, the failed TSDB node must be removed from Paragon Insights. However, when you try to replace the failed node with a new node, the backup of the node fails with a system error because the replication factor was set to one. If you want to proceed with replacing the node, you must turn the **Force** toggle on.

- e. Delete a node that was previously assigned as a TSDB node by clicking **X** next to the name of the TSDB node. The node is removed as a TSDB node when you deploy the new configuration changes.

4. Do one of the following:

- Click **Save** to only save the configuration changes to the database without applying the changes to the TSDB nodes.

You must commit (or rollback) the configuration changes later. For more information, see ["Commit or Roll Back Configuration Changes in Paragon Insights" on page 307](#).

- Click **Save & Deploy** to save configuration changes to the database and to apply the changes to the TSDB nodes.

5. In the pop-up that appears, click **OK** to confirm.

You are returned to the TSDB Settings page.

Paragon Insights CLI Configuration Options

The Paragon Insights CLI provides a means to add and delete TSDB nodes from the system and to change the replication factor as a result.

Add a TSDB Node to Paragon Insights

```
# set healthbot system time-series-database nodes <IP address or hostname> dedicate <true or false>
```

or

```
#set healthbot system time-series-database nodes [ space-separated list of IP addresses or hostnames] dedicate <true or false>
```

Manage the Replication Factor

```
# set healthbot system time-series-database replication-factor <replication-factor>
```

Set the replication factor to a multiple of the number of TSDB nodes present in the system. If you have two TSDB nodes, set the replication factor at 2, 4, 6, etc.

Usage Notes

- Paragon Insights performs a ping to determine if the new node(s) is reachable. A warning is shown if the ping fails.
- The *dedicate* option specifies whether or not the TSDB nodes perform only TSDB functions.

RELATED DOCUMENTATION

[Paragon Insights Installation Guide](#)

Paragon Insights Machine Learning (ML)

IN THIS SECTION

- [Paragon Insights Machine Learning Overview | 65](#)
- [Understanding Paragon Insights Anomaly Detection | 66](#)
- [Understanding Paragon Insights Outlier Detection | 68](#)
- [Understanding Paragon Insights Predict | 72](#)
- [Paragon Insights Rule Examples | 73](#)

Paragon Insights Machine Learning Overview

Paragon Insights (formerly HealthBot) uses machine learning to detect anomalies and outliers, and predict future device or network-level behavior. The machine learning-enabled Paragon Insights features include:

Anomaly Detection

Anomaly detection using the Paragon Insights Anomaly Detection algorithms involves comparison of new data points with data points collected from the same device during a specific learning period. Paragon Insights supports the following machine learning algorithms for anomaly detection:

- 3-sigma
- K-means
- Holt-Winters

Anomaly detection can be activated within Paragon Insights rules by setting a rule field's ingest type to formula, and then selecting anomaly detection. (**Configuration > Rules > Fields tab > Ingest Type > Formula**).

Outlier Detection

Outlier detection using the Paragon Insights Outlier Detection algorithms involves analysis of data from a collection of devices across your network during a specific learning period. Paragon Insights supports the following machine learning algorithms for outlier detection:

- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
- K-fold cross-validation using 3-sigma (k-fold/3-sigma)

Prediction Prediction of future device or network-level behavior involves using the Paragon Insights median prediction machine learning algorithm or, the Holt-Winters prediction algorithm.

Starting with HealthBot Release 3.1.0, you can choose the Holt-Winters prediction algorithm from **Configuration > Rules > Fields > Ingest Type > Formula**.

Understanding Paragon Insights Anomaly Detection

IN THIS SECTION

- [Field | 66](#)
- [Algorithm | 66](#)
- [Learning period | 67](#)
- [Pattern periodicity | 68](#)

This section describes the input parameters associated with Paragon Insights rules configured to detect anomalies using Anomaly Detection algorithms. Once the machine learning models are built, they can be used in production to classify new data points as normal or abnormal. The accuracy of the results increases with larger amounts of data.

Field

To apply a machine learning algorithm, you must first define the numeric data field on which to apply the algorithm. For information on how to create a user-defined data field for a Paragon Insights rule, see the Fields section in the Paragon Insights User Guide.

Algorithm

The Paragon Insights Anomaly Detection algorithms include Holt-Winters, 3-sigma and k-means:

- Holt-Winters** The Holt-Winters algorithm uses traffic entropy measurements and seasonal variations in traffic to detect anomalous traffic flowing through an interface. The seasonality aspect provides a means to de-emphasize normal increases and decreases in traffic that happen regularly over time intervals. For example, network traffic in an enterprise network could be expected to have a weekly seasonality since there would be significantly more traffic on the network during the work week than on the weekend.
- Since Holt-Winters can predict a traffic drop starting on Friday evening, an anomaly might be triggered if traffic actually increased on Friday evening.
- 3-Sigma** The 3-sigma algorithm classifies a new data point as normal if it's within 3 standard deviations from the mean (average across all the data points in the data set). A new data point is classified as abnormal if it's outside this range.
- K-means** The Paragon Insights k-means algorithm uses k-means clustering and other building blocks to create a machine learning model for classifying new data points as normal or abnormal:
- K-means clustering splits n data points into k groups (called clusters), where $k \leq n$. For Paragon Insights, k is set to 5 buckets.
 - For forming the clusters, a 32-dimensional vector is considered for each point, thus taking into account the trend (not just the current point, but its past 31 historical values).
 - Each cluster has a center called the centroid. A cluster centroid is the mean of a cluster (average across all the data points in the cluster).
 - All new data points are added to a cluster, however, if a data point is considered too far away from the centroid, it is classified as abnormal.

Learning period

The learning period specifies the time range to collect data from which the algorithm uses to build the machine learning models. Supported units of time for learning period include: seconds, minutes, hours, days, weeks, and years. You must enter the plural form of the time unit with no space between the number and the unit. For example, a learning period of one hour must be entered as 1hours.

Paragon Insights builds machine learning models daily starting at midnight. For example, if the learning period is 5 days and triggered on 11th Feb 2019 00:00, then data collected from 6th Feb 2019 00:00 to 11th Feb 2019 00:00 is used by Paragon Insights to build the machine learning models. For the Holt-Winters prediction algorithm, the learning period must be at least twice the pattern periodicity to ensure there is enough of a pattern to learn.

Pattern periodicity

The pattern periodicity specifies the buckets for which data should be collected and used to build machine learning models. Each bucket of data represents a user-defined time period and a specific pattern of data. A minimum number of data points is required for a machine learning algorithm to build a model:

- 3-sigma requires at least 10 data points per bucket of data.
- K-means requires at least 32 data points per bucket of data.

Supported units of time for pattern periodicity include: minutes, hours, days, weeks, and months. You must enter the plural form of the time unit with no space between the number and the unit.

For example:

- If the pattern periodicity is 1 day (entered as 1days), the data for each day of the week has a specific pattern. Paragon Insights creates 7 buckets of data and 7 different models, one for each day of the week.
- If the pattern periodicity is 1 hour (entered as 1hours), regardless of the day, week, or month, the data for every hour has a specific pattern. Paragon Insights creates 24 buckets of data and 24 different models, one for each hour (00:00-00:59, 1:00-1:59, 2:00-2:59 ... 23:00-23:59) of the day.
- If the pattern periodicity is 1 day 1 hour (entered as 1days 1hours), the data for every hour of each day of the week has a specific pattern. Paragon Insights creates $7 * 24 = 168$ buckets of data and 168 different models. 24 buckets for Monday (1 for every hour), 24 buckets for Tuesday (1 for every hour), and so on. In this case, it doesn't matter from which month data is collected.

Understanding Paragon Insights Outlier Detection

IN THIS SECTION

- Dataset | 69
- Algorithm | 70
- Sigma coefficient (k-fold-3sigma only) | 71
- Sensitivity | 71
- Learning period | 71

This section describes the input parameters associated with Paragon Insights rules used for outlier detection algorithms. Once the machine learning models are built, they can be used in production to identify time series data sets as outliers. The accuracy of the results increases with larger amounts of data.

The results of the Paragon Insights outlier detection algorithm are stored in a table in the times series database. Each row in the table contains outlier detection output and metadata that is associated with a particular time series. You can use the information in the table to configure Paragon Insights rule triggers. Each column in the table is identified by a unique name that starts with the user-defined outlier detection field name. For example, you can use the *field-name-is-outlier* and *field-name-device* column names to configure a trigger that detects outliers and produces a message that indicates which specific device was determined to be the outlier. For more information, see the “Triggers” section of the ["Paragon Insights Outlier Detection Example" on page 83](#).

Dataset

For the outlier detection formula, input data is specified as a list of XPATHs from a variable. For information on how to create a user-defined variable for a Paragon Insights rule, see the [Variables](#) section in the Paragon Insights User Guide.

The following is an example of a list of XPATHs:

```
/device-group[device-group-name=DG0]/device[device-name=D0]/topic[topic-name=T0]/ rule[rule-name=R0]/field[re=RE[01] AND hostname=10.1.1.*]/re-memory,/ device-group[device-group-name=DG0]/device[device-name=D1]/topic[topic-name=T0]/ rule[rule-name=R0]/field[re=RE[01] AND hostname=10.1.1.*]/re-memory
```

This path list specifies that on devices D0 and D1 in device-group DG0, get re-memory from topic T0 rule R0, where the RE is either RE0 or RE1 and the hostname is in the 10.1.1.* block. This path allows for selecting data at the field-key level, which is necessary because different field keys may have different purposes.

For example:

- On D0 and D1, with the field named “memory usage on routing engine,” keys RE0 and RE1 represent two routing engines per device.
- There’s no guarantee that RE0 is a primary on all devices, therefore they might not be comparable when checking for outliers.
- This mechanism allows for selecting only the primaries: D0-RE0 and D1-RE1.

Algorithm

The outlier detection algorithms include k-fold, 3-sigma, and dbscan:

K-Fold Cross-Validation Using 3-Sigma

K-fold cross-validation using the 3-sigma (k-fold 3-sigma) algorithm is used to create a machine learning model for identifying outliers. K-fold cross-validation splits the entire data set into k groups and uses the following general process to create the machine learning models:

- Each unique k group is used as a test data set.
- The k groups (that are not being used as a test data set) are used as the training data set to build a machine learning model.
- Each test data set is evaluated using its associated machine learning model.
- The test data set with the most outliers relative to their associated machine learning model is classified as an outlier.

For example, if k is the number of devices in a device group and the group has 4 devices, then $k=4$. For cross-validation, four machine learning models are built and the test data sets are evaluated as follows:

- Model 1: Trained with the data points collected from device 1, device 2, and device 3, then tested with the data points collected from device 4.
- Model 2: Trained with the data points collected from device 1, device 2, and device 4, then tested with the data points collected from device 3.
- Model 3: Trained with the data points collected from device 1, device 3, and device 4, then tested with the data points collected from device 2.
- Model 4: Trained with the data points collected from device 2, device 3, and device 4, then tested with the data points collected from device 1.

Using the k-fold 3-sigma algorithm is more applicable if it's known that outliers will skew in one direction or another. If there are outliers on both sides of normal data points, or there are enough outlier data points to make the algorithm believe that nothing is outlying, then the k-fold 3-sigma algorithm will not provide significant results.

DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is an unsupervised machine learning algorithm used to create a machine learning model for identifying time series data sets as outliers:

- Time series data sets are grouped in such a way that data points in the same cluster are more similar to each other than those in other clusters.
- Clusters are dense regions of time series data sets, separated by regions of lower density.
- If a time series data set belongs to a cluster, it should be near many other time series data sets in that cluster.
- Time series data sets in lower density regions are classified as outliers.

Using the DBSCAN algorithm is more applicable if outliers appear inside the 3-sigma threshold of the other data points. DBSCAN can find outlying behavior that doesn't appear as a significant deviation from the normal behavior at any given time step.

Sigma coefficient (k-fold-3sigma only)

The sigma coefficient is a thresholding argument (default value is 3). The thresholding argument determines, at each point in time for a series, how far away a value must be from the other values to be marked as an outlier.

Sensitivity

Sensitivity is used to calculate the outliers, m , that the algorithm seeks to find in the data. Sensitivity determines the number of time series test data sets to return as outliers (the top m are returned):

- Sensitivity "low": 0.03% of the number of sensors
- Sensitivity "medium": 5% of the number of sensors
- Sensitivity "high": 36% of the number of sensors
- Absolute percentage x : $x \times \text{number of sensors}$ (float, 0.0-1.0)

Learning period

See the "[Learning period](#)" on [page 67](#) description of the "Understanding Paragon Insights Anomaly Detection" section.

Understanding Paragon Insights Predict

IN THIS SECTION

- [Field | 72](#)
- [Algorithm | 72](#)
- [Learning period | 72](#)
- [Pattern periodicity | 73](#)
- [Prediction offset | 73](#)

This section describes the input parameters associated with Paragon Insights rules used for forecasting future values with the Paragon Insights median prediction machine learning algorithm or the Holt-Winters prediction machine learning algorithm. Once the machine learning models are built, they can be used in production to predict trends and forecast future values. The accuracy of the results increases with larger amounts of data.

Field

See the ["Field" on page 66](#) description of the "Understanding Paragon Insights Anomaly Detection" section.

Algorithm

The Paragon Insights Predict feature uses either the median prediction algorithm, or the Holt-Winters prediction algorithm.

The median value represents the midpoint for a range of values within a data sampling. For every pattern periodicity bucket, a median is calculated from the data samples available in the bucket.

Learning period

See the ["Learning period" on page 67](#) description of the "Understanding Paragon Insights Anomaly Detection" section.

Pattern periodicity

See the ["Pattern periodicity" on page 68](#) description of the "Understanding Paragon Insights Anomaly Detection" section. For the median prediction algorithm, we recommend a minimum number of 10 data points for building a machine learning model. For the Holt-Winters algorithm, the pattern periodicity should be half or less of the learning period.

Prediction offset

The prediction offset value is a time in the future at which you want to predict a field value. For example, if the present time is 6th Feb 2019 10:00 and the prediction offset is set to 5 hours, then Paragon Insights will predict a field value for 6th Feb 2019 15:00.

Supported units of time for prediction offset include: seconds, minutes, hours, days, weeks, and years. You must enter the plural form of the time unit with no space between the number and the unit. For example, a prediction offset of one hour must be entered as 1hours.

Paragon Insights Rule Examples

IN THIS SECTION

- [Paragon Insights Anomaly Detection Example | 73](#)
- [Paragon Insights Outlier Detection Example | 83](#)

The machine learning Paragon Insights rules described in this section are available for upload from the [Paragon Insights Rules and Playbooks](#) GitHub repository.

Paragon Insights Anomaly Detection Example

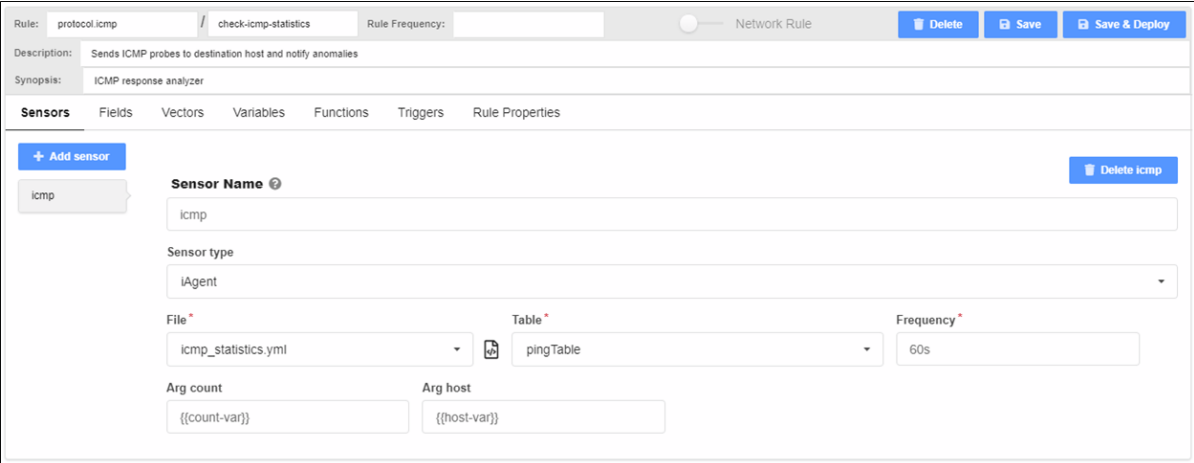
This example describes how the `check-icmp-statistics` Paragon Insights device rule is configured to send ICMP probes to user-defined destination hosts to detect anomalies when round trip average response time is above static or dynamic thresholds.

The following sections show how to configure the applicable input parameters for each Paragon Insights rule definition block (such as, Fields, Variables, and Triggers) using the Paragon Insights GUI. For more information about how to configure Paragon Insights rules, see [Creating a New Rule using the Paragon Insights GUI](#).

Sensors (check-icmp-statistics)

Figure 8 on page 74 shows the general properties and iAgent sensor configured for the check-icmp-statistics rule. For information about the count-var and host-var variables, see "Variables (check-icmp-statistics)" on page 76.

Figure 8: General properties (check-icmp-statistics) and Sensors definition (icmp)



Fields (check-icmp-statistics)

The following fields are configured for the check-icmp-statistics rule:

- dt-response-time** (See [Figure 9 on page 75](#)) Configuration for anomaly detection using the k-means algorithm. When an anomaly is detected, Paragon Insights returns a value of 1.
- rtt-average-ms** (See [Figure 10 on page 75](#)) Round trip average response time.
- rtt-threshold** (See [Figure 11 on page 76](#)) Static threshold for the round trip average response time. The rtt-threshold variable populates this field.

Figure 9: Fields definition (dt-response-time)

Sensors **Fields** Vectors Variables Functions Triggers Rule Properties

+ ADD FIELD

dt-response-time
host
packet-loss-percent
rtt
rtt-average-ms
rtt-threshold

Field Name * ?
dt-response-time

Description ?
Dynamic threshold for field rtt-average-ms

Field Type
integer

☐ **Add to Rule Key** ?

Ingest type (Field source)
Formula

Formula **Field *** **Algorithm *** **Learning period ***

Anomaly Detection \$rtt-average-ms k-means 7d

Pattern periodicity *
1h

DELETE DT-RESPONSE-TIME

Figure 10: Fields definition (rtt-average-ms)

Sensors **Fields** Vectors Variables Functions Triggers Rule Properties

+ Add field

dt-response-time
rtt-average-ms
rtt-threshold

Field name * ?
rtt-average-ms

Description ?
Round trip time

Field type
float

☐ **Add to rule key**

Ingest type (Field source)
Formula

Formula **Argument micros ***

micro-milli \$rtt-average

Delete rtt-average-ms

5085205

Figure 11: Fields definition (rtt-threshold)

SensorsFieldsVectorsVariablesFunctionsTriggersRule Properties

+ Add field

dt-response-time

rtt-average-ms

rtt-threshold

Field name* ?

rtt-threshold

Description ?

RTT response time static threshold

Field type

string

☐ Add to rule key

Ingest type (Field source)

Constant

Constant value *

{{rtt-threshold-var}}

Delete rtt-threshold

Variables (check-icmp-statistics)

The following three variables are configured for the check-icmp-statistics rule:

- count-var

(See [Figure 12 on page 76](#)) ICMP ping count. Count is set to 1 by default.
- host-var

(See [Figure 13 on page 77](#)) Destination IP address or host name where ICMP probes are periodically sent.
- rtt-threshold-var

(See [Figure 14 on page 77](#)) Static threshold value for round trip average response time. Threshold value is 1 ms by default. This variable populates the rtt-threshold field.

Figure 12: Variables definition (count-var)

SensorsFieldsVectorsVariablesFunctionsTriggersRule Properties

+ Add variable

count-var

host-var

rtt-threshold-var

Variable name* ?

count-var

Default value ?

1

Type* ?

Sensor Argument

Description ?

Input ICMP probe count

Delete count-var

Figure 13: Variables definition (host-var)

The screenshot shows the 'Variables' tab in a configuration tool. On the left, a list of variables includes 'count-var', 'host-var' (selected), and 'rtt-threshold-var'. The main area displays the configuration for 'host-var':

- Variable name***: host-var
- Default value***: Enter default value
- Type***: Sensor Argument
- Description**: Input Destination host

Buttons for '+ Add variable' and 'Delete host-var' are visible.

Figure 14: Variables definition (rtt-threshold-var)

The screenshot shows the 'Variables' tab in a configuration tool. On the left, a list of variables includes 'count-var', 'host-var', and 'rtt-threshold-var' (selected). The main area displays the configuration for 'rtt-threshold-var':

- Variable name***: rtt-threshold-var
- Default value***: 1
- Type***: Integer
- Description**: Input RTT response static threshold in milli seconds

Buttons for '+ Add variable' and 'Delete rtt-threshold-var' are visible.

Functions (check-icmp-statistics)

Figure 15 on page 77 shows the function configured for the check-icmp-statistics rule. This function converts the unit of measure for the round trip average response time from microseconds to milliseconds.

Figure 15: Functions definition (micro-milli)

The screenshot shows the 'Functions' tab in a configuration tool. On the left, a list of functions includes 'micro-milli' (selected). The main area displays the configuration for 'micro-milli':

- Function name***: micro-milli
- Path to function***: micro_milli.py
- Method name***: rtt_micro_milli
- Description**: This function converts microseconds to milliseconds
- Arguments**:
 - Name**: micros
 - Mandatory**: ☒

Buttons for '+ Add function', 'Delete micro-milli', and '+ Add Argument' are visible.

Triggers (check-icmp-statistics)

The following triggers and terms are configured for the check-icmp-statistics rule:

- packet-loss — (See [Figure 16 on page 79](#))

The following terms are configured for the packet-loss trigger:

is-device-not-reachable	(See Figure 17 on page 79) When the ICMP packet loss is 100%, the Paragon Insights health status severity level is set to major (red).
is-device-up	(See Figure 18 on page 80) When the packet loss is greater than 0, the severity level is set to minor (yellow).
no-packet-loss	(See Figure 19 on page 80) Otherwise, the severity level is set to normal (green).

- round-trip-time — (See [Figure 20 on page 81](#))

The following terms are configured for the round-trip-time trigger:

is-rtt-fine	(See Figure 21 on page 81) When the host is not reachable or the round trip average response time is above the static threshold, the Paragon Insights health status severity level is set to major (red).
is-rtt-medium	(See Figure 22 on page 82) When an anomaly is detected using the anomaly detection formula, Paragon Insights returns a value of 1 for the dt-response-time field, and the severity level is set to minor (yellow). In this case, the response time is above the anomaly detection.
rtt-normal	(See Figure 23 on page 82) Otherwise, the severity level is set to normal (green).

Figure 16: Triggers definition (packet-loss)

The screenshot shows the 'Triggers' tab in a configuration interface. On the left, a sidebar lists 'packet-loss' and 'round-trip-time'. The main area is for defining the 'packet-loss' trigger. It includes a 'Trigger name' field with the value 'packet-loss' and a 'Delete packet-loss' button. Below this is a 'Frequency' field set to '60s'. A list of terms is shown, each with a 'Term' field and a delete icon: 'is-device-not-reachable', 'is-device-up', and 'no-packet-loss'. An '+ Add Term' button is at the bottom left. The top navigation bar includes 'Sensors', 'Fields', 'Vectors', 'Variables', 'Functions', 'Triggers', and 'Rule Properties'. A vertical label '5008212' is on the right side.

Figure 17: Terms definition (is-device-not-reachable)

The screenshot shows the 'Terms' definition interface for the 'is-device-not-reachable' term. The top bar shows the term name and a close button. The 'WHEN' section contains a condition: 'Left operand' is '\$packet-loss', 'Operator' is '==', 'Right operand' is '100', and 'Time range' is '2m'. There is a red minus icon next to the time range. Below this is an '+ Add Condition' button. The 'THEN' section has a 'Color' dropdown set to red, a 'Message' text area containing 'Host \$host not reachable', and an 'Evaluate next term' toggle switch. An '+ Add Function' button is at the bottom left. A vertical label '5008213' is on the right side.

Figure 18: Terms definition (is-device-up)

The screenshot shows a web interface for defining a term named 'is-device-up'. The interface is divided into two main sections: 'WHEN' and 'THEN'. In the 'WHEN' section, there are four input fields: 'Left operand' with a dropdown menu showing '\$packet-loss', 'Operator' with a dropdown menu showing '>', 'Right operand' with a dropdown menu showing '0', and 'Time range' with a text input showing '2m'. Below these fields is a blue button labeled '+ Add Condition'. In the 'THEN' section, there is a 'Color' dropdown menu showing a yellow color swatch, a 'Message' text area containing the text 'Host \$host reachable and \$packet-loss % packet loss', and a toggle switch labeled 'Evaluate next term' which is currently turned off. Below the toggle is a blue button labeled '+ Add Function'. The interface has a light gray header bar with a dropdown menu showing 'Term' and a close button 'X'.

Term: is-device-up

WHEN

Left operand: \$packet-loss Operator: > Right operand: 0 Time range: 2m

+ Add Condition

THEN

Color: [Yellow swatch]

Message: Host \$host reachable and \$packet-loss % packet loss

☐ Evaluate next term

+ Add Function

Figure 19: Terms definition (no-packet-loss)

The screenshot shows a web interface for defining a term named 'no-packet-loss'. The interface is divided into two main sections: 'WHEN' and 'THEN'. In the 'WHEN' section, there is a blue button labeled '+ Add Condition'. In the 'THEN' section, there is a 'Color' dropdown menu showing a green color swatch, a 'Message' text area containing the text 'Host \$host reachable and \$packet-loss % packet loss', and a toggle switch labeled 'Evaluate next term' which is currently turned off. Below the toggle is a blue button labeled '+ Add Function'. The interface has a light gray header bar with a dropdown menu showing 'Term' and a close button 'X'.

Term: no-packet-loss

WHEN

+ Add Condition

THEN

Color: [Green swatch]

Message: Host \$host reachable and \$packet-loss % packet loss

☐ Evaluate next term

+ Add Function

Figure 20: Triggers definition (round-trip-time)

The screenshot shows the 'Triggers' tab in a configuration tool. On the left, a sidebar lists 'packet-loss' and 'round-trip-time', with 'round-trip-time' selected. The main area displays the configuration for 'round-trip-time'. At the top right is a 'Delete round-trip-time' button. The configuration includes a 'Trigger name' field with the value 'round-trip-time', a 'Frequency' field with the value '60s', and a list of terms. The terms list contains three entries: 'is-rtt-fine', 'is-rtt-medium', and 'rtt-normal'. Each entry has a right-click menu icon (X). At the bottom of the terms list is an '+ Add Term' button. The top navigation bar includes tabs for 'Sensors', 'Fields', 'Vectors', 'Variables', 'Functions', 'Triggers', and 'Rule Properties'.

Figure 21: Terms definition (is-rtt-fine)

The screenshot shows the 'Terms' configuration window for the term 'is-rtt-fine'. The window has a title bar with a dropdown for 'Term' set to 'is-rtt-fine' and a close button (X). The main content is divided into two sections: 'WHEN' and 'THEN'. The 'WHEN' section contains four fields: 'Left operand' (set to '\$rtt-average-ms'), 'Operator' (set to '>='), 'Right operand' (set to '\$rtt-threshold'), and 'Time range' (with a placeholder 'Enter a time range' and a red minus icon). Below these fields is an '+ Add Condition' button. The 'THEN' section contains a 'Color' field (set to red) and a 'Message' field (containing the text 'Round trip time(\$rtt-average-ms ms) to \$host is above static threshold'). Below the message field is a toggle switch for 'Evaluate next term' (currently off) and an '+ Add Function' button. The right side of the image has a vertical label 's008217'.

Figure 22: Terms definition (is-rtt-medium)

Term: is-rtt-medium

WHEN

Left operand	Operator	Right operand	Time range
\$dt-response-time	==	1	Enter a time range

+ Add Condition

THEN

Color:

Message: Round trip time(\$rtt-average-ms ms) to \$host is above dynamic threshold

☐ Evaluate next term

+ Add Function

5008218

Figure 23: Terms definition (rtt-normal)

Term: rtt-normal

WHEN

+ Add Condition

THEN

Color:

Message: Round trip time(\$rtt-average-ms ms) to \$host is normal

☐ Evaluate next term

+ Add Function

5008219

Rule Properties (check-icmp-statistics)

Figure 24 on page 83 shows the rule properties configured for the check-icmp-statistics rule.

Figure 24: Rule Properties definition (check-icmp-statistics)

The screenshot shows the 'Rule Properties' tab for a rule named 'check-icmp-statistics'. The interface includes several input fields and a tree view for device support.

- version:** 1
- Contributor:** juniper (with a dropdown arrow)
- author-email:** Enter author-email
- Date:** (empty field)
- supported-healthbot-version:** 2.0.0
- Supported Device:** A tree view showing 'Juniper Devices' expanded, with 'Junos' selected. Under 'Junos', 'Product Name' is set to 'MX'. Below this, 'Release Name' is '11.4', 'Release Support' is 'min-supported-releases' (with a dropdown arrow), and 'Platform' is 'All'.

On the right side of the form, there is a vertical label 's008220'.

Paragon Insights Outlier Detection Example

This example describes how the check-outlier Paragon Insights network rule is configured to detect outliers across the devices in a device group using the round trip time average response time.

The following sections show how to configure the applicable input parameters for each Paragon Insights rule definition block (such as, Fields, Variables, and Triggers) using the Paragon Insights GUI. For more information about how to configure a Paragon Insights rule, see [Creating a New Rule using the Paragon Insights GUI](#).

Sensors (check-outlier)

Figure 25 on page 83 shows the general properties configured for the check-outlier rule. Note that this rule is a network rule.

Figure 25: General properties (check-outlier)

The screenshot shows the 'General' tab for a rule named 'check-outlier'. The rule is identified as 'protocol icmp' and 'check-outlier' with a frequency of '60s'. It is marked as a 'Network Rule'.

- Rule:** protocol icmp / check-outlier
- Rule Frequency:** 60s
- Network Rule:** (checked)
- Description:** Detects ICMP outlier response time and notify anomalies
- Synopsis:** ICMP outlier analyzer
- Sensors:** A section with a '+ Add sensor' button and a message: 'No sensor objects added yet, click 'Add sensor' to add one.'

On the right side of the form, there is a vertical label 's008221'.

Fields (check-outlier)

Figure 26 on page 84 shows the field configured for the check-outlier rule. This field defines the DBSCAN algorithm and rtt-xpath variable for outlier detection. For information about the rtt-xpath variable, see "Variables (check-outlier)" on page 84.

The results of the Paragon Insights outlier detection algorithm are stored in a table in the times series database. Each row in the table contains outlier detection output and metadata that is associated with a particular time series. You can use the information in the table to configure Paragon Insights rule triggers. Each column in the table is identified by a unique name that starts with the user-defined outlier detection field name. For example, you can use the *field-name-is-outlier* (rtt-ol-is-outlier) and *field-name-device* (rtt-ol-device) column names to configure a trigger that detects outliers and produces a message that indicates which specific device was determined to be the outlier (see "Triggers (check-outlier)" on page 85.).

Figure 26: Fields definition (rtt-ol)

The screenshot shows the 'Fields' configuration page in Paragon Insights. The 'rtt-ol' field is selected in the left sidebar. The main configuration area includes the following fields:

- Field name:** rtt-ol
- Description:** Outlier detection of devices in a group using xpath
- Field type:** string
- Add to rule key:** (toggle switch, currently off)
- Ingest type (Field source):** Formula
- Formula:** Outlier Detection
- Dataset:** {{rtt-xpath}}
- Algorithm:** dbscan
- Sensitivity:** low
- Learning Period:** 30m

Buttons for '+ Add field' and 'Delete rtt-ol' are visible at the top. The bottom right corner of the interface shows the identifier '3006222'.

Variables (check-outlier)

Figure 27 on page 85 shows the variable configured for the check-outlier rule. This variable defines the devices in the network from which Paragon Insights collects round trip average response time data for the outlier detection machine learning models.

Figure 27: Variables definition (rtt-xpath)

SensorsFieldsVectorsVariablesFunctionsTriggersRule Properties

+ Add variable

rtt-xpath

Variable name*

rtt-xpath

Default value

*

Type*

String

Delete rtt-xpath

Description

Input xpath of devices to be in outlier detection format e.g. /device-group[device-group-name=core]/device[device-id=R0]/topic[topic-name=protocol.icmp]/rule[rule-name=check-icmp-statistics]/rtt-average-ms,/device-group[device-group-name=core]/device[device-id=R1]/topic[topic-name=protocol.icmp]/rule[rule-name=check-icmp-statistics]/rtt-average-ms,/device-group[device-group-name=core]/device[device-id=R2]/topic[topic-name=protocol.icmp]/rule[rule-name=check-icmp-statistics]/rtt-average-ms

Triggers (check-outlier)

Figure 28 on page 85 shows the trigger configured for the check-outlier rule. The following terms are configured for the icmp-outlier-detection trigger:

- is-outlier-detected

(see Figure 29 on page 86) When an outlier is detected, Paragon Insights returns a value of 1 for the rtt-ol-is-outlier field, and the Paragon Insights health status severity level is set to major (red). This term also produces a message that indicates which specific device was determined to be the outlier.
- no-outlier

(See Figure 30 on page 86) Otherwise, Paragon Insights returns a value of 0, and the severity level is set to normal (green).

Figure 28: Triggers definition (icmp-outlier-detection)

SensorsFieldsVectorsVariablesFunctionsTriggersRule Properties

+ Add trigger

icmp-outlier-detection

Delete icmp-outlier-detection

Trigger name

icmp-outlier-detection

Frequency

60s

Term

is-outlier-detected

Term

no-outlier

+ Add Term

s008223

s008224

Figure 29: Terms definition (is-outlier-detected)

The screenshot shows a configuration window for a term named 'is-outlier-detected'. It is divided into two main sections: 'WHEN' and 'THEN'.

WHEN Section:

- Left operand:** A dropdown menu with '\$rtt-ol-is-outlier' selected.
- Operator:** A dropdown menu with '==' selected.
- Right operand:** A dropdown menu with '1' selected.
- Time range:** A text input field with the placeholder 'Enter a time range' and a red minus icon to its right.
- + Add Condition:** A blue button with a plus icon.

THEN Section:

- Color:** A dropdown menu with a red color swatch selected.
- Message:** A text area containing the text 'Outlier detected on \$rtt-ol-device'.
- Evaluate next term:** A toggle switch that is currently turned off.

Footer: A light blue box with an information icon and the text: 'Functions can be used as Trigger actions too, define them using the 'Functions' menu at the top.'

5008225

Figure 30: Terms definition (no-outlier)

The screenshot shows a configuration window for a term named 'no-outlier'. It is divided into two main sections: 'WHEN' and 'THEN'.

WHEN Section:

- Left operand:** A dropdown menu with '\$rtt-ol-is-outlier' selected.
- Operator:** A dropdown menu with '==' selected.
- Right operand:** A dropdown menu with '0' selected.
- Time range:** A text input field with the placeholder 'Enter a time range' and a red minus icon to its right.
- + Add Condition:** A blue button with a plus icon.

THEN Section:

- Color:** A dropdown menu with a green color swatch selected.
- Message:** A text area containing the text 'No outlier detected on \$rtt-ol-device'.
- Evaluate next term:** A toggle switch that is currently turned off.

Footer: A light blue box with an information icon and the text: 'Functions can be used as Trigger actions too, define them using the 'Functions' menu at the top.'

5008103

Rule Properties (check-outlier)

Figure 31 on page 87 shows the rule properties configured for the check-outlier rule.

Figure 31: Rule Properties definition (check-outlier)

SensorsFieldsVectorsVariablesFunctionsTriggersRule Properties

versionContributorauthor-emailDateupported-healthbot-version

1juniperEnter author-email2.0.0

Supported Device

Juniper Devices

JunosProduct NameMXRelease Name11.4Release Supportmin-supported-releasesPlatformAll

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
3.1.0	Starting with HealthBot Release 3.1.0, you can choose the Holt-Winters prediction algorithm

Frequency Profiles and Offset Time

IN THIS SECTION

- Frequency Profiles | 87
- Offset Time Unit | 94

Frequency Profiles

IN THIS SECTION

- Configuration Using Paragon Insights GUI | 89

- [Configuration Using Paragon Insights CLI | 92](#)
- [Apply a Frequency Profile Using the Paragon Insights GUI | 92](#)
- [Apply a Frequency Profile Using the Paragon Insights CLI | 94](#)

Frequency profiles are a central location in which sensor and rule time frequencies can be managed. To understand frequency profiles, consider the following.

When defining rules in Paragon Insights (formerly HealthBot) you can:

- Define multiple rules that use the same sensor
- Define different sensor frequencies for each of the rules
- Apply all of these rules to the same device group/devices

This creates complexity in rule application and frequency adjustments within the individual rules:

- A key, consisting of sensor-path for OpenConfig and Native GPB sensors, or the tuple of file and table for iAgent sensors is used to identify the specific rules.
- Paragon Insights takes the minimum defined frequency for that sensor from the applied rules and uses it to subscribe to, or fetch, data from the devices.
- This make it hard to identify what the data rate should be for that sensor. To do that, you would have to go through the all the applied rules.
- A change in the sensor frequency of an applied rule might not take effect as intended.

To address these complexities, Paragon Insights needed a common place from which to control these frequencies.

Starting in HealthBot Release 3.0.0, frequency profiles can be created that allow you to manage sensor and rule frequencies from a single location and then apply the profiles in various locations in Paragon Insights. Application of these profiles allows for persistent and repeatable behavior in regard to frequencies for rules, sensors, triggers, formulas, references, learning periods, and hold times.

A sensor profile consists of a profile name and two optional sections: the sensors section and the non-sensors section. In each section, an entry consists of a sensor or rule name and a frequency. Frequency profiles are applied to device groups or network groups.

The steps for configuration are shown below.

Configuration Using Paragon Insights GUI

Frequency profiles are configured and managed in the Paragon Insights GUI or in the Paragon Insights CLI. In the GUI, they are managed by navigating to the **Settings > Ingest Settings** page and selecting the **Frequency Profile** tab from the left side of the page.

NOTE: While the sections of the frequency profile are both optional, at least one section must be filled out per frequency profile if you want the applied profile to be able to do anything.

Add a Frequency Profile

1. Click the **add (+)** icon to add a profile.

The Add Frequency Profile window appears.

2. Give the profile a name such as Profile1
3. Click the **add (+)** icon to add sensors.
4. In the **Sensor Name** field, enter the sensor name as per the following guidelines:
 - *OpenConfig Sensors:* Enter the OpenConfig path for the desired sensor, such as /components or / interfaces.
 - *iAgent Sensors:* Enter the table name used in the sensor definition, such as ChassisAlarmTable or REutilizationTable
 - *SNMP:* Enter the sensor name such as npr_qmon_ext
 - *BYOT:* Enter <topic-name/ rule-name/ sensor-name>, such as topic1/rule1/sensor1
5. In the **Frequency** field, enter the appropriate frequency, such as 30seconds, 1minute, 2hours, and so on.
6. (Optional) Click the **add (+)** icon to add non-sensors.
7. In the **Rule Name** field, enter the rule name such as check-chassis-alarms.
8. In the **Frequency** field, enter the appropriate frequency, such as 45seconds, 3minutes, 1hour, and so on.

Repeat steps 3 through 5 or 6 through 8 as desired for the profile.

9. Click the **SAVE & DEPLOY** button to save and deploy the profile.

The new sensor profile is added to the list.

Edit a Frequency Profile

To edit an existing frequency profile:

1. Click the **Frequency Profile** tab to view the Frequency Profile page.
2. Select the < *Profile Name* > that you want to modify.
3. Click the **edit (pencil)** icon to edit the profile.

The Edit Frequency Profile page appears, displaying the same fields that are presented when you add a frequency profile.

4. Modify the parameters.
5. Click the **SAVE** button to save the profile for later deployment or the **SAVE & DEPLOY** button to save and deploy immediately.

Delete a Frequency Profile

To delete a frequency profile:

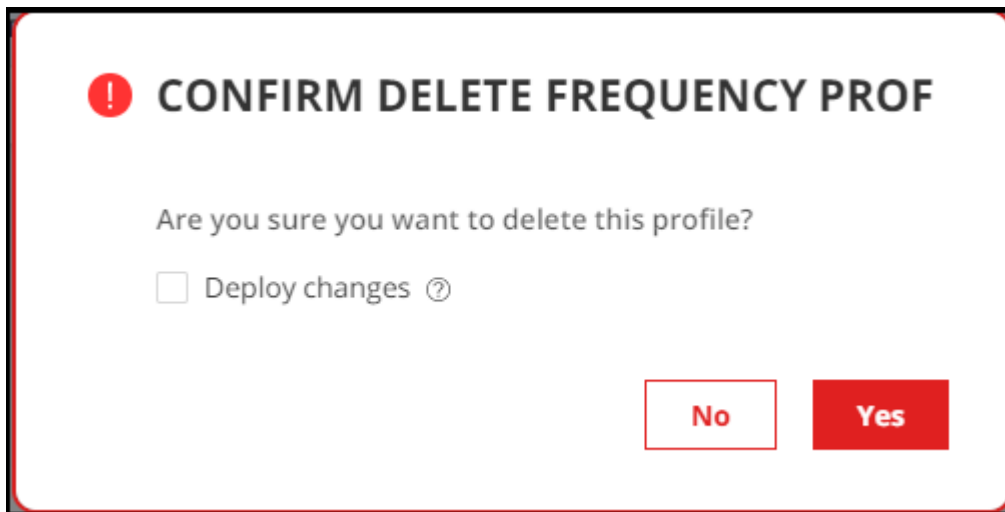
1. Click **Settings** > **Ingest** from the left-nav bar.

The **Ingest Settings** page is displayed.

2. Click the **Frequency Profile** tab to view the **Frequency Profiles** page.
3. Select the frequency profile that you want to delete, and click the **delete (trash can)** icon.

The **CONFIRM DELETE FREQUENCY PROFILE** pop-up appears.

Figure 32: Confirm Delete Frequency Profile Pop-up



4. Do any one of the following:

- Click **Yes** to delete the frequency profile from the database. However, the changes are not applied to the ingest service.

NOTE:

- We recommended that you do not delete a frequency profile that is currently in use.
- After you delete a frequency profile from the database, you cannot apply that frequency profile to another device group even if you have not deployed changes.
- You can also deploy changes to the ingest service or roll back the changes that you have already deleted, from the **PENDING CONFIGURATION** page. For more information, see ["Commit or Roll Back Configuration Changes in Paragon Insights" on page 307](#).

- Select the **Deploy changes** check box and then click **Yes** to delete the frequency profile from the database, and to apply the changes to the ingest service.
- (Optional) Click **No** to cancel this operation.

Clone a Frequency Profile

To clone a frequency profile:

1. Click the **Frequency Profile** tab to view the Frequency Profile page
2. Select the < *Profile Name* > that you want to clone and then click the **Clone** button at the top-right corner of the page.

The Clone Frequency Profile page appears.

3. Specify an appropriate name for the cloned profile.
4. Click **OK** to save your changes.

A clone of the profile is created and listed on the Frequency profile page.

Usage Notes:

- *Profile Entries*—Multiple entries can be configured in each section.
- *Override*—Sensor or rule frequency defined within an applied frequency profile overrides those defined within the individual rule or sensor.
- *Order of Precedence*—If a sensor or rule is defined in multiple frequency profiles, each with different frequency settings, the minimum frequency value for the sensor or rule is used.

Configuration Using Paragon Insights CLI

In the Paragon Insights CLI, you can configure the same frequency profile described above. An example of the CLI configuration needed to complete the example above looks like:

```
user@mgd-69ab987fbc6-pt9sh> show configuration healthbot ingest-settings frequency-profile
Profile1
sensor /components {
    frequency 60seconds;
}
sensor /interface {
    frequency 30seconds;
}
non-sensor net-topic/net-rule {
    frequency 2minutes;
}
```

Apply a Frequency Profile Using the Paragon Insights GUI

Frequency profiles are applied to Paragon Insights device groups or network groups. When you create or edit a device or network group, you apply frequency profiles by selecting them from the **Ingest Frequency** section of the **Device Group** definition. [Figure 33 on page 93](#) below shows an example of frequency profiles being applied to a device group.

Figure 33: Apply Frequency Profiles

Edit "ix-lab-group"

Description

Describe the device group.

Devices*

spice

Select devices to add to this group.

Timezone

+/-00:00

Specify the Timezone in the format +/-hh:mm.

Retention Policy

Select the name of the retention policy to be applied.

Native Ports

Comma separated values

Specify the native sensors receiver port(s).

Flow Ports

Comma separated values

Specify the netflow sensor receiver port(s).

Syslog Ports

Comma separated values

Specify the syslog messages receiver UDP port(s).

Flow Deploy Nodes

Select flow ingest deploy nodes for this device group.

Reports

Select reports to generate for this device group.

Summarization

Ingest Frequency

Ingest Frequency Profiles

Profile1 Profile2

All profiles...

NewProfile

Notifications

Once you have applied the needed profiles, save and deploy the device group using the **SAVE & DEPLOY** button.

BEST PRACTICE: It is strongly recommended that you only apply frequency profiles to rules that make use of the [Offset Time Unit](#) feature.

Apply a Frequency Profile Using the Paragon Insights CLI

An example of a device group CLI configuration which includes a frequency profile and could be deployed in Paragon Insights is shown below.

```
user@mgd-69ab987fbc6-pt9sh> show configuration healthbot device-group lab-group
devices router1;
ingest-frequency Profile1;
```

Offset Time Unit

IN THIS SECTION

- [Offset Used in Formulas | 95](#)
- [Offset Used in References | 97](#)
- [Offset Used in Vectors | 98](#)
- [Offset Used in Triggers | 100](#)
- [Offset Used in Trigger Reference | 101](#)

The Paragon Insights (formerly HealthBot) offset time unit is used in conjunction with the [Frequency Profiles](#) to automatically manage time range calculations in various places within Paragon Insights rules. To understand the Paragon Insights offset function, consider the following scenario:

In Paragon Insights, you can define a rule which

- Uses a sensor to gather data with a frequency of 10 seconds
- Has a field that calculates the mean every 60 seconds

If you later decide to increase the frequency of the sensor to 60 seconds, then calculating the mean every 60 seconds would not make any sense. The result is that you would have to manually update the field calculation any time up want to change the sensor frequency.

Starting with HealthBot Release 3.0.0, you can set the time range for the mean calculation to a value of 60, or 60offset, rather than 60s, or 60 seconds. Using an offset time value rather than a static time value tells Paragon Insights to automatically multiply the sensor frequency by the numeric value of the offset.

Thus, any change to sensor frequency will automatically be included in the time range calculation for a formula.

An offset time unit can be used in place of standard time units in the following time range locations:

- Formulas
- References
- Vectors
- Trigger Frequency
- Trigger Term

Below we discuss GUI and CLI examples of how to configure the offset time unit in each of the locations mentioned above.

Offset Used in Formulas

In this example, we are creating a rule, *Rule1*, with a sensor, *Sensor1*. The sensor frequency is set to 10 seconds.

In [Figure 34 on page 96](#) below, you can see the **Fields** block definition for Rule1 with the **Sensors** block definition framed in green. The formula, *formula1*, has a **Time range** value of *20*

Figure 34: Offset in a Formula

The screenshot shows a rule configuration interface for a rule named 'Rule1'. The rule is of type 'external' and has a frequency of '10s'. The rule description is 'Demonstration Rule' and the synopsis is 'This rule is used only to demonstrate various rule features and concepts.' The field aggregation time-range is set to '20s'.

The 'Fields' tab is selected, showing a list of fields: 'formula1' and 'field1'. The 'formula1' field is selected, and its configuration is shown on the right. The field name is 'formula1', the description is 'Add a description for this field', the field type is 'integer', and the ingest type is 'Formula'. The formula is 'Max' and the field is '\$field1'. The time range is '20s'.

A green box highlights a smaller version of the same interface, illustrating the 'Offset in a Formula' concept. This smaller interface shows the 'Sensors' tab selected, with a sensor named 'Sensor1' of type 'Open Config' and path '/interfaces'. The frequency is '10s'.

A CLI formatted configuration snippet for the same field looks like:

```
user@mgd-97bb5d555-sw87n$ show healthbot topic external rule rule1
synopsis "This rule is used only to demonstrate various rule features and concepts";
description "Demonstration Rule";
sensor Sensor1 {
    open-config {
        sensor-name /interfaces;
        frequency 10s;
    }
}
field field1 {
    constant {
        value 833;
    }
    type integer;
}
field formula1 {
    formula {
        max {
            field-name "$field1";
```



```

        time-range 20;
    }
}
type integer;
}

```

Usage Notes for Offset Used in Formulas

- An offset value applied to the time range of a formula multiplies the rule, sensor, or trigger frequency of that rule.
- The result of this example is that the time range for the Max formula is now 2 times the *Sensor1* frequency of 10 seconds, or 20 seconds ($2 * 10s = 20s$).
- If a frequency profile of 30 seconds is applied to the rule used in the example, then the resulting time-range would be 60 seconds ($2 * 30s = 60s$).
- Offset time can be applied to the following formulas: latest, count, min, max, sum, mean, on-change, and stdev.

Offset Used in References

In this example, there are two rules in play, but only one is shown. The unseen rule, Rule1, is in the topic routing-engines and is named routing-engines (routing-engines/routing-engines). Rule1 has a frequency of 20 seconds. Rule1 is referenced .

Rule2, shown in [Figure 35 on page 98](#), is a network rule which has a reference field named *ref1*. The field, *ref1*, references back to Rule1 through the **Reference XPath Expression** with a **Time Range** setting of 3offset.

Figure 35: Offset Time Used in Reference Field

Rule: external / Rule2 Rule Frequency: 10s Network Rule

SAVE & DEPLOY

SAVE

DELETE

CLONE

Description: Demonstration Rule

Synopsis: This rule is used only to demonstrate various rule features and concepts.

Field aggregation time-range:

Sensors

Fields

Vectors

Variables

Functions

Triggers

Rule Properties

+ ADD FIELD

formula1

ref1

DELETE REF1

Field Name * ?

ref1

Description ?

Add a description for this field

Field Type

integer

Add to Rule Key ?

Ingest type (Field source)

Reference

Reference XPATH expression *

Time Range

Data if missing

"/device-group[device-group-name='Core4']/device[device-id='R1']/topic[topic-name='routing']

3offset

Default value

Usage Notes for Offset Used in References

- Offset values used in references multiply the frequency of the referenced rule by the offset value. In this case, the 20 second frequency of Rule1 is multiplied by 3, resulting in a 60 second time-range (3 * 20s = 60s).
- If a frequency profile of 60 seconds (60s) was applied to Rule1, the time range for the reference would increase to 180 seconds (3 * 60s = 180s).

Offset Used in Vectors

In this example, there are 3 rules at play. Rules *Rule1* and *Rule2* are not shown but are referenced by the vector.

Rule1 is in topic line-cards and is named line-cards (line-cards/line-cards) and has a frequency of 20 seconds (20s).

Rule2 is in topic routing-engines and is named routing-engines (routing-engines/routing-engines) and has a frequency of 30 seconds (30s).

In *Rule3* below, we have defined a vector, *vector1* that consists of 2 path references and 1 field. The vector has a **Time Range** defined as 3offset. [Figure 36 on page 99](#) shows the vector block definition in the Paragon Insights GUI.

Figure 36: Offset Time Used in Vector Block

The screenshot shows the Paragon Insights GUI for configuring a rule. The rule is named 'Rule3' and has a frequency of '10s'. The 'Vectors' tab is selected, showing a vector named 'vector1'. The vector configuration includes:

- Vector Name:** vector1
- Ingest Type:** path
- Time-Range:** 3offset
- References:**
 - "/device-group[device-group-name='Core4']/device[device-id='R1']/topic[topic-name='routing-engines']/rule[rule-name='routing-engines']/field[slot='0']/ref1"
 - "/device-group[device-group-name='Core4']/device[device-id='R1']/topic[topic-name='line-cards']/rule[rule-name='line-cards']/memory"
 - \$formula1

Usage Notes for Offset Used in Vectors

- An offset value defined in the time range of a vector multiplies the sensor or rule frequency of the referenced rule or sensor. If a field from the same rule is used as the reference, then the frequency of the rule containing the vector is used.
- When multiple references or fields are defined for a vector and offset time is used, the offset value is applied to each path independently. So, for this example in which our offset value is 3 (3offset):
 - The path reference to Rule1: `"/device-group[device-group-name='Core4']/device[device-id='R1']/topic[topic-name='line-cards']/rule[rule-name='line-cards']/memory"` which has a frequency of 20 seconds, would result in a time range of 60 seconds for the vector ($3 * 20s = 60s$).
 - The path reference to Rule2: `"/device-group[device-group-name='Core4']/device[device-id='R1']/topic[topic-name='routing-engines']/rule[rule-name='routing-engines']/field[slot='0']/ref1"` which has a frequency of 30 seconds, would result in a time range of 90 seconds for the vector ($3 * 30s = 90s$).

NOTE: There are no spaces or line breaks in the path references. They are added to this document only to enhance readability.

- The field reference to `formula1`: Since `formula1` is a normal field used within the vector block, the rule frequency of the current rule is used as a basis. So, the time range for `formula1` is 30 seconds ($3 * 10s = 30s$).
- If a frequency profile of 60 seconds is applied to Rule1 (line-cards/line-cards), then the time range for that path in the vector would be 180 seconds ($3 * 60s = 180s$).
- If a frequency profile of 120 seconds is applied to Rule2 (routing-engines/routing-engines), then the time range for that path in the vector would be 360 seconds ($3 * 120s = 360s$).
- The time range for the field reference, `formula1`, would remain the same as before at 30 seconds unless a change is made to Rule3 or a frequency profile with a different time is applied to Rule3.

Offset Used in Triggers

In this example, we have one rule, Rule1. [Figure 37 on page 101](#) below shows the trigger block definition with the sensor block definition overlaid with a green border.

The rule, *Rule1*, has a sensor frequency of 10 seconds (10s) applied.

The trigger itself, *Trigger1*, has an offset frequency of 2 (2o) applied.

The trigger term, *term1*, has its own time range offset of 2 (2o) applied as well.

Figure 37: Offset Time Used in Trigger Block

The screenshot displays the 'Network Rule' configuration page for 'Rule1' in the 'external' namespace. The interface is divided into two main sections: 'Trigger' and 'Sensor'.

Trigger Configuration:

- Trigger Name:** trigger1
- Frequency:** 2o (representing 20 seconds)
- Disable alarm deduplication:** ☐
- Term:** term1
- WHEN clause:**
 - Left operand:** \$name
 - Operator:** FPC1
 - Right operand:** 2o (representing 20 seconds)

Sensor Configuration:

- Sensor Name:** Sensor1
- Sensor Type:** Open Config
- Sensor Path:** /interfaces
- Frequency:** 10s (representing 10 seconds)

Usage Notes for Offset Time Used in Triggers

- An offset value defined in trigger frequency multiplies the sensor or rule frequency. So, the offset value of 2 in *trigger1* causes the trigger frequency to be interpreted as 20 seconds ($2 * 10s = 20s$) because the sensor frequency of the rule is used as the basis.
- An offset value defined in a trigger term multiplies the trigger frequency value. So, the offset value of 2 in *term1* causes the term frequency to be interpreted as 40 seconds ($2 * 20s = 40s$).

Offset Used in Trigger Reference

In this example, we have 2 rules in the topic external, and one frequency profile, *prof1*.

The rules are:

- external/test*: The *test* rule has a sensor named components which is an OpenConfig sensor with a sensor path of */components* and a sensor frequency of 10 seconds.

It also has a trigger, *trig1* with a frequency of 2o or 2offset. The trigger has a term named *Term_1*, which is not used in the example.

The screenshot shows a configuration interface for a rule named `external/test`. The interface has several tabs: **Sensors**, **Fields**, **Vectors**, **Variables**, **Functions**, and **Triggers**. The **Triggers** tab is currently selected and highlighted with a green border. In the **Sensors** tab, a sensor named `components` is visible. The **Triggers** tab contains a list of triggers, with `trig1` being the first one. To the right of the trigger list, there are configuration options for `trig1`: **Trigger Name** (set to `trig1`), **Frequency** (set to `2o`), a toggle for **Disable alarm deduplication** (currently off), and a **Term** dropdown menu (set to `Term_1`). Below these options is a blue button labeled **+ ADD TERM**. At the bottom of the **Triggers** tab, there are two input fields: **Sensor Path** (set to `/components`) and **Frequency** (set to `10s`).

- *external/ref*: The *ref* rule is a non-sensor rule, which means that the rule uses a reference field, *trigger_reference*, to reference the sensor defined in another rule; in this case *external/test*.

Rule: external / ref

Rule Frequency: 30s

Network Rule

SAVE & DEPLOY

SAVE

Description: Description

Synopsis: Synopsis

Field aggregation time-range:

Sensors

Fields

Vectors

Variables

Functions

Triggers

Rule Properties

+ ADD FIELD

trigger_reference

Field Name * ?

trigger_reference

Description ?

Add a description for this field

Field Type

Field type

☐ Add to Rule Key ?

Ingest type (Field source)

Reference

Reference XPATH expression *

/topic[topic-name='external']/rule[rule-name='test']/trigger[trigger-name=trig1]/color

Time Range

2offset

And the frequency profile is:

- *prof1*: The *prof1* profile sets the frequency for the */components* sensor at 30 seconds.

Edit Frequency Profile: prof1

Name

prof1

SENSORS

Sensor Name

/components

Frequency *

30seconds

+

ADD SENSORS

NON-SENSORS

+

ADD NON-SENSORS

CANCEL

SAVE

SAVE & DEPLOY

2

CHAPTER

Paragon Insights Management and Monitoring

- [Manage Paragon Insights Users and Groups | 107](#)
- [Manage Devices, Device Groups, and Network Groups | 121](#)
- [Paragon Insights Rules and Playbooks | 141](#)
- [Monitor Device and Network Health | 172](#)
- [Understand Resources and Dependencies | 206](#)
- [About the Resources Page | 209](#)
- [Add Resources for Root Cause Analysis | 212](#)
- [Configure Dependency in Resources | 215](#)
- [Example Configuration: OSPF Resource and Dependency | 221](#)
- [Edit Resources and Dependencies | 232](#)
- [Filter Resources | 234](#)
- [Upload Resources | 235](#)
- [Download Resources | 236](#)
- [Clone Resources | 236](#)
- [Delete Resources and Dependencies | 237](#)
- [Monitor Network Device Health Using Grafana | 239](#)
- [Understanding Action Engine Workflows | 244](#)
- [Manage Action Engine Workflows | 244](#)

[Alerts and Notifications | 252](#)

[Generate Reports | 268](#)

[Use Exim4 for E-Mails | 282](#)

[Configure the Exim4 Agent to Send E-mail | 283](#)

[Manage Audit Logs | 284](#)

[Configure a Secure Data Connection for Paragon Insights Devices | 286](#)

[Configure Data Summarization | 290](#)

[Modify the UDA, UDF, and Workflow Engines | 299](#)

[Commit or Roll Back Configuration Changes in Paragon Insights | 307](#)

[Logs for Paragon Insights Services | 309](#)

[Troubleshooting | 312](#)

[Paragon Insights Configuration – Backup and Restore | 321](#)

Manage Paragon Insights Users and Groups

IN THIS SECTION

- [Default User and First Login | 108](#)
- [Default User Roles | 109](#)
- [User Management | 109](#)
- [Group Management | 114](#)
- [LDAP Authentication in Paragon Insights | 116](#)
- [Password Recovery | 119](#)
- [Limitations | 120](#)

HealthBot Release 3.0.0 employs role-based access control (RBAC) to control access to the user interface, and tools and objects. RBAC is applied to user groups that are made up of a list of users.

The use of access controls within Paragon Insights (formerly HealthBot) allows you to grant one group of users, like operators, read-only access to certain pages like **Configuration > Device Configuration**; while granting a different group of users, like administrators, read-write access to that same page.

Starting with Release 4.0.0, Paragon Insights executes user management, authentication, and authorization through Identity and Access Management (IAM) service available in the 4.0.0 installation package.,

There are no changes in the installation process. See [Paragon Insights Installation Guide](#) for installation or migration procedure.

In new installations of Paragon Insights Release 4.0.0, a user can be registered through e-mail. This mode of registration requires you to perform additional steps at the time of installation. For existing Paragon Insights installations, you can register new users with username, without including an e-mail address. For more information on first login, see "[Default User and First Login](#)" on page 108.

NOTE: Starting from Paragon Insights Release 4.1.0, username is case insensitive.

In Paragon Insights, there are two administrators: one is the default *admin* user who first logs into Insights after a new installation. The *admin* user has complete control over all of Insight's access

controls. The other is sp-admin user who is created in the Paragon Insights interface. To know more about roles in Paragon Insights, see ["Default User Roles" on page 109](#).

Paragon Insights 4.0.0 also supports Lightweight Directory Access Protocol (LDAP) based authentication. The authorization data such as organizational ID, username, and password are stored and managed in IAM. For more information, see ["LDAP Authentication in Paragon Insights" on page 116](#).

Default User and First Login

In standalone Paragon Insights installations, the default username and password are set as *admin* and *Admin123!*, respectively. The admin user has complete control over all of Paragon Insights' access controls. The credentials above are used for the first login at the URL **`https:// <Paragon Insights hostname or IP>:8080`**.

Upon successful first login and before the admin user is granted access to the GUI, they are required to create a new password. The **Set Password** window pops up and provides instructions regarding password length, capitalization, special characters, and so on. Once you save this password, a pop-up window notifies you that the password has been changed. From this time forward, the admin user logs in with the new password.

If Paragon Insights is migrated from 3.x.x or earlier versions to 4.0.0 version, the admin user creates other users and assigns them roles and an initial password in the Administration > User Management page. The users created with sp-admin and sp-operator roles can login for the first time with their *username* by entering the initial password provided by the *admin* user.

NOTE: All users must change their password after the first login.

To change password:

1. Click the circle with the first letter of your username at the top right corner of the interface.
2. Click change password in the drop-down menu.

A Change Password window appears.

3. Enter your current password. Enter your new password and re-enter your new password to confirm.

Passwords must be at least 8 characters long and must contain uppercase letters, lowercase letters, numbers, and special characters.

4. Click **OK**.

A window notifies that your password is changed successfully.

Starting with standalone Paragon Insights 4.0.0 installations, the admin user can register a user with an e-mail address if Insights is configured for this registration method during installation. The registered user gets a login link in their inbox that expires after 24 hours. When they click on the link, a **Set Password** window appears where they can set a new password before they log into Paragon Insights.

Default User Roles

In Paragon Insights 4.0.0, the hbadmin group in earlier releases is converted to the sp-admin role whereas the hbmonitor, hbconfig, and hboperator groups are merged into the sp-operator role.

Paragon Insights is shipped with two pre-defined user roles:

- *sp-admin* – The user gets read and write access to add resources such as device groups, network groups, rules and playbooks, configure data summarization profiles, create backup of Paragon Insights configuration or time series database, and the ability to manage users and groups.
- *sp-operator* – Provides login capability and the ability to read-only access to read and observe any configured entity in Insights.

None of the pre-defined user roles can be changed or removed.

User Management

The **User Management** page is the first page shown when you navigate to **Administration > User Management** from the left navigation bar. This page is used to:

- **View a list of current Paragon Insights users**

The list shows user details including username, role, status, and provider type. User status can be active (green) or inactive (red).

- **Add new users**

Click the + to bring up the **Create User** window. Enter the following details.

NOTE: In Paragon Insights Release 4.0.0, an sp-admin can map a user to a role without creating user groups. The sp-admin can also create user groups, associate roles to user groups and then, add users to the user groups.

Table 4: Create User Fields for Installations without E-mail Registration

Fields	Description
Username	<p>Enter a username of maximum 32 characters. The <i>username</i> is used to log into the Paragon Insights portal.</p> <p>NOTE: Starting from Paragon Insights Release 4.1.0, username is case insensitive.</p>
First Name	Enter the first name of the user. You cannot exceed 32 characters.
Last Name	Enter the last name of the user. You cannot exceed 32 characters.
Status	Enable or disable the user. If you disable the user, they cannot log into the Paragon Insights portal.
Provider Type	<p>There are two provider types – Local (IAM) and LDAP.</p> <p>You can choose Local to configure users in IAM or choose LDAP to map user to LDAP user group.</p>
(Optional) Mapping Provider Group	If you choose the provider type as LDAP, you can enter the LDAP user group name in this field.
Password	<p>Enter a password for the user.</p> <p>Passwords must be at least 8 characters long and must contain uppercase letters, lowercase letters, numbers, and special characters.</p> <p>A password must be unique and must not be previously used passwords.</p>
Role	<p>Select multiple roles at the left-side panel and click the right arrow button to add the roles to the user.</p> <p>The roles are sp-admin, sp-operator, or a custom role with select create, read, update, and delete access permissions.</p>

If you configured Paragon Insights to register users using e-mail address, you must configure SMTP settings in the portal before adding users. The SMTP configuration is used to send user account related e-mails, such as when user changes password, user resets password, admin user adds a new user, and so on.

NOTE: If you want to configure the SMTP settings, you must configure the environment variables `export HB_IAM_SKIP_MAIL_VERIFICATION=false` and `export HB_IAM_DISABLE_SMTP_SETTINGS=false` during Paragon Insights installation. See [installation guide](#) for more information.

To configure SMTP settings:

1. Select **Administration > Authentication > SMTP Settings**.

The SMTP Settings page appears. Fill in the details described in [Table 5 on page 111](#).

Table 5: Fields in SMTP Settings

Fields	Description
Server Address	Enter the SMTP server address. For example, smtp.domain.com
TLS	Toggle the switch on to enable TLS, if you want to encrypt the e-mails sent to your users' account from Paragon Insights.
Port Number	Enter the port number. The standard port number is set to 25 if TLS is disabled and is set to 587 if you enable TLS in SMTP settings.
SMTP Authentication	
SMTP Authentitcation (Optional)	Enable SMTP authentication to allow only verified users to send e-mails to or receive e-mails from the Paragon Insights application.
Username	Enter the username to be used for authentication. The username must not exceed 32 characters. NOTE: Starting from Paragon Insights Release 4.1.0, username is case insensitive.
Password	Enter a password.

Table 5: Fields in SMTP Settings (Continued)

Fields	Description
Confirm Password	Re-enter the password.
From Name	If you did not enable SMTP authentication, then you must enter this field. This name appears as sender's name to the e-mail recipients.
From Email Address	Enter the e-mail address from which messages from Paragon Insights must be sent to recipients. The syntax is example@domain.com
Test SMTP Settings	
Email Address	Enter your e-mail address to check if SMTP settings configured in previous fields work as intended. Click Send Test Email . If you receive an e-mail from Paragon Insights in the inbox of the e-mail address entered in this field, then you have successfully configured SMTP settings.

2. Click **Save**.

The SMTP Settings for Paragon Insights is complete. You can now register users using their e-mail address.

To register a user using e-mail address:

1. Select **Administration > User Management > User** in the left navigation bar.

The Users page appears.

2. Click on the + icon to add a new user.

The Create User page appears. Fill in the following details.

Table 6: Create User Fields for Installations with E-mail Registration

Fields	Descriptions
First Name	Enter the first name of the user. You cannot exceed 32 characters.
Last Name	Enter the last name of the user. You cannot exceed 32 characters.
Status	Enable or disable the user. If you disable the user, they cannot log into the Paragon Insights portal.
Username (E-mail)	Enter the e-mail address of the user that will be used to log into the Paragon Insights portal.
Role	<p>Select multiple roles at the left-side panel and click the right arrow button to add the roles to the user.</p> <p>The roles are sp-admin, sp-operator, or a custom role with select create, read, update, and delete access permissions.</p>

3. Click **OK**.

The user you added is listed in the Users page.

- **Edit existing users**

Select an existing user by clicking anywhere on that user's line in the list. Then click the **Edit User (Pencil)** icon to bring up the **Edit User** window. You can change any parameter except the username and the Provider Type.

- **Delete a user**

Select an existing user by clicking anywhere on that user's line in the list. Then click the **Delete User (Trash Can)** icon. Confirm the action and the user is deleted.

NOTE:

- If you set a user's status to inactive or delete that user, they are immediately prevented from logging in to Paragon Insights through the login page.

You can also export (backup) user configurations and restore the configurations in Paragon Insights. The backup and restore feature is not applied to pre-canned roles. For more information, see ["Paragon Insights Configuration – Backup and Restore" on page 321](#).

Group Management

A user group is a collection of roles to which a Paragon Insights user can be assigned. The roles within a user group define the access (read-only or read-write) that all members of the group have in common. In other words, user groups are where RBAC controls are applied.

The **User Groups** page is accessed by navigating to **Administration > User Management** from the left-nav and selecting **User Groups** on the left side of the **User Management** page.

- **View a list of current Paragon Insights user groups**

The list shows user group details including group name and description.

- **Add new user groups**

Click the + to bring up the **Add Group** window.

Starting in HealthBot Release 3.1.0, RBAC has been enhanced to include the roles selector helper. The roles selector helper appears when you add or edit a user group. See [Figure 38 on page 115](#).

Figure 38: Add User Group

Add Group

Group Name*

uGroup1

Unique Name for the group

Group Description

New user group for documentation purposes.

Description of the group

ROLES SELECTOR HELPER

Search

Functionality	Read	Write
> Dashboard	<input checked="" type="checkbox"/>	<input type="checkbox"/>
> Monitor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
> Configuration	<input checked="" type="checkbox"/>	<input type="checkbox"/>
> Settings	<input type="checkbox"/>	<input type="checkbox"/>
> Administration	<input type="checkbox"/>	<input type="checkbox"/>

Include Access Roles

SELECT ALL

CLEAR

Selected Roles

System Roles*

R configuration

W configuration

R data-store

W data-store

R data

R debug

R device-groups

R device

R devices

R event

R files

W first-login

R health-tree

R health

R license

W login

W logout

W mgd

R network-groups

R playbook

R playbooks

R sensors

R system-settings

W token

R topic

R topics

R tsdb-counters

R user-profile

W user-profile

GUI Roles

R ui-configuration-device

R ui-configuration-devicegroups

R ui-configuration-network

R ui-configuration-playbooks

R ui-configuration-rules

R ui-dashboard-device-list

R ui-dashboard-device-status

R ui-dashboard-device-vendors

R ui-dashboard-devicegroup-list

R ui-dashboard-devicegroup-status

R ui-dashboard-network-status

R ui-dashboard-networkgroup-list

R ui-dashboard-tsdb

R ui-monitor-alarms

R ui-monitor-device-health

R ui-monitor-devicegroup-health

R ui-monitor-graph

R ui-monitor-network-health

Associated Users

CANCEL

SAVE

SAVE & DEPLOY

- **Edit existing user groups**

Select an existing user group by clicking anywhere on that group's line in the list. Then click the **Edit User (Pencil)** icon to bring up the **Edit <groupname>** window.

NOTE: When you add or edit a user group, the window has sections called **System Roles** and **GUI Roles** under the **Selected Roles** pull-down. These sections show the specific read-only (R) or read-write (W) permissions that are assigned to the group as a result of the selections made in the **ROLES SELECTOR HELPER**.

- **Delete a user group**

Select an existing user group by clicking anywhere on that group's line in the list. Then click the **Delete User (Trash Can)** icon. A confirmation window appears. Confirm the action (Save and Deploy) to complete the deletion. The pre-defined user groups hbdefault and hbadmin cannot be deleted.



WARNING: Adding and editing user groups in Paragon Insights is an advanced feature that requires a deep understanding of the available roles and how they apply to RBAC. We recommend that you use only the Role Selector check-boxes to add or remove permissions. We do not recommend that you add or remove individual system or GUI roles.

LDAP Authentication in Paragon Insights

LDAP users can log into Paragon Insights GUI using LDAP credentials after an sp-admin configures LDAP settings in Paragon Insights. The sp-admin must also map the LDAP user group to the Paragon Insights user group. In Paragon Insights Release 4.0.0 and later releases, Active Directory service installed on Windows Server 2012 R2 and OpenLDAP version 2.4 as the protocol are validated for LDAP implementation.

A typical workflow of LDAP-based authentication involves the following steps:

1. An LDAP administrator configures LDAP group in an external server and adds users to the LDAP group.
2. The sp-admin configures LDAP settings in Paragon Insights.
3. The sp-admin creates a user group for LDAP users in Paragon Insights Release 4.0.0 interface, maps this user group to the existing LDAP user group, and then assigns roles to that user group.

NOTE: The Paragon Insights user group and LDAP user group must have the same name.

During authentication, the LDAP server produces a list of LDAP groups associated with the user. The Paragon Insights IAM service checks for the corresponding user group name in Paragon Insights and generates roles associated with that Paragon Insights user group. The IAM service then converts the roles into a JSON Web Token (JWT) that is used for authorizing the LDAP user in Paragon Insights.

NOTE: If a user is configured both in LDAP and IAM (locally), then LDAP takes priority over IAM during authentication. When the user tries to login, Paragon Insights checks the user details first in LDAP and then in IAM.

To configure LDAP settings in Paragon Insights:

1. Click **Administration > Authentication > LDAP Settings** option in the left navigation bar.
2. Enter the necessary fields in the LDAP Settings page.

The following table describes the attributes in the LDAP Settings page.

Table 7: Configure LDAP to Integrate with Paragon Insights

Attributes	Description
LDAP Server	
Server Address	Enter the LDAP server url. For example, ldap.example.net.
SSL	Enable SSL to encrypt the LDAP channel.
Port Number	Enter the port number for the LDAP server. The default port number if SSL is enabled is 636 and the default port without SSL is 389.
LDAP Authentication	

Table 7: Configure LDAP to Integrate with Paragon Insights (Continued)

Attributes	Description
Authentication Method	The authentication method is set to <i>Simple</i> . The password sent from the client to bind to the LDAP server is plain text.
Base Domain Name	Enter the domain name that constitutes the search base for querying the LDAP server. For example: dc=mycompany, dc=net/com .
Bind Domain Name	Enter the user name configured for LDAP authentication. For example: user@mycompany.net .
Bind Password	Enter a password for LDAP authentication.
User Options (Optional)	
User Attribute	Setting a user attribute is optional. This filter improves the search functionality on the LDAP server using the specified attribute name.
User Filter	Specify the objectClass attribute to filter the type of entities that can access Paragon Insights. For example, <i>Person</i> as a user filter.

3. Click *Save*.

The configuration settings of LDAP server in Paragon Insights is complete.

After configuring LDAP settings, the sp-admin must create an LDAP user group in Paragon Insights to map the users created in LDAP server to Paragon Insights. The LDAP group created in Paragon Insights allows sp-admins to map roles for LDAP users.

To map an LDAP group to Paragon Insights user group:

1. Click *Administration > User Management > User Groups* option in the left navigation bar.

The User Group page appears.

2. Click the plus icon to create a new user group.

The Create User Group page appears.

3. Enter a group name and select **Provider Type** as LDAP from the drop down menu.

The **Mapping Provider Group** section appears.

4. In the **Mapping Provider Group** field, enter the LDAP group name.
5. Select the roles to be associated with the LDAP group in Paragon Insights and click **OK**.

The users configured in LDAP server can log into Paragon Insights by entering their LDAP credentials. The resources and pages accessible to the LDAP user depends on the permissions granted in the role mapped through Paragon Insights.

Password Recovery

The default admin user does not require an e-mail address to access the interface in standalone deployment. If the initial password set by an admin user is lost, it can be recovered by a system administrator who has access to the physical server or virtual machine that hosts the Paragon Insights application. The system administrator has to run the following curl command in any shell in one of the nodes in the Kubernetes cluster.

Curl command to reset Paragon Insights admin user credential using IAM service token.

```
curl -k --request POST 'https://{server-ip}:{port}/iam/reset-password' --header 'x-service-token: '$
(kubectl get secret -n {{namespace}} $(kubectl get sa -n {{namespace}} iam -o jsonpath='{.secrets[0].name}')
```

```
-o jsonpath='{.data.token}' | base64 --decode)'' --header 'x-service-scope: {}' --header 'Content-Type:
application/json' --data '{
  "user_name" : "{{username}}",
  "new_password" : "{{password}}"
```

```
}'
```

The IAM service validates the token and resets the password for the admin user.

NOTE: The port number for standalone Paragon Insights deployment is 8080 and the namespace is **healthbot**. The server-ip in the POST request body denotes the virtual IP address you configured for Paragon Insights services during the installation.

There is currently no self-service type of lost password mechanism for users registered without an e-mail address. Password reset must be done manually by an administrator with read-write access to the **User Management** page. The administrator must edit the user, change the password, and then notify the

user by appropriate means. The default password expiry for users with sp-admin and sp-operator roles is 180 days.

To recover password of user accounts registered with e-mail address:

1. Enter your username (e-mail address) in the Paragon Insights login page.
2. Place the cursor in the password field.

The **Forgot Password?** link appears beneath the **Log in** button.

3. Click on the **Forgot Password?** link.

A Forgot Password window appears displaying the message that an e-mail with link to reset password is sent to your account.

4. Click on the **Reset your password** button in your account recovery e-mail.

The reset password link expires after 24 hours.

5. In the **Set Password** window, enter a new password and enter the same password in the **Confirm Password** field.

Passwords must be at least 8 characters long and must contain uppercase letters, lowercase letters, numbers, and special characters.

A password must be unique and must not be previously used passphrases.

6. Click **OK**.

You will receive a second e-mail notifying you that your password is changed. Log into Paragon Insights using your latest password.

Limitations

In HealthBot Release 3.1.0, the RBAC implementation is limited in some ways:

- The available roles, such as **R-Devices**, **W-Devices**, **R-Datastore**, etc. are all pre-defined. There is no way to add new roles or delete existing roles.
- All roles are endpoint driven, not specific to any resource. This means that if you have read permission for devices, you can read all devices in the system. There is no means to restrict the read access to a subset of devices.
- Roles are permissive in nature. You cannot create a role that blocks access to any given endpoint such as rules. If a user is created but not given any group membership, they will not be able to access the Paragon Insights GUI.

- RBAC is currently limited to API service. This means that if you have read-only access to a page such as **Configuration > Devices**, you can see the entire page and interact with all of its controls. You could even go through the motions of creating a device in the GUI. However, when you click **SAVE** or **SAVE & DEPLOY** an API is called and it will recognize that you do not have the required permission to create a device. Errors are displayed at that time.
- If you migrate data from your existing 2.1.X installation to your 3.0.0 or later installation, user data is not migrated. Any existing users must be recreated manually, by the admin user, after migration.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.0.0	Starting with Release 4.0.0, Paragon Insights executes user management, authentication, and authorization through Identity and Access Management (IAM) service available in the 4.0.0 installation package.
3.1.0	Starting in HealthBot Release 3.1.0, RBAC has been enhanced to include the roles selector helper
3.0.0	HealthBot Release 3.0.0 employs role-based access control (RBAC) to control access to the user interface, and tools and objects.

RELATED DOCUMENTATION

| [Paragon Insights API Guide](#)

Manage Devices, Device Groups, and Network Groups

IN THIS SECTION

- [Adding a Device | 122](#)
- [Editing a Device | 129](#)
- [Adding a Device Group | 129](#)

- [Editing a Device Group | 136](#)
- [Configuring a Retention Policy for the Time Series Database | 136](#)
- [Adding a Network Group | 137](#)
- [Editing a Network Group | 140](#)

Use the appropriate Configuration pages from the left-navigation menu to manage devices, device groups, and network groups. Paragon Insights (formerly HealthBot) supports both Junos devices by default and third party vendor devices with the required license installed. You must add a device to one or more device groups or create a network group before you can apply Paragon Insights rules and playbooks to a device. Network groups allow you to correlate health status data between multiple devices across the network. For example, you can create a network group that monitors the ping times between two or more devices and notifies you if the ping times are too high.

Adding a Device

To add a device:

1. Click the **Configuration > Device** option in the left-nav bar.
2. Click the add device button (+).
3. Enter the necessary values in the text boxes and select the appropriate options for the device.

The following table describes the attributes in the **Add a Device** window:

Attributes	Description
Name	Name of the device. Default is hostname. (Required)
Hostname / IP Address / Range	Hostname or IP address of a single device. If you are providing a range of IP addresses, enter the IP address for the device that marks the start and end of the address range. (Required)

(Continued)

Attributes	Description
System ID to use for JTI	<p>Unique system identifier required for JTI native sensors. Junos devices use the following format: <code><host_name>:<jti_ip_address></code></p> <p>When a device has dual routing engines (REs), it might send different system IDs depending on which RE is primary. You can use a regular expression to match both system IDs.</p>
Flow/IFA Source IPs	<p>Enter the IP address(es) that this device uses to send NetFlow data to Paragon Insights.</p> <p>The IP address or addresses are used to send probe packets for flow monitoring using Inband Flow Analyzer (IFA).</p> <p>If there are more than one IP address, separate them with a comma.</p>
OpenConfig Port Number	<p>Port number required for JTI OpenConfig sensors. The default value is 32767.</p>
iAgent Port Number	<p>Port number required for iAgent. The default value is 830.</p>

(Continued)

Attributes	Description
Vendor	<p data-bbox="857 361 1370 426">Lists the vendor or supplier of the device you are using.</p> <p data-bbox="857 457 1370 556">The operating system you can select from the OS drop-down list depends on the vendor you select from the Vendor drop-down list.</p> <p data-bbox="857 588 1398 653">The following are the list of options you can choose from.</p> <ul data-bbox="857 684 1409 1596" style="list-style-type: none"> <li data-bbox="857 684 1409 783">• Select Juniper from the vendor drop-down list to select either Junos or Junos Evolved operating systems from the OS drop-down list. <li data-bbox="857 821 1409 919">• Select CISCO from the Vendor drop-down list to select either IOSXR or NXOS operating systems from the OS drop-down list. <p data-bbox="894 951 1409 1016">With Paragon Insights Release 4.0.0, NXOS OS is also supported.</p> <p data-bbox="894 1047 1398 1184">NOTE: If you plan to use Cisco IOS XR devices, you must first configure the telemetry. For more information, see Paragon Insights Installation Requirements</p> <ul data-bbox="857 1220 1419 1730" style="list-style-type: none"> <li data-bbox="857 1220 1398 1318">• Select Arista from the Vendor drop-down list to select EOS operating system from the OS drop-down list. <li data-bbox="857 1356 1419 1455">• Select Paloalto from the Vendor drop-down list to select PANOS operating system from the OS drop-down list. <li data-bbox="857 1493 1398 1591">• Select Linux from the Vendor drop-down list and you can enter the name of the operating system in the OS field. <li data-bbox="857 1629 1419 1730">• If you select Other Vendor, the Vendor Name and OS Name fields are enabled. You can enter the name of the vendor of your choice in the Vendor

(Continued)

Attributes	Description
	<p>Name field and the corresponding operating system for the vendor in the OS field.</p> <p>Consider the following example. If the operating system of a vendor (listed in the Vendor drop-down list) is not listed (in the OS drop-down list), you can select the Other Vendor option to enter name of the vendor and operating system of your choice.</p> <p>Starting with Paragon Insights Release 4.0.0, Paragon Insights supports Arista Networks, Paloalto Networks, and Linux vendors.</p>
Timezone	Timezone for this device, specified as <i>+</i> or <i>-</i> hh:mm. For example, +07:00
Syslog Source IPs	List of IP addresses for the device sending syslog messages to Paragon Insights. For example, 10.10.10.23, 192.168.10.100.
Syslog Hostnames	List of hostnames for the device sending syslog messages to Paragon Insights. For example, router1.example.com.
SNMP	
SNMP Port Number	Port number required for SNMP. The port number is set to the standard value of 161 .
SNMP Version	Select either v2c or v3 in the drop-down menu.

(Continued)

Attributes	Description
SNMP Community	<p>This field appears if you selected v2c in SNMP Version field.</p> <p>Enter an SNMP Community string if you configure SNMPv2c for traps and ingest.</p> <p>In SNMPv2c, Community string is used to verify the authenticity of the trap message issued by the SNMP agent (devices such as routers, switches, servers, and so on).</p>
Authentication None	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enable this option on if you want to set SNMPv3 authentication to <i>None</i>.</p>
Protocol None	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enable this option on if you want to set SNMPv3 protocol to <i>None</i>.</p>
SNMPv3 Username	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enter a username for trap notifications. The <i>username</i> you enter here is checked against the SNMPv3 users configured in Paragon Insights.</p> <p>If there is a match, the trap message is further processed else, the message is dropped.</p>
Context Engine ID	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enter the <i>Engine ID</i> of the device (SNMP agent) that sends the trap notification.</p> <p>For inform notifications, the <i>Engine ID</i> must be set to that of Paragon Insights.</p>

(Continued)

Attributes	Description
SNMPv3 Authentication Protocol	<p>This field appears if you selected v3 in SNMP Version field and disabled <i>Authentication None</i>.</p> <p>Select an authentication protocol from the drop-down menu.</p> <p>SNMP authentication protocol hashes the SNMP <i>username</i> with the passphrase you enter. The hashed output is sent along with the trap notification message. Paragon Insights again hashes the <i>username</i> with the passphrase you entered for authentication. If the output matches, the trap notification is further processed.</p>
SNMPv3 Authentication Passphrase	<p>This field appears if you selected v3 in SNMP Version field and disabled <i>Privacy None</i>.</p> <p>Enter a passphrase for SNMPv3 authentication.</p>
SNMPv3 Privacy Protocol	<p>Select a privacy protocol from the drop-down menu.</p> <p>Privacy algorithm encrypts the trap notification message with the protocol passphrase so that the message cannot be read by an unauthorized application in the network.</p>
SNMPv3 Privacy Passphrase	<p>This field appears if you selected v3 in SNMP Version field and disabled <i>Privacy None</i>.</p> <p>Enter a passphrase to encrypt the trap notification.</p>
Source IP Address	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enter the source IP address of the device.</p> <p>If you use NAT or an SNMP Proxy, the <i>Context Engine ID</i> cannot be used to identify the device that send trap notifications. In such cases, the source IP address of the device is used to verify the source of trap notifications.</p>

(Continued)

Attributes	Description
Authentication (Required either here or at Device Group level)	
Password	<p>Username Authentication username.</p> <p>Password Authentication password.</p>
SSL	<p>Server Common Name Server name protected by the SSL certificate.</p> <p>CA Profile* Choose the applicable CA profile(s) from the drop-down list.</p> <p>Local Certificate* Choose the applicable local certificate profile(s) from the drop-down list.</p>
SSH	<p>SSH Key Profile* Choose the applicable SSH key profile(s) from the drop-down list.</p> <p>Username Authentication username.</p>
Outbound SSH	
Reset	The reset option is set by default. If you disabled outbound SSH for this device, you can enable it back by selecting Reset in the dropdown menu.
Disable	You can disable outbound SSH connections for a device by selecting Disable from the dropdown menu.

*To edit or view details about saved security profiles, go to the **Security** page under the **Settings** menu in the left-navigation bar.

4. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration. For information on how to use the Devices table, see ["Monitor Device and Network Health" on page 172](#).

Editing a Device

To edit a device:

1. Click the **Configuration > Device** option in the left-nav bar.
2. Click anywhere on the line that contains the device name in the table under **DEVICES**.
You can search and filter the device names in the table.
3. Click the **Pencil** (Edit Device) icon.
4. Modify the attributes, as needed.
See ["Adding a Device" on page 122](#) for a description of each attribute.
5. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration. For information on how to use the Devices table, see ["Monitor Device and Network Health" on page 172](#).
6. (Optional) A device can be deleted by clicking the **Trash Can** (Delete Device) icon with the device selected.

Adding a Device Group

To add a device group:

1. Click the **Configuration > Device Group** option in the left-nav bar.
2. Click the add group button (+).
The Add Device Group page appears.
3. Configure the device group with the details described in [Table 8 on page 130](#).
4. Do one of the following:
 - Click **Save**—Paragon Insights saves the configuration but does not initiate operations based on the saved configuration.

You can use this option to save make multiple changes in Paragon Insights configurations and commit the changes in bulk or to roll back the changes. See ["Commit or Roll Back Configuration Changes in Paragon Insights" on page 307](#) for more information.
 - Click **Save & Deploy**—Paragon Insights saves and deploys the configuration.

The configuration changes are applied to the IFA ingest service in Paragon Insights.

Table 8: Fields in Device Group Configuration

Attributes	Description
Name	Name of the device group. (Required)
Description	Description for the device group.
Devices	<p>Add devices to the device group from the drop-down list. (Required)</p> <p>Starting in Paragon Insights Release 4.0.0, you can add more than 50 devices per device group. However, the actual scale of the number of devices you can add depends on the available system resources.</p> <p>For example, consider that you want to create a device group of 120 devices. In releases earlier than release 4.0.0, it is recommended that you create three device groups of 50, 50, and 20 devices respectively. With Paragon Insights Release 4.0.0, you just create one device group.</p>
Native Ports	(Native GPB sensors only) List the port numbers on which the Junos Telemetry Interface (JTI) native protocol buffers connections are established.
Flow Ports	(NetFlow sensors only) List the port numbers on which the NetFlow data is received by Paragon Insights. The port numbers must be unique across the entire Paragon Insights installation.
Syslog Ports	Specify the UDP port(s) on which syslog messages are received by Paragon Insights.
Retention Policy	Select a retention policy from the drop-down list for time series data used by root cause analysis (RCA). By default, the retention policy is 7 days.
Disable Trigger Action Scheduler (for a particular device group)	<p>By default, this field is marked False because the option to add a UDA scheduler in Trigger Action page is enabled.</p> <p>You can set this field to True if you want to disable UDA scheduler settings in Trigger Action page.</p>

Table 8: Fields in Device Group Configuration (Continued)

Attributes	Description
Reports	<p>In the Reports field, select one or more health report profile names from the drop-down list to generate reports for the device group. Reports include alarm statistics, device health data, as well as device-specific information (such as hardware and software specifications).</p> <p>To edit or view details about saved health report profiles, go to the System page under the Settings menu in the left-nav bar. The report profiles are listed under Report Settings.</p> <p>For more information, see "Alerts and Notifications" on page 252.</p>
SNMP	
SNMP Port Number	Port number required for SNMP. The port number is set to the standard value of 161 .
SNMP Version	<p>Select either v2c or v3 in the drop-down menu.</p> <ul style="list-style-type: none"> If you select v2c, the SNMP Community name field appears. The string used in v2c authentication is set to <i>public</i> by default. It is recommended that users change the community string. If you select v3, you are given an option to set username, authentication and privacy methods, and authentication and privacy secrets.
Notification Ports	<p>Enter notification ports separated by comma.</p> <p>Paragon Insights listens on these notification ports for trap notification messages.</p>
SNMP Community	Enter an SNMP Community string if you configure SNMPv2c for traps and ingest.
This field appears if you selected v2c in SNMP Version field.	In SNMPv2c, Community string is used to verify the authenticity of the trap message issued by the SNMP agent.
Authentication None	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enable this option on if you want to set SNMPv3 authentication to <i>None</i>.</p>

Table 8: Fields in Device Group Configuration (*Continued*)

Attributes	Description
Protocol None	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enable this option on if you want to set SNMPv3 protocol to <i>None</i>.</p>
SNMPv3 Username	<p>This field appears if you selected v3 in SNMP Version field.</p> <p>Enter a username for trap notifications. The <i>username</i> you enter here is checked against the SNMPv3 users configured in Paragon Insights.</p> <p>If there is a match, the trap message is further processed else, the message is dropped.</p>
SNMPv3 Authentication Protocol	<p>This field appears if you selected v3 in SNMP Version field and disabled <i>Authentication None</i>.</p> <p>Select an authentication protocol from the drop-down menu.</p> <p>SNMP authentication protocol hashes the SNMP <i>username</i> with the passphrase you enter. The hashed output is sent along with the trap notification message. Paragon Insights again hashes the <i>username</i> with the passphrase you entered for authentication. If the output matches, the trap notification is further processed.</p>
SNMPv3 Authentication Passphrase	<p>This field appears if you selected v3 in SNMP Version field and disabled <i>Privacy None</i>.</p> <p>Enter a passphrase for SNMPv3 authentication.</p>
SNMPv3 Privacy Protocol	<p>Select a privacy protocol from the drop-down menu.</p> <p>Privacy algorithm encrypts the trap notification message with the protocol passphrase so that the message cannot be read by an unauthorized application in the network.</p>
SNMPv3 Privacy Passphrase	<p>This field appears if you selected v3 in SNMP Version field and disabled <i>Privacy None</i>.</p> <p>Enter a passphrase to encrypt the trap notification.</p>

Table 8: Fields in Device Group Configuration *(Continued)*

Attributes	Description
Summarization	<p>To improve the performance and disk space utilization of the Paragon Insights time series database, you can configure data summarization methods to summarize the raw data collected by Paragon Insights. Use these fields to configure data summarization:</p> <p>Time Span The time span (in minutes) for which you want to group the data points for data summarization.</p> <p>Summarization Profiles Choose the data summarization profiles from the drop-down list for which you want to apply to the ingest data. To edit or view details about saved data summarization profiles, go to the Data Summarization Profiles page under the Settings menu in the left-nav bar.</p> <p>For more information, see "Configure Data Summarization" on page 290.</p>
Ingest Frequency	Select existing Ingest Frequency Profiles to override rule or sensor frequency settings.
Authentication (Required here or at Device level)	
Password	<p>Username Authentication user name.</p> <p>Password Authentication password.</p>
SSL	<p>Server Common Name Server name protected by the SSL certificate.</p> <p>CA Profile* Choose the applicable CA profile(s) from the drop-down list.</p> <p>Local Certificate* Choose the applicable local certificate profile(s) from the drop-down list.</p>
SSH	<p>SSH Key Profile* Choose the applicable SSH key profile(s) from the drop-down list.</p> <p>Username Authentication username.</p>

Table 8: Fields in Device Group Configuration *(Continued)*

Attributes	Description				
Notifications	<ul style="list-style-type: none"> You can use the Alarm Manager feature to organize, track, and manage KPI event alarm notifications received from Paragon Insights devices. To receive Paragon Insights alarm notifications for KPI events that have occurred on your devices, you must first configure the notification delivery method for each KPI event severity level (Major, Minor, and Normal). Select the delivery method from the drop-down lists. <p>To edit or view details about saved delivery method profiles, go to the System page under the Settings menu in the left-nav bar. The delivery method profiles are listed under Notification Settings.</p> <p>For more information, see "Alerts and Notifications" on page 252.</p>				
Logging Configuration	<p>You can collect different severity levels of logs for the running Paragon Insights services of a device group. Paragon Insights Release 4.0.0 supports collecting log data for SNMP notification.</p> <p>Use these fields to configure which log levels to collect:</p> <table> <tr> <td data-bbox="508 1066 678 1136">Global Log Level</td><td data-bbox="678 1066 1421 1136">From the drop-down list, select the level of the log messages that you want to collect for every running Paragon Insights service for the device group. The level is set to error by default.</td></tr> <tr> <td data-bbox="508 1205 678 1304">Log Level for specific services</td><td data-bbox="678 1205 1421 1304">Select the log level from the drop-down list for any specific service that you want to configure differently from the Global Log Level setting. The log level that you select for a specific service takes precedence over the Global Log Level setting.</td></tr> </table> <p>For more information, see "Logs for Paragon Insights Services" on page 309.</p>	Global Log Level	From the drop-down list, select the level of the log messages that you want to collect for every running Paragon Insights service for the device group. The level is set to error by default.	Log Level for specific services	Select the log level from the drop-down list for any specific service that you want to configure differently from the Global Log Level setting. The log level that you select for a specific service takes precedence over the Global Log Level setting.
Global Log Level	From the drop-down list, select the level of the log messages that you want to collect for every running Paragon Insights service for the device group. The level is set to error by default.				
Log Level for specific services	Select the log level from the drop-down list for any specific service that you want to configure differently from the Global Log Level setting. The log level that you select for a specific service takes precedence over the Global Log Level setting.				

Table 8: Fields in Device Group Configuration *(Continued)*

Attributes	Description
Publish	<p>You can configure Paragon Insights to publish sensor and field data for a specific device group:</p> <p>Destinations Select the publishing profiles that define the notification type requirements (such as authentication parameters) for publishing the data.</p> <p>To edit or view details about saved publishing profiles, go to the System page under the Settings menu in the left-nav bar. The publishing profiles are listed under Notification Settings.</p> <p>Field Select the Paragon Insights rule topic and rule name pairs that contain the field data you want to publish.</p> <p>Sensor (Device group only) Select the sensor paths or YAML tables that contain the sensor data you want to publish. No sensor data is published by default.</p>
Outbound SSH	
Ports	Enter one or more port numbers for outbound SSH connections for this device group.
IFA Deploy Nodes	Enter IP address of the node where the IFA ingest must be deployed in Paragon Insights.
IFA Ports	<p>Enter the UDP port number on which Paragon Insights receives IFA sensor data.</p> <p>Range: 1 to 65,535.</p>
Root Cause Analysis	
RCA Support	RCA is enabled by default. Disable RCA if you do not want the device group to be a part of root cause analysis.
Exclude Resources	Exclude Resources field allows you to select device resources that must be excluded from resource and dependency formation.

Editing a Device Group

To edit a device group:

1. Click the **Configuration > Device Group** option in the left-nav bar.
2. Click on the device group name under **DEVICE GROUPS**.
3. Click on the **Pencil** (Edit Device Group) icon.
4. Modify the attributes, as needed.

See ["Adding a Device Group" on page 129](#) for a description of each attribute.

5. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration. For information on how to use the device group cards, see ["Monitor Device and Network Health" on page 172](#).
6. (Optional) A device group can be deleted by clicking the **Trash Can** (Delete Device Group) icon with the device group selected.

Configuring a Retention Policy for the Time Series Database

To configure a retention policy for the time series data used for root cause analysis (RCA):

1. Click the **Settings > System** option in the left-nav bar.
2. Select **Retention Policy Settings**.
3. Click the **+ Retention Policy** button.
4. Enter the necessary values in the text boxes for the retention policy.

The following table describes the attributes in the **Add a Retention Policy** window:

Attributes	Description
Name	Name of the retention policy.

(Continued)

Attributes	Description
Duration	<p>Amount of time the root cause analysis (RCA) data is retained in the Paragon Insights RCA database. By default, data is retained for 7 days.</p> <p>The data must be entered in hours or days. For example, 1 day is entered as 1d or 24h.</p>

- Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration. You can now apply the retention policy to a device group. For information on how to apply a retention policy to a device group, see ["Adding a Device Group" on page 129](#).

Adding a Network Group

To add a network group:

- Click the **Configuration > Network** option in the left-nav bar.
- Click the + (Add Network) button.
- Enter the necessary values in the text boxes and select the appropriate options for the network group.

The following table describes the attributes in the **Add a Network Group** window:

Attributes	Description
Name	Name of the network group. (Required)
Description	Description for the network group.

(Continued)

Attributes	Description
Reports	<p>In the Reports field, select one or more health report profile names from the drop-down list to generate reports for the network group. Reports include alarm statistics, device health data, as well as device-specific information (such as hardware and software specifications).</p> <p>To edit or view details about saved health report profiles, go to the System page under the Settings menu in the left-nav bar. The report profiles are listed under Report Settings.</p> <p>For more information, see "Alerts and Notifications" on page 252.</p>
Disable Trigger Action Scheduler (for a particular network group)	<p>By default, this field is marked False because the option to add a UDA scheduler in Trigger Action page is enabled.</p> <p>You can set this field to True if you want to disable UDA scheduler settings in Trigger Action page.</p>
Notifications	<ul style="list-style-type: none"> You can use the Alarm Manager feature to organize, track, and manage KPI alarm notifications received from Paragon Insights devices. To receive Paragon Insights alarm notifications for KPI events that have occurred on your devices, you must first configure the notification delivery method for each KPI event severity level (Major, Minor, and Normal). Select the delivery method from the drop-down lists. <p>To edit or view details about saved delivery method profiles, go to the System page under the Settings menu in the left-nav bar. The delivery method profiles are listed under Notification Settings.</p> <p>For more information, see "Alerts and Notifications" on page 252.</p>
Ingest Frequency	<p>Select existing Ingest Frequency Profiles to override rule or sensor frequency settings.</p>

(Continued)

Attributes	Description
Logging Configuration	<p>You can collect different severity levels of logs for the running Paragon Insights services of a network group. Paragon Insights Release 4.0.0 supports collecting log data for SNMP notification.</p> <p>Use these fields to configure which log levels to collect:</p> <p>Global Log Level From the drop-down list, select the level of the log messages that you want to collect for every running Paragon Insights service for the network group. The level is set to error by default.</p> <p>Log Level for specific services Select the log level from the drop-down list for any specific service that you want to configure differently from the Global Log Level setting. The log level that you select for a specific service takes precedence over the Global Log Level setting.</p> <p>For more information, see "Logs for Paragon Insights Services" on page 309.</p>
Publish	<p>You can configure Paragon Insights to publish Paragon Insights sensor and field data for a specific network group:</p> <p>Destinations Select the publishing profiles that define the notification type requirements (such as authentication parameters) for publishing the data.</p> <p>To edit or view details about saved publishing profiles, go to the System page under the Settings menu in the left-nav bar. The publishing profiles are listed under Notification Settings.</p> <p>Field Select the Paragon Insights rule topic and rule name pairs that contain the field data you want to publish.</p>
Root Cause Analysis	
RCA Support	RCA is enabled by default. Disable RCA if you do not want the network group to be a part of root cause analysis.
Exclude Resources	Exclude Resources field allows you to select network resources that must be excluded from resource and dependency formation.

4. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.
For information on how to use the network, see ["Monitor Device and Network Health" on page 172](#).

Editing a Network Group

To edit a network group:

1. Click the **Configuration > Network** option in the left-nav bar.
2. Click anywhere on the line that contains the group name in the table under **NETWORK LIST**.
3. Click on the **Edit Network (Pencil)** icon.
4. Modify the attributes, as needed.
See ["Adding a Network Group" on page 137](#) for a description of each attribute.
5. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.
For information on how to use the network group cards, see ["Monitor Device and Network Health" on page 172](#).
6. (Optional) A network can be deleted by clicking the **Delete Network (Trash Can)** icon.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.0.0	Starting in Paragon Insights Release 4.0.0, you can add more than 50 devices per device group.
4.0.0	Paragon Insights Release 4.0.0 supports collecting log data for SNMP notification.
4.0.0	Paragon Insights Release 4.0.0 supports collecting log data for SNMP notification.

RELATED DOCUMENTATION

[Paragon Insights Rules and Playbooks | 141](#)

[Monitor Device and Network Health | 172](#)

Paragon Insights Rules and Playbooks

IN THIS SECTION

- [Add a Pre-Defined Rule | 141](#)
- [Create a New Rule Using the Paragon Insights GUI | 142](#)
- [Edit a Rule | 159](#)
- [Add a Pre-Defined Playbook | 159](#)
- [Create a New Playbook Using the Paragon Insights GUI | 160](#)
- [Edit a Playbook | 161](#)
- [Clone a Playbook | 162](#)
- [Manage Playbook Instances | 163](#)

The device or network performance elements that are important to one company may not be important to another. Paragon Insights (formerly HealthBot) uses Rules and Playbooks to define key performance indicators (KPIs) and organize them into groups that are applied to network devices.

This document presents the tasks involved in creating, editing, and deleting Paragon Insights rules and playbooks.

Add a Pre-Defined Rule

Juniper has created a set of pre-defined rules that you can use to gather information from various Juniper components and the networks they reside in. You can add these rules to Paragon Insights at any time. After installation, many default pre-defined rules appear in the Rules and Playbooks pages. Pre-defined rules cannot be changed or removed; however, you can clone any rule (pre-defined or user defined) simply by clicking the **CLONE** button on the upper-right part of the rule definition. A cloned rule goes to the *external*/topic and can be re-configured at will.

To upload additional pre-defined rules to Paragon Insights:

1. Using a browser, go to <https://github.com/Juniper/healthbot-rules> and download the pre-defined rule file to your system.
2. In the Paragon Insights GUI, click the **Configuration > Rules** icon in the left-nav bar.

- 3. Click the **↑ Upload Rule Files** button.
- 4. Click the **Choose Files** button.
- 5. Navigate to the rule file and click **Open**.
- 6. Select one of the following options:

Upload	Upload the file and save the rule within the defined topic area but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time.
Upload & Deploy	Upload the file, save the rule within the defined topic area, and immediately deploy the configuration.

NOTE: You can use the **Upload** and **Upload & Deploy** to upload user-defined action (UDAs) and user-defined function (UDFs) files.

Create a New Rule Using the Paragon Insights GUI

IN THIS SECTION

- Rule Filtering | 144
- Sensors | 145
- Fields | 147
- Vectors | 149
- Variables | 151
- Functions | 152
- Triggers | 154
- Rule Properties | 156
- Pre/Post Action | 157

To create a new rule using the Paragon Insights GUI, you'll first fill out general descriptive information about the rule and then navigate through several rule definition blocks in the Rules page to provide the specific configuration for the Paragon Insights rule.

To start creating a new Paragon Insights rule:

1. Click the **Configuration > Rules** icon in the left-nav bar. A list of Paragon Insights rules organized by Paragon Insights topic is displayed along the left side of the Rules page.
2. Click the add rule button (+ **Add Rule**).
3. Enter general descriptive information about the rule using the following input parameters:

Parameter	Description
Rule	<p>For a new rule, this parameter is pre-populated with <i>external / user_rule_random_characters</i>, for example, <i>external / user_rule_2p0ghk</i>. The fields separated by the slash (/) represent Paragon Insights topic name and Paragon Insights rule name, respectively.</p> <p><i>external</i> is the topic name used for user-defined topics. For the Paragon Insights rules pre-defined by Juniper, Juniper has curated a set of pre-defined device component-based topic names. For more information about Paragon Insights topics, see Paragon Insights Topics.</p> <p>Replace the <i>user_rule_random_characters</i> rule name with a name that appropriately represents the rule's description such as packets-in, packets-out, system_memory, etc.</p>
Rule frequency	(Network rule only) Specify how often data for the network rule is collected by Paragon Insights. This setting is overridden if the rule is included in a frequency profile that is applied to a network group.
Description	(Optional) Enter a detailed description for the rule.
Synopsis	(Optional) Enter a brief description for the rule. The synopsis is displayed when you hover over the rule name listed along the left side of the Rules page.
Field Aggregation Time Range	This optional value defines how often Paragon Insights aggregates the data received by the sensor. This helps reduce the number of data point entries in the time series database.

- 4. (Network rule only) If the new rule is a network rule, toggle the Network rule switch to the right.
- 5. Configure the rule definition blocks as needed.

Located directly below the Synopsis input parameter, you'll find links to the following rule definition blocks: **Sensors**, **Fields**, **Vectors**, **Variables**, **Functions**, **Triggers**, and **Rule Properties**. The following sections describe the input parameters for each of these rule definition blocks.

- 6. Select one of the following options to save the new rule:

Save	Save the rule within the defined topic area but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time.
Save & Deploy	Immediately deploy the configuration and save the rule within the defined topic area.

The following sections describe the input parameters for each of the Paragon Insights rule definition blocks:

Rule Filtering

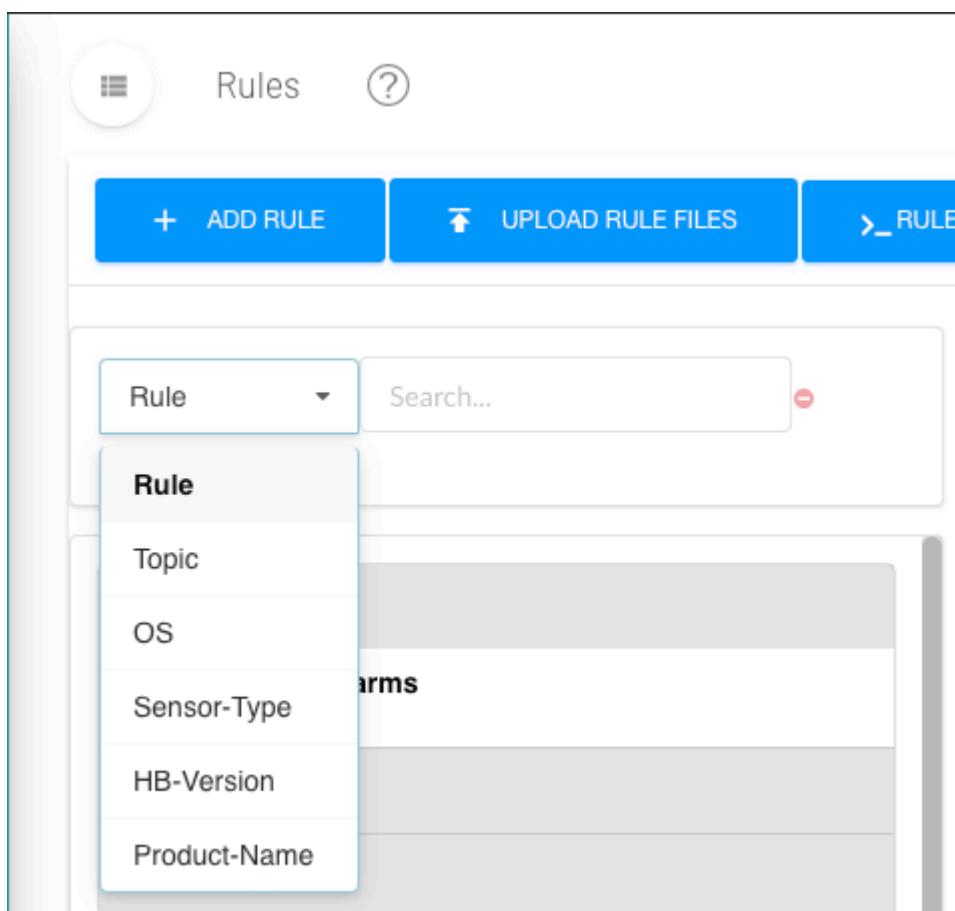
Starting in HealthBot Release 3.0.0, you can filter the Topics and Rules displayed on the left side of the Rules page. This allows you to quickly find rules that you are looking for. The search function works for topics, rules, sensor-types and other categories; working not only on titles, but also on the defined contents of rules.

The following procedure explains this filtering feature.

- 1. Navigate to **Configuration > Rules** in the left-nav bar.

The **Rules** page is displayed. To the left of the rule definition area is a new section as shown in [Figure 39 on page 145](#) below.

Figure 39: Rule Filtering



2. From the pull-down menu, select the type of search you want to perform.
3. In the search field, begin entering your search text.

The topic list below shrinks to display only topics and rules that match your search criteria.

Sensors

Start configuring the new rule using the **Sensor** block. [Figure 40 on page 146](#) shows the sensor definition for the OpenConfig sensor `pppoe-error-statistics`.

Figure 40: A Sensor Definition

Rule: / Rule Frequency: ☐ Network Rule

Description: Collects the PPPoE error periodically and notifies in case of anomalies

Synopsis: Monitors PPPoE error statistics

Sensors Fields Vectors Variables Functions Triggers Rule Properties

+ Add sensor

pppoe-error-statistics

Sensor Name ?

pppoe-error-statistics

Sensor type

Open Config

Sensor path *

/junos/system/subscriber-management/client-protocols/ppp

Frequency *

60s

1. Click the add sensor button (+ Add Sensor).

A new sensor definition appears and is named **Sensor_***random characters*, like Sensor_2kgf04.

2. Change the sensor name to something that makes sense for the rule you are defining.
3. From the drop-down list, choose the sensor type. You can choose one of: **OpenConfig**, **Native GPB**, **iAgent**, **SNMP**, **Syslog**, or **NetFlow**.

The required elements for defining the **Sensor Type** change depending on the selection you make. The frequency is expressed in #s, #m, #h, #d, #w, #y, or #o where # is a number and s, m, h, d, w, y specifies seconds, minutes, hours, days, weeks, years, and offset respectively. The o expression is used for defining an offset multiplier for use in formulas, references, triggers, learning periods, and hold-times.

The following list describes the elements that change based on your choice of **Sensor Type**. None of the other rule elements change because of a **Sensor Type** selection.

- **OpenConfig** **Sensor path** is defined from a drop-down list of available OpenConfig sensors. **Frequency** refers to how often, in seconds, the sensor reports to Paragon Insights. The frequency can be overridden if the sensor is included in a frequency profile.
- **Native GPB** **Sensor path** refers to the path for a Native GPB sensor. **Port** refers to the GPB port over which the sensor communicates with Paragon Insights.

- **iAgent** **File** is the name of a YAML-formatted file that defines the NETCONF-accessible sensor. **Table** is defined from a drop-down list of available PyEZ tables and views in the YAML file. **Frequency** refers to how often the sensor is polled by Paragon Insights and can be overridden by including the sensor in a frequency profile.

Based on the table you've selected, input fields for a target or dynamic arguments might also be provided. For these additional fields, you can do one of the following:

- Leave the input field blank. No default value will be applied.
- Enter a fixed value that will remain constant.
- Enter a variable name enclosed in double curly/flower brackets (for example, `{{test-variable}}`) The variable name must belong to a variable that was previously defined in the Paragon Insights rule, and the variable's **Type** option must be set to **Sensor Argument**.
- **SNMP** **Table** is defined from a drop-down list of available SNMP tables. **Frequency** refers to how often, in seconds, Paragon Insights polls the device for data and can be overridden by including the sensor in a frequency profile.
- **Syslog** **Pattern set** is a user-configured element that includes one or more patterns (you configure a pattern for each event you want to monitor). The **Maximum hold period** is used in advanced cases and refers to the maximum time that the system will wait when correlating events using multiple patterns within a pattern set.

NOTE: The syslog sensor requires some pre-configuration. See [Syslog Ingest](#) for more details.

- **Flow** **Template Name** is a Juniper-supplied built-in list of NetFlow v9 and IPFIX templates.

Fields

A sensor will generally carry information about multiple things. For example, a sensor that watches interfaces has information about all of the interfaces on the device. In Paragon Insights, we call these things Fields. Now that you've defined a sensor, you'll need to tell Paragon Insights which fields you're interested in.

1. Click the **Fields** link.

The screen updates and shows the defined field objects.

2. Click the add field button (+ **Add Field**).
3. Replace the random field name with a name that make sense for the rule you are defining, such as **interface-name**, **configured-threshold**, etc.
4. (Optional) Add descriptive text for the new field.
5. Set the appropriate **Field Type**. The options for field type are: string, integer, float, and unsigned integer. String is the default field type.

Starting in Paragon Insights Release 4.2.0, you can also select unsigned integer as a field type. An unsigned integer is a data type that can contain values from 0 through 4,294,967,295.

6. (Optional) Toggle the **Add to rule key** switch.

The add rule to key switch tells Paragon Insights that this field should be indexed and searchable. For example, when you enable this switch, the field name will be listed on the Devices page under the **Keys** column.

7. Select the appropriate ingest type (Field source) from the pull-down menu.

The following list shows the options available for the **Ingest type (Field source)** menu.

- **Sensor**—Use this or another sensor definition.
 - **Path**—Follow this Open Config or Netconf path within the sensor definition to gather specific data like the names of the interfaces. For iAgent sensors the **Path** refers to the path defined in the YAML file.
 - **Where**—Filter the available data to gather information about a specific element within, like a specific interface. This field can reference the **Variables** defined elsewhere within the rule. When referencing variables, use moustache notation, enclosed in slashes, such as: **{{interface_name}}**.
 - **Zero suppression**—For some sensors associated with devices running Junos OS, such as Junos Telemetry Interface Open Config and native GPB sensors, no field data is sent from the sensor when the data's value is zero. Enable the zero suppression switch to set the field data value to zero whenever no field data is sent from the sensor.
 - **Data if missing**—Specify a value as the default data value whenever no data is sent from the sensor. The format of the specified value should match the defined field type (string, integer, or float). If the zero suppression switch is also enabled, then the specified data-if-missing value is ignored, and the default sensor data value is set to zero.
- **Reference**—A reference to a field or trigger value from another rule.

- **Data if missing**—Specify a value as the default data value whenever no reference data is fetched. The format of the specified value should match the defined field type (string, integer, or float).
 - **Constant**—Use a constant when referring to a **Variable** defined within the rule, whose value doesn't change, such as **IO_Drops_Threshold**. A constant can also be a string or numeric value that does not change and is not a reference to a variable.
 - **Constant value**—Use moustache notation to reference the variable like this: `{{IO_Drops_Threshold}}`.
 - **Formula**—Select the desired mathematical formula from the **Formula** pull-down menu.
8. (Optional) Set the **Field aggregation time-range**. Located above the Fields tab with the general rule parameters, this periodic aggregation setting helps to reduce the number of data points entered in the database. For example, if the sensor settings specify that Paragon Insights ingests data every 10 seconds, you could configure this setting to aggregate and record the relevant field data, say, every 60 seconds. Note that when using this setting, any field-specific time ranges must use the same value.

NOTE: Starting in HealthBot Release 3.1.0, you can add fields and keys to rules based on whether the incoming data meets user-defined conditions defined in tagging profiles. Tagging profiles are defined in Paragon Insights under **Settings > Ingest Settings** on the left-nav. See ["Paragon Insights Tagging" on page 33](#) for details.

Vectors

(Optional) Now that you have a sensor and fields defined for your rule, you can define vectors.

A vector is used when a single field has multiple values or when you receive a value from another field.

1. Click on the **Vectors** link. [Figure 41 on page 150](#) shows the Vectors block for a newly added vector.

Figure 41: Vectors Block

SensorsFields**Vectors**VariablesFunctionsTriggersRule Properties

+ Add vector

vector_sd19e3

Vector name* ?

vector_sd19e3

Ingest Type

path

List of fields ?

Select fields

subscriber-count-maximum

subscriber-count-minimum

total-subscriber-count

total-subscribers-pred

Time-range ?

7d

- 2. Click the add vector button (+ Add Vector)
- 3. Replace the random vector name with a name that makes sense for your rule.
- 4. Select an ingest type from the drop-down list. The additional input fields will vary depending on the selection you make.

For path:

Parameter	Description
List of fields	Select a field from the drop-down list. The list of fields is derived from all of the defined fields in this rule.
Time-range	Specify a time range from which the data should be collected. The time range is expressed in #s, #m, #h, #d, #w, #y where # is a number and s, m, h, d, w, y specifies seconds, minutes, hours, days, weeks, and years respectively. For example, enter 7 days as 7d.

For formula:

Parameter	Description
Formula Type	<p>Select a formula type from the drop-down list:</p> <p>unique Creates a vector with unique elements from another vector.</p> <p>and Compares two vectors and returns a vector with elements common to both vectors.</p> <p>or Compares two vectors and returns a vector with elements from both vectors.</p> <p>unless Compares two vectors and returns a vector with elements from the left vector but not the right vector.</p>
Vector name	(Unique formula type only) Select a vector name from the drop-down list. The list of vectors is derived from all of the defined vectors in this rule.
Left vector	Select a vector name from the drop-down list. The list of vectors is derived from all of the defined vectors in this rule.
Right vector	Select a vector name from the drop-down list. The list of vectors is derived from all of the defined vectors in this rule.

Variables

(Optional) The **Variables** block is where you define the parts of the sensor that you are interested in. For example, a rule that monitors interface throughput needs to have a way to identify specific interfaces from the list of available interfaces on a device. The field details are discussed below.

1. Click on the **Variables** tab.
2. Click the add variable button (+ **Add Variable**)
3. Replace the random **Variable name** with a variable name that makes sense for your rule, such as **pem-power-usage-threshold**.

BEST PRACTICE: The accepted convention within Juniper for naming of elements within Paragon Insights is to always start with a lower-case letter and use hyphens to separate

words. Make sure that your variable names are unique, clearly named, and follow a recognizable pattern so that others can understand what the variable is for. Any abbreviations should be used consistently throughout.

4. Set an appropriate default value in the **Default value** field.

Default values vary depending on field type. Integer field types use numeric default values, while string field types use strings to set exact defaults and regular expressions that allow you to set the default from a list. Any default values set during rule definition can be overridden at apply-time at either the device or device group level.

5. Select the appropriate variable type from the **Type** drop-down list.

Available field types are: Integer, Floating Point, String, Boolean, Device, Device Group, and Unsigned Integer.

Starting in Paragon Insights Release 4.2.0, you can also select unsigned integer as a variable type. An unsigned integer is a data type that can contain values from 0 through 4,294,967,295.

Functions

(Optional) Define any needed functions.

The **Functions** block allows users to create functions in a python file and reference the methods that are available in that file. The python file must be created outside of Paragon Insights. You must know about the method names and any arguments because you will need those when defining the functions.

Starting with Paragon Insights 4.2.0 release, you can use a Python user-defined function to return multiple values.

For example, consider that you have a function `example_function.py` that has three return values. When you call the `example_function.py` in a rule, the first return value in the user-defined functions (UDF) is stored in the rule field that calls the function. You only need to configure return fields, such as `r2` and `r3`, for the remaining two return values. You can configure these fields for return values in the Return List of the Functions tab.

In the time-series database, the name of Return List fields are prefixed with the name of the rule field that uses the UDF. For example, `rule_field_name-r2`.

[Figure 42 on page 153](#) shows the **Functions** block for the `chassis.power/check-pem-power-usage` rule. The field details are discussed below.

Figure 42: The Functions Block

To configure a function:

1. Click on the **Functions** link.
2. Click **+ Add Function**.
3. Enter a function name such as **used-percentage**.
4. In the **Path to function** field, enter the name of the python file that contains the functions. These files must be stored in the **/var/local/healthbot/input/hb-default** directory. The pull-down list is populated with all the Python (.py) files in that directory.
5. In the **Method Name** field, enter the name of the method as defined in the python file. For example, **used_percentage**.
6. (Optional) Enter a description for the function in the description box.
7. (Optional) For each argument that the python function can take, click the add argument button (**+ Add Argument**).

Each time you click the add argument button, you'll need to enter the name of the argument and set the toggle switch as to whether the argument is mandatory or not. The default is that none of the arguments are mandatory.

8. (Optional) If there are more than one return value in the function, click **+Add Return List**.

9. Enter a name for the return value and the data type, such as an integer.

Triggers

A required element of rule definition that you'll need to set is the trigger element. [Figure 43 on page 154](#) shows the **Triggers** block for the `system.memory/check-system-memory` rule. The field details are discussed below.

Figure 43: The Triggers Block

The screenshot displays the 'Triggers' configuration panel within a rule editor. At the top, a navigation bar includes 'Sensors', 'Fields', 'Vectors', 'Variables', 'Functions', 'Triggers' (selected), and 'Rule Properties'. On the left, a sidebar shows a list of triggers, with 're-memory-buffer-utilization' selected. The main area contains the configuration for this trigger. The 'Trigger name' is 're-memory-buffer-utilization'. The 'Frequency' is set to 'Enter trigger frequency'. The 'Term' section is expanded, showing a 'WHEN' condition: '\$re-memory-buffer' >= '\$re-memory-buffer-high-threshold' with a 'Time range' of '5m'. Below this is a 'THEN' section with a 'Color' dropdown set to red, a 'Message' field containing 'Srouting-engine memory buffer utilization(\$re-memory-buffer) exceed high threshold(\$re-memory-buffer-high-threshold)', and an 'Evaluate next term' toggle. A blue information box at the bottom states: 'Functions can be used as Trigger actions too, define them using the "Functions" menu at the top.'

Setting up triggers involves creating terms that are used to set policy. If the terms within a trigger are matched, then some action is taken. Terms can evaluate the fields, functions, variables, etc that are defined within the rule against each other or in search of specific values. Terms are evaluated in order from the top of the term list to the bottom. If no match is found, then the next term (if any) is evaluated until a match is found or until the bottom of the term stack is reached.

1. Click on the **Triggers** link.
2. Click on the add trigger button (+ **Add Trigger**).

3. Replace the random trigger name with one that makes sense for the trigger you are defining, such as **foo-link-operation-state**. We recommend using a name that is very unique to the rule and trigger to avoid having the same trigger name across two or more rules.
4. (Optional) Enter a value in the **Frequency** field. This value tells Paragon Insights how often the field data and triggers should be queried and evaluated. If no entry is made here, then the sensor frequency is applied for this value. The frequency entered here can be entered as a multiple or, an offset, of the sensor frequency such as 2o. For example, if the sensor frequency is 10s and the trigger frequency is 2o, then the trigger frequency would be 20s (2*10s).
5. Click the add term button (+ **Add Term**).

The Term area will expand and show an add condition button, (+ **Add Condition**) in the **When** section and **Color** and **Message** fields in the **Then** section.

6. To define a condition that the term will evaluate, click the + **Add Condition** button.

The **When** section expands to show **Left operand**, **Operator**, and **Time range** fields.

NOTE: Setting a condition is not required. If you want to guarantee that a **Term** takes a specific action, don't set a condition. This could be useful, for example, at the bottom of a term stack if you want some indication that none of the terms in your trigger matched.

7. Select values from the pull-down menus for each of these fields.

Depending on which **Operator** is chosen, a new field, **Right operand** may appear in between the **Operator** and **Time range** fields.

The left and right operand pull-down menus are populated with the fields and variables defined in the rule. The operator field determines what kind of comparison is done. The time range field allows the trigger to evaluate things such as if there were any dropped packets in the last minute.

8. (Optional) Set values for the **Color** and **Message** fields, and add **Action Engine** information in the **Then** section.

These fields are the action fields. If a match is made in the condition set within the same term, then whatever action you define here is taken. A color value of green, yellow, or red can be set. A message can also be set and is not dependent on whether any color is set.

Starting in Paragon Insights Release 4.2.0, you link action workflows (Action Engine) to rules while setting up triggers. You can also add input arguments while linking action workflows. The **Action Engine** section is enabled only with a PIN-Advanced license. For more information, see "[Paragon Insights Licensing Overview](#)" on page 326.

If color or message are set, a toggle button labelled **Evaluate next term** appears at the bottom of the **Then** section. The default value for this button is off (not active).

NOTE: If no match is made in the **When** section of a term, the **Then** section is ignored. If this happens, the next term down, if any, is evaluated.

If a match is made in the **When** section, then the actions in the **Then** section, if any, are taken and processing of terms stops unless the **Evaluate next term** button is set to on (active).

Setting the **Evaluate next term** button allows you to have Paragon Insights make more complex evaluations like 'if one condition and another condition are both true, then take this action'.

Rule Properties

(Optional) Specify metadata for your Paragon Insights rule in the **Rule Properties** block. Available options include:

Attributes	Description
Version	Enter the version of the Paragon Insights rule.
Contributor	Choose an option from the drop-down list.
Author	Specify a valid e-mail address.
Date	Choose a date from the pop-up calendar.
Supported Paragon Insights Version	Specify the earliest Paragon Insights release for which the rule is valid.
Supported Device > Juniper Devices	<p>Choose either Junos or Junos Evolved. Device metadata includes Product Name, Release Name, Release Support (drop-down list), and platform. You can add metadata for multiple devices, multiple products per device, and multiple releases per product.</p> <p>Starting in Paragon Insights Release 4.2.0, you can select default sensors that you want to apply to all supported devices. You also have the option to select default sensors that you want to apply to Juniper devices, and to Juniper devices that run a specific OS.</p>

(Continued)

Attributes	Description
Supported Device > Other Vendor Devices	<p>Paragon Insights Release 4.1.0 and earlier—You can add metadata for multiple devices.</p> <p>Paragon Insights Release 4.2.0 and later—You can add vendor identifier, vendor name, product, platform, release, and operating system-related information for non-Juniper vendors.</p> <p>Starting in Paragon Insights Release 4.2.0, you can select default sensors that you want to apply to all non-Juniper devices.</p>
Helper Files	Specify files that are required by the Paragon Insights rule.

For more information on rules and telemetry sensors supported in a Junos OS and Junos OS Evolved software release for a specific hardware platform, see [Telemetry Sensor Explorer](#).

Pre/Post Action

You can use the following general workflow to work with pre- or post-action tasks:

1. Configure Action Engine Workflows. See ["Manage Action Engine Workflows" on page 244](#) for more information.
2. Configure Pre-Action or Post-Action tasks depending on your use case.

See ["Pre-action Tasks" on page 157](#) to configure pre-action tasks and see ["Post-action Tasks" on page 158](#) to configure post-action tasks.
3. Create a playbook with rules that have pre or post-action tasks. See ["Paragon Insights Rules and Playbooks" on page 141](#) for more information.
4. Run an instance of the new playbook on device groups to execute pre-action tasks. See ["Paragon Insights Rules and Playbooks" on page 141](#) for more information.
5. Stop the instance of the new playbook to execute post-action tasks. See ["Paragon Insights Rules and Playbooks" on page 141](#) for more information.
6. Monitor the status of pre-action and post-action tasks. See ["Manage Action Engine Workflows" on page 244](#) for more information.

Before you configure pre-action and post-action tasks in rules, you must configure Action Engine workflows. See ["Manage Action Engine Workflows" on page 244](#) to configure Action Engine workflows.

To configure pre-action tasks:

1. Click the Pre/Post Actions tab on the Rules page.
2. In the Pre-Action section, select an action engine workflow in the Action Engine field.
3. Enter the input argument from the list in the device field. The list shows arguments you previously configured in the selected action engine workflow as the pre-action task.
4. Enable execute-once if you want Paragon Insights to execute the pre-action task only once on each device in a device group.
5. Do one of the following:
 - Click **Save** to save your configuration changes but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time later. See ["Commit or Roll Back Configuration Changes in Paragon Insights" on page 307](#) for more information.
 - Click **Save and Deploy** to save the rule configuration in the GUI and deploy the configuration. When the rule is included in a playbook and a playbook instance runs on a device group, Paragon Insights executes the pre-action task parallel to ingesting telemetry data.

Before you configure pre-action and post-action tasks in rules, you must configure Action Engine workflows. See ["Manage Action Engine Workflows" on page 244](#) to configure Action Engine workflows.

To configure post-action tasks:

1. Click the Pre/Post Actions tab on the Rules page.
2. In the Post-Action section, select an action engine workflow in the Action Engine field.
3. Enter the input argument from the list in the device field. The list shows arguments you previously configured in the selected action engine workflow as the post-action task.
4. Enable execute-once if you want Paragon Insights to execute the post-action task only once on each device in a device group.
5. Do one of the following:
 - Click **Save** to save your configuration changes but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time later. See ["Commit or Roll Back Configuration Changes in Paragon Insights" on page 307](#) for more information.
 - Click **Save and Deploy** to save the rule configuration in the GUI and deploy the configuration. When you stop the playbook instance (with this rule) from running on a device group, Paragon Insights executes the post-action task after it stops the service for playbooks.

Edit a Rule

To edit a rule:

1. Click the **Rules** icon in the left-nav bar.
2. Click the name of the rule listed along the left side of the Rules page.
3. Modify the necessary fields.
4. Select one of the following options:

Save	Save your edits but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time.
Save & Deploy	Immediately deploy the configuration and save the rule.

5. (Optional) Delete a rule by clicking the **Delete** button located to the right of the rule name.

Add a Pre-Defined Playbook

Juniper curates a set of pre-defined playbooks designed to address common use cases. You can add these playbooks to your Paragon Insights installation at any time. The default pre-defined playbooks cannot be changed or removed.

To add a pre-defined playbook to Paragon Insights:

1. Using a browser, go to <https://github.com/Juniper/healthbot-rules> and download the pre-defined playbook file to your computer.
2. In the Paragon Insights GUI, click the **Configuration > Playbooks** icon in the left-nav bar.
3. Click the upload playbook button (↑ **Upload Playbook**).
4. Click the **Choose Files** button.
5. Navigate to the playbook file and click **Open**.
6. Select one of the following options:

Upload	Upload the file and add the playbook but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time.
---------------	---

Upload & Deploy

Upload the file, add the playbook, and immediately deploy the configuration.

NOTE: You can use the **Upload** and **Upload & Deploy** to upload user-defined action (UDAs) and user-defined function (UDFs) files.

Create a New Playbook Using the Paragon Insights GUI

Paragon Insights operates on playbooks, which are a collection of rules for solving a specific customer use case. For example, the system-kpi-playbook monitors the health of system parameters such as system-cpu-load-average, storage, system-memory, process-memory, etc. and notifies the operator or takes corrective action in case any of the KPIs cross pre-set thresholds. Any single rule can be a part of 0, 1, or more playbooks. The playbook is the rule element that gets deployed on devices. Rules that are not included in any playbook will not be deployed to any device.

NOTE: Click the **Name**, **Running**, **Paused**, or **Synopsis** column headers in the Playbooks table to organize the data in ascending or descending order.

Starting with Release 4.2.0, Paragon Insights supports splat operator in ICMP outlier detection playbook. The **icmp-outlier** playbook earlier required that you enter the round trip time (RTT) XPath for each device (using device id). As the playbook is applied to all devices in a device group, you can enter the splat operator (*) in the regular expression format instead of device id of each device.

For example:

```
/device-group[device-group-name=core]/device[device-id=~/.*/*]]
/topic[topic-name=protocol.icmp]/rule[rule-name=check-icmp-statistics]/rtt-average-ms
```

NOTE: If you delete or add a device in a device group on which an ICMP outlier playbook instance runs, you must pause the playbook instance, modify the XPath configuration to reflect the addition or deletion of devices, and re-apply the playbook instance for the device group.

To create a new playbook using the Paragon Insights GUI:

1. Click the **Configuration > Playbooks** icon in the left-nav bar.

2. Click the create playbook button (+ **Create Playbook**).

A new window appears with 4 fields: Name, Synopsis, Description, and Rules. We describe the use of each field below.

3. Enter a name for the playbook in the **Name** field.

4. Enter a short description for the playbook in the **Synopsis** field.

This text appears in the Synopsis column of the table on the **Playbooks** page.

5. (Optional) Enter a description of each of the rules that make up this playbook in the **Description** field.

This text can only be seen if you click on the playbook name on the **Playbooks** page.

6. From the rules drop-down list, select the rules that make up this playbook.

7. Select one of the following options:

Save	Save and add the playbook but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time.
Save & Deploy	Immediately deploy the configuration and save and add the playbook.

Edit a Playbook

To edit a playbook:

1. Click the **Configuration > Playbooks** icon in the left-nav bar.

2. Click the name of the playbook.

3. Modify the necessary text boxes.

4. Select one of the following options:

Save	Save your edits to the playbook but do not deploy the updated configuration. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time.
Save & Deploy	Immediately deploy the configuration and save your edits to the playbook.

5. (Optional) Delete a playbook by clicking the trash can icon in the **Delete** column.

NOTE: You cannot edit or delete a system defined (Juniper provided) playbook.

When you update a playbook, the new changes on the playbook will not be applied to the existing instances of the playbook. For example, a playbook instance that is associated to a device group will not be updated when the playbook is edited or updated. You must delete the existing playbook instance and create a new one for updates to be applied.

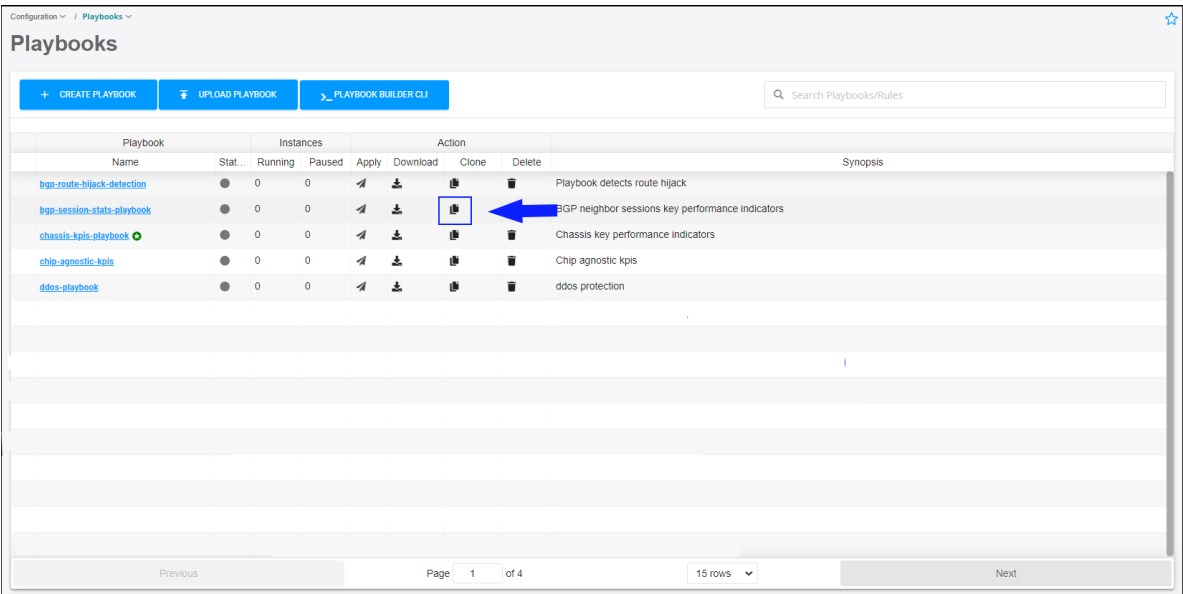
Clone a Playbook

Starting with HealthBot Release 3.2.0, you can clone an existing playbook and modify configurations.

Follow these steps to clone a playbook by using the Paragon Insights UI.

- 1. Click **Configuration>Playbooks** in the left-nav bar.
The Playbooks page is displayed.
- 2. Click the **Clone** icon as show in [Figure 44 on page 162](#).

Figure 44: Clone a Playbook



The Clone Playbook: *<name of playbook>* page is displayed.

You can now edit **Name**, **Synopsis**, and **Description** sections. You can also add new rules or remove existing rules from the Rules drop-down list.

3. Click **Save** to save configurations and confirm cloning the playbook.

Alternatively, click **Save & Deploy** to save configurations, confirm cloning the playbook, and deploy the newly cloned playbook.

Manage Playbook Instances

IN THIS SECTION

- [View Information About Playbook Instances | 163](#)
- [Create a Playbook Instance | 166](#)
- [Manually Pause or Play a Playbook Instance | 167](#)
- [Create a Schedule to Automatically Play/Pause a Playbook Instance | 169](#)

The term *playbook instance* refers to a specific snapshot of a playbook that is applied to a specific device group or network group. You can manually play and pause playbook instances. Alternatively, you can apply a customized schedule to a playbook instance that will automatically perform play and pause actions.

The following sections describe tasks that you can perform to manage playbook instances:

View Information About Playbook Instances

To view information about playbook instances:

1. Click the **Configuration > Playbooks** option in the left-nav bar.

The saved playbooks are listed in the table on the main Playbooks page.

Playbooks

+ CREATE PLAYBOOK

📁 UPLOAD PLAYBOOK

🔧 PLAYBOOK BUILDER CLI

🔍 Search Playbooks/Rules

Playbook	Instances		Action			Synopsis
	Name	Running	Paused	Apply	Live	
bgp-route-hijack-detection	0	0	🚀	●	🗑️	Playbook detects route hijack
bgp-session-stats-playbook 🟢	0	0	🚀	●	🗑️	BGP neighbor sessions key performance indicators
chassis-kpis-playbook 🟢	0	0	🚀	●	🗑️	Chassis key performance indicators
chip-agnostic-kpis	0	0	🚀	●	🗑️	Chip agnostic kpis
DHCP-server-statistics 🟢	0	0	🚀	●	🗑️	DHCP Local Server and Relay Statistics KPIs
dot1x-user-authentication-kpis	0	0	🚀	●	🗑️	dot1x authentication KPI
evpn-irb-icmp-probe	0	0	🚀	●	🗑️	EVPN-VXLAN irb key performance indicators
evpn-vxlan-kpis 🟢	0	0	🚀	●	🗑️	EVPN-VXLAN key performance indicators
forwarding-table-summary 🟢	0	0	🚀	●	🗑️	Forwarding table and protocol routes key performance indicators
get-vpn-stats Requires advanced licensing	0	0	🚀	●	🗑️	Interface and routing instance collector
icmp-outlier 🟢	0	0	🚀	●	🗑️	ICMP outlier detector
icmp-probe 🟢	0	0	🚀	●	🗑️	ICMP RTT response checker
▶ interface-kpis-playbook	1	0	🚀	●	🗑️	Interface key performance indicators
interface-optical-kpis	0	0	🚀	●	🗑️	Optical interface key performance indicators
isis-stats-playbook	0	0	🚀	●	🗑️	ISIS adjacency key performance indicators

Previous

Page 1 of 3

15 rows

Next

- Playbooks that have been applied to a device group or network group are identified in the table by a right caret next to the playbook name.
- The Instances column in the table shows the number of playbook instances running and paused.
- Starting with HealthBot Release 3.1.0, some playbooks require the purchase and installation of advanced or premium licenses. These playbooks are identified by the green circle with a white star in it. As shown above, it tells you which license is required when you hover your mouse over the icon.
- The Live column (in the Action section of the table) shows a colored circle indicator that represents the overall status of the playbook instances for each playbook. The following table provides the color definitions:

Table 9: Color Definitions for the Live Column

Color	Definition
Green	All instances associated with this playbook are currently running.
Yellow	One or more instances associated with this playbook are paused.
Gray/Black	There are no instances associated with this playbook, or an instance is associated with this playbook but the configuration has not been deployed yet.

- Click on the caret next to the playbook name to expand or collapse the playbook instance details. If no caret is present, then the playbook has not been applied to any device groups or network groups.

The following playbook instance details are displayed:

Column Name or Widget	Description
Instance Name	User-defined instance name.
Schedule	<p>Name of the schedule profile applied to the playbook instance. For information on how to configure a schedule profile, see "Create a Schedule to Automatically Play/Pause a Playbook Instance" on page 169</p> <p>Click on the name to display the schedule details.</p>
Device/Network Group	Device group or network group to which the schedule is applied.
No. of devices	Number of devices on which this playbook instance is deployed. This is applicable for device group instances only, not for network group instances.
Status	<p>Current status of the playbook instance. Status can be either Running or Paused. The Status column also indicates whether the action was performed automatically or manually.</p> <p>Note: If the status of a playbook instance is Running (automatic), you can manually pause the schedule for this instance using the Pause Schedule button. In this case, the status will change to Paused (manual). To resume running the schedule for this instance, you must manually run the instance using the Play Schedule button. In this case, the status will change back to Running (automatic), and the play and pause actions associated with the schedule will resume.</p>
Started/Paused at	Date and time when the playbook instance was last started or paused. The date reflects local browser time zone.
Next Action	This column applies only to playbook instances associated with a schedule. It indicates whether the playbook instance is scheduled to automatically pause or play in the future. This column is blank if no schedule is associated with the playbook instance or if the status of the instance is Paused (manual) .

(Continued)

Column Name or Widget	Description
Play/Pause button	<p>Pauses or plays a playbook instance or the schedule for a playbook instance.</p> <p>The Play/Pause button toggles between the two states. For more information, see "Manually Pause or Play a Playbook Instance" on page 167.</p>
Trash can icon	Deletes the playbook instance.

Create a Playbook Instance

To create a playbook instance for a device group:

1. Click the **Configuration > Playbooks** option in the left-nav bar.
2. Click the **Apply** icon (in the Action section of the table) for the desired playbook.

A pane titled *Run Playbook: <playbook-name>* appears.

3. In the **Name of Playbook Instance** field, fill in an appropriate name for this instance of the playbook. This is a required field.
4. (Optional) In the **Run on schedule** field, choose the name of the schedule that you want to apply to this playbook instance. You can apply only one schedule per playbook instance. If you want a specific playbook instance to run on multiple schedules, you must create multiple versions of the instance, each with its own unique name and schedule.

For information on how to configure a schedule, see ["Create a Schedule to Automatically Play/Pause a Playbook Instance" on page 169](#).

To view information about an existing schedule:

- a. Click the **Settings** option in the left-nav bar.
 - b. In the **Scheduler Settings** section, a summary of the properties for each saved schedule is shown in the table. Click on a specific schedule name to view additional details.
5. In the **Device Group** section under **Rules**, apply this playbook instance to the appropriate device group using the drop-down list.

The list of devices in the **Devices** section changes based on the device group selected.

NOTE: If your playbook contains network rules, the **Device Group** section does not appear. Instead, it is replaced with a **Network Groups** section (not shown).

6. Click one of the devices listed in the **Devices** section.

Here is where you can customize the variable values that will be set for this device when the playbook is run.

7. In the section titled **Variable values for Device <Device Name>**, the variables for each rule in the playbook can be seen by clicking on the rule name. The default values for each variable are displayed as grey text in each field. You can leave these values as-is or override them by entering a new value.

Repeat steps 6 and 7 for each device in the device group, as needed.

8. When you are satisfied that all of the variable values are appropriate for all the devices in the device group, select one of the following options.

Save Instance	Save your edits but do not deploy the updated configuration and do not run the instance. You can use this option when, for example, you are making several changes and want to deploy all your updates at the same time.
Run Instance	Deploy the configuration, and, if no schedule profile was applied, immediately run the instance. If a schedule profile was applied, the instance will run according to the configuration of the profile.

Manually Pause or Play a Playbook Instance

When an instance is paused, Paragon Insights does not collect, analyze or raise an alarm on the device or network group data associated with the playbook rules. Data collected prior to pausing the instance is retained in the system, but no new data is collected or analyzed until the instance is played again.

The following table describes the state of the playbook instance when a particular button is displayed in the Play/Pause column:

If the displayed Play/Pause button is...	Then the state of the playbook instance is...
Pause Instance	<ul style="list-style-type: none"> • The instance is running. • The instance is not associated with a schedule. • Data is being collected.
Pause Schedule	<ul style="list-style-type: none"> • The instance is associated with a schedule. • The schedule is running. • The schedule determines when the instance is running.
Play Instance	<ul style="list-style-type: none"> • The instance is not running. • The instance is not associated with a schedule. • No data is being collected.
Play Schedule	<ul style="list-style-type: none"> • The instance is associated with a schedule. • The schedule and the instance is not running. • No data is being collected.

To manually pause a playbook instance or schedule:

1. Click the **Configuration > Playbooks** option in the left-nav bar.

The list of existing playbooks is displayed.

2. Click the caret next to the name of the playbook that you want to pause.
3. Choose one of the following options:
 - Click the **Pause Instance** button to pause a playbook instance (not associated with a schedule).
 - Click the **Pause Schedule** button to pause the schedule that's associated with a playbook instance.
4. The Play/Pause Playbook Instance dialog box appears. Select one of the following options:

Pause	Flags this playbook instance to be paused the next time you deploy the configuration. Use this option if you are making several changes and want to deploy all your edits at the same time.
Pause & Deploy	<p>Immediately pause the playbook instance and deploy the configuration. It will take a few seconds for the playbook table to update to show the instance is paused.</p> <p>There is a slight delay in updating the table because the play and pause actions are asynchronous and run in the background, allowing you to perform other actions. The status of this asynchronous activity can be tracked through the deploy icon located in the upper right corner of the window (as indicated in the success message of deploy action). Once this action is complete, the status is reflected in the playbook table as well.</p>

Once the playbook table is refreshed, the playbook name shows a yellow icon in the Live column as a visual indicator that an instance is paused.

5. To resume a paused playbook instance, follow the same steps as above except choose one of the following options for Step 3:
 - Click the **Play Instance** button to resume running a playbook instance (not associated with a schedule).
 - Click the **Play Schedule** button to resume running the schedule associated with a playbook instance. The schedule determines when the instance resumes playing.

Create a Schedule to Automatically Play/Pause a Playbook Instance

To automatically play/pause a playbook instance, you must first create a schedule and then apply the schedule to the playbook instance. You can apply only one schedule per playbook instance. If you want a specific playbook instance to run on multiple schedules, you must create multiple versions of the instance, each with its own unique name and schedule.

To create a schedule for a playbook instance:

1. Click the **Settings > System** option in the left-nav bar.
2. Click the **Scheduler** tab.
3. In **Scheduler Settings**, click the add scheduler button (+ **Scheduler**).
4. Enter the necessary values in the text boxes and select the appropriate options for the playbook instance schedule.

The following table describes the attributes in the **Add a scheduler** and **Edit a scheduler** panes:

Attributes	Description
Name	Enter the name of the playbook instance schedule.
Scheduler Type	<p>Choose discrete.</p> <p>You can configure a discrete length of time to play the playbook instance using the Run for field. Once the run time has ended, Paragon Insights will automatically pause the instance. You can also configure Paragon Insights to automatically resume playing the instance using the Repeat field.</p>
Start On	Use the pop-up calendar to select the date and time to play the playbook instance for the first time.
Run for	<p>Configure a discrete length of time to play the playbook instance. First enter an integer value and then choose the unit of measure (minute, hour, or day) from the drop-down list.</p> <p>Once the run time has ended, Paragon Insights will automatically pause the instance. You can also configure Paragon Insights to automatically resume playing the instance using the Repeat field.</p>
End On	<p>(Optional) Use the pop-up calendar to select the date and time to pause the playbook instance indefinitely. Leave blank if you want the playbook instance to play indefinitely.</p> <p>Note: If a playbook instance is associated with a schedule and is running when the End On time is reached, then the instance will continue to run until the configured Run for length of time is reached. At this time, the instance will pause indefinitely.</p>
Repeat	<p>Configure the Run for field before configuring the Repeat field. The Repeat interval must be larger than the configured Run for length of time.</p> <p>In the drop-down list, choose one of the following:</p> <ul style="list-style-type: none"> • The frequency (every day, week, month, or year) at which you want the playbook instance to play. • The Never option if you want the playbook to play only once. • The Custom option to specify a custom frequency at which you want the playbook instance to play. Use the Repeat Every field to configure the custom frequency.

(Continued)

Attributes	Description
Repeat Every	(Optional) If you chose the Custom option for the Repeat field, enter the custom frequency at which you want the playbook instance to play. First enter an integer value and then choose the unit of measure (minute, hour, or day) from the drop-down list.

5. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.
6. Now you're ready to apply the schedule to a playbook instance. For more information, see ["Create a Playbook Instance" on page 166](#).

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
3.1.0	Starting in HealthBot Release 3.1.0, you can add fields and keys to rules based on whether the incoming data meets user-defined conditions defined in tagging profiles.
3.1.0	Starting with HealthBot Release 3.1.0, some playbooks require the purchase and installation of advanced or premium licenses.
3.0.0	Starting in HealthBot Release 3.0.0, you can filter the Topics and Rules displayed on the left side of the Rules page.

RELATED DOCUMENTATION

[Paragon Insights Concepts | 6](#)

[Manage Devices, Device Groups, and Network Groups | 121](#)

[Telemetry Sensor Explorer](#)

Monitor Device and Network Health

IN THIS SECTION

- [Dashboard | 172](#)
- [Health | 183](#)
- [Network Health | 192](#)
- [Graphs Page | 192](#)

Paragon Insights (formerly HealthBot) offers several ways to detect and troubleshoot device-level and network-level health problems. Use the information provided by the following Paragon Insights GUI pages to investigate and discover the root cause of issues detected by Paragon Insights:

Dashboard

IN THIS SECTION

- [Device Dashlets | 174](#)
- [Device Group List Dashlet | 175](#)
- [Network Group List Dashlet | 177](#)
- [Health Alert Dashlets | 178](#)
- [TSDB Dashlets | 180](#)

Paragon Insights Release 4.0.0 introduces GUI enhancements brought about by the migration to a new framework. This change does not have an impact on existing functionality or resources in the GUI but introduces the following changes in the interface:

- **Favorites** option
- **Launchpad** icon

- TSDB (time series database) dashlets

The **Favorites** option, denoted by a star button at top right corner of all pages, allows users to mark pages under the **Favorites** section for easier access.

In the top right corner of the UI, if you click the **Launchpad** button (rocket icon), you get a drop down menu that takes you to the **Sizing Tool** and the Github repository for Paragon Insights rules called **Playbooks (github)**. **Sizing Tool** allows you to estimate the compute (vCPU), memory (RAM), and storage requirements to deploy or scale Paragon Insights in your network. Visit the [sizing tool app](#) to estimate your requirements.

Starting with Paragon Insights Release 4.0.0, the Alarms option is renamed as **Alerts**. To access the Alerts page, go to **Monitor > Alerts**.

Use the **Dashboard** to create a custom view of what you're most interested in. Paragon Insights pre-populates the dashboard with the **Device List**, **Device Group List**, and **Network Group List** dashlets and calls this view **My Dashboard**. You can create your own dashboard view by clicking the **+** to the right of **My Dashboard**. Custom views can be added, renamed, and deleted as you see fit.

The **Dashboard** also has a graphical list of pre-defined dashlets across the top that is initially hidden from view. Click the cluster of 9 blue dots on the upper right part of the page to display or hide the available dashlets. Each dashlet provides graphical information from a specific point of view. Many of the dashlets can be clicked on to drill down deeper into the information presented.

- **Devices** Consists of devices dashlet, device vendor dashlet, and device status dashlet (see "[Device Dashlets](#)" on page 174).
- **Device Groups** Consists of device group dashlets (see "[Device Group List Dashlet](#)" on page 175).
- **Network Groups** Consists of network group dashlets (see "[Network Group List Dashlet](#)" on page 177).
- **Health Alert** Consists of health alert dashlets (see "[Health Alert Dashlets](#)" on page 178).
- **TSDB (Time Series Database)** Consists of TSDB dashlets that have line charts for **Buffer Bytes**, **Buffer Length**, and bar charts for **Read Error for Last 5 Minutes**, **Write Error for Last 5 Minutes**, and **Buffer Length** (see "[TSDB Dashlets](#)" on page 180).

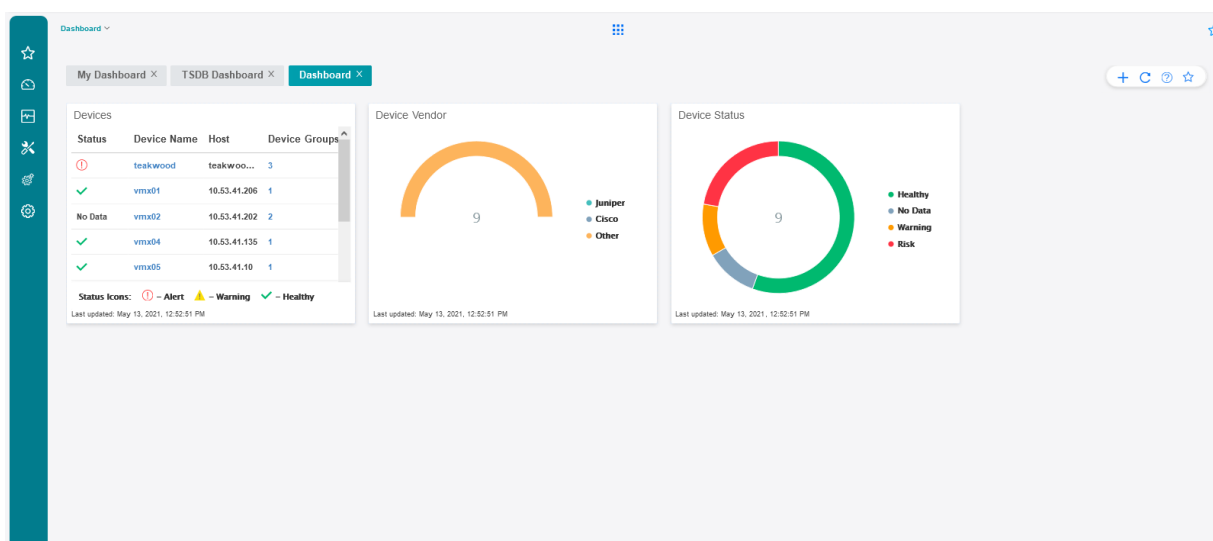
The Dashboard uses two types of colored objects to provide health status: halos and bars. The following table describes the meaning of the severity level colors displayed by the status halos and bars on the Dashboard:

Color	Definition
Green	The overall health of the device, device group, or network group is normal. No problems have been detected.
Yellow	There might be a problem with the health of the device, device group, or network group. A minor problem has been detected. Further investigation is required.
Red	The health of the device, device group, or network group is severe. A major problem has been detected.
Gray	No data is available.

Device Dashlets

The following table describes the main features of the Devices dashlet, Device Vendor dashlet, and Device Status dashlet on the dashboard.

Figure 45: Devices Dashlets

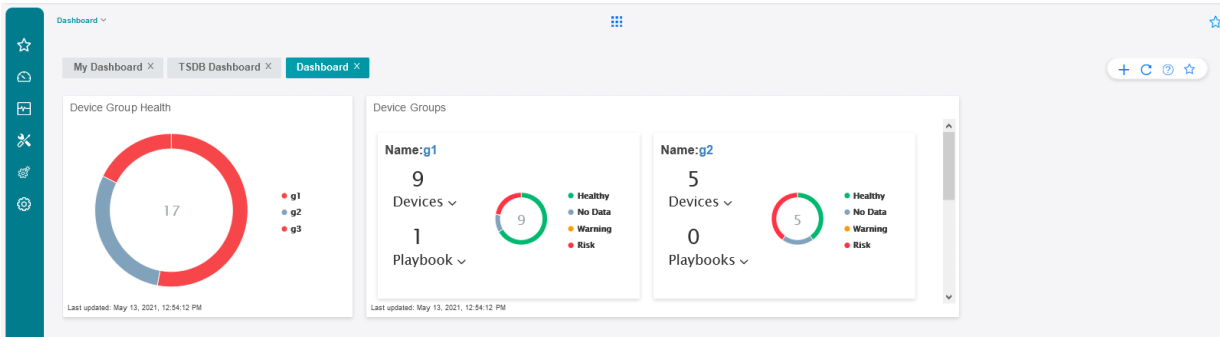


Dashlets	Description
Devices	<p>Devices dashlet lists the status and hostname of all devices configured in Paragon Insights.</p> <p>Click on the number under the Device Groups column in the dashlet to trace the device group to which the device is added. It goes to the Device Group Configuration page.</p> <p>Click the circular arrow at the top of the dashlet to refresh the dashlet.</p> <p>Click the X at the top of the dashlet to remove the dashlet from the dashboard.</p>
Device Vendor	<p>The half pie chart in this dashlet shows the total number of devices classified by vendor name. Each vendor is distinguished by a unique color.</p> <p>If you hover over the chart, the status halo displays the number of devices from a vendor and the percentage share of the devices in the total.</p> <p>The legend by the chart displays the names of the vendors. If you click on the name of a particular vendor, the devices from that vendor is filtered out from the data shown in the pie chart.</p>
Device Status	<p>The pie chart in this dashlet shows the total number of devices in the platform classified by health status. Each health status is distinguished by a unique color.</p> <p>If you hover over the chart, the status halo of each segment displays the number of devices with the health status denoted by the segment color and the percentage share of the devices in the total.</p> <p>The legend by the chart displays the health statuses of devices. If you click on the name of a particular status, the devices with that health status is filtered out from the pie chart.</p>

Device Group List Dashlet

The following table describes the main features of the Device Groups dashlet and Device Group Health dashlet on the dashboard.

Figure 46: Device Group Dashlets



Dashlets	Description
Device Groups	<p>To edit device group properties, click the device group name. For information on device group properties, see "Manage Devices, Device Groups, and Network Groups" on page 121.</p> <p>To display the list of devices that belong to a device group, click the integer number on this dashlet that represents the number of devices included in the device group.</p> <p>To display the list of playbooks applied on a device group, click the integer number that represents the number of playbooks applied on the device group.</p> <p>To remove this dashlet from the dashboard, click the X button at the top corner of the dashlet.</p> <p>The color of each segment in the pie chart represents the health status of the devices in the device group. For example, if a chart has one half segment as green and the other half segment as yellow, then no problems are detected in the number of devices displayed in the green segment and minor problems are detected in the number of devices displayed in the yellow segment.</p> <p>Clicking on a segment takes you to the Monitor > Device Configuration page.</p> <p>The coloring of the status halo in the pie chart segments represents the percentage of devices in the device group that have the health status defined by the color. For example, if the circle halo is all green, then the health of 100% of the devices in the device group is normal.</p> <p>The legend by the pie chart displays the different health statuses of device groups. If you click on the name of a particular status such as healthy, the device groups with that status are filtered out from the pie chart.</p>

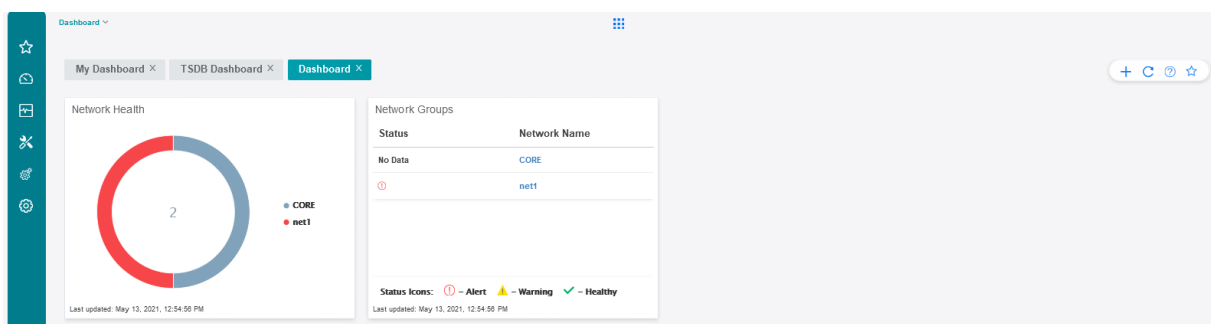
(Continued)

Dashlets	Description
Device Group Health	<p>The pie chart in this dashlet classifies all device groups in the platform by their health status. Each health status is distinguished by a unique color.</p> <p>If you hover over the chart, the status halo of each segment displays the number of devices in the device group with the health status denoted by the segment color and the percentage share of the devices in the total.</p> <p>The legend by the chart displays the device groups. If you click on the name of a particular device group, the devices in that device group are filtered out from the pie chart.</p>

Network Group List Dashlet

The following table describes the main features of the Network Groups dashlet and Network Health dashlet.

Figure 47: Network Group Dashlets

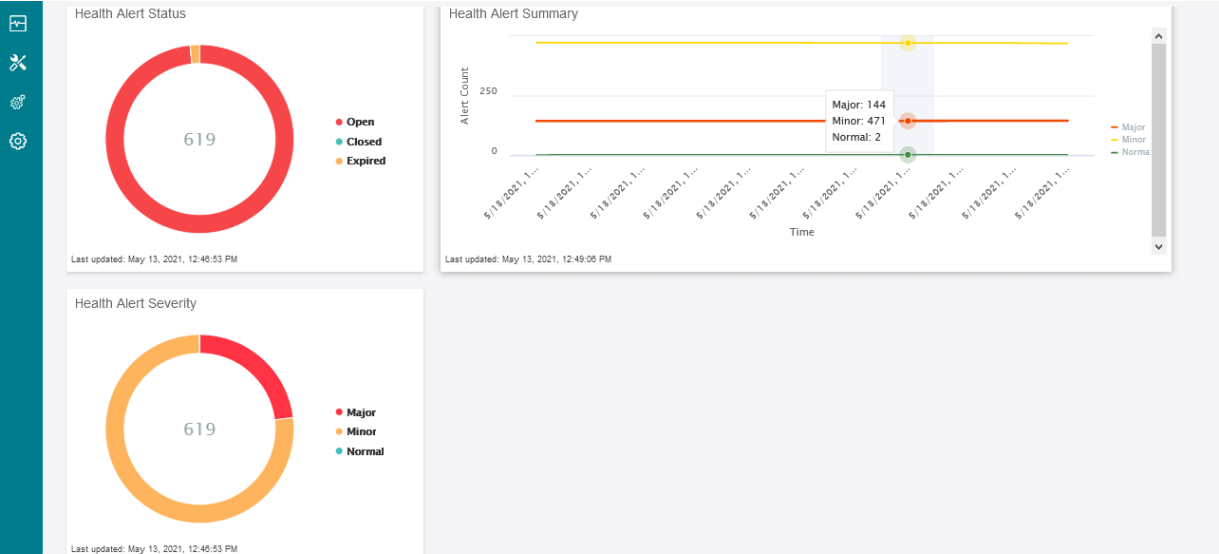


Dashlets	Description
Network Groups	<p>To edit network group properties, click the network group name. For information on network group properties, see "Manage Devices, Device Groups, and Network Groups" on page 121.</p> <p>Click the X at the top of the dashlet to remove the dashlet from the dashboard.</p> <p>Click the name of a network group in the dashlet to open the Network Health page of a particular network group.</p> <p>The status icons displayed against a network group name represents the overall health status of the network group. It can read no data or display an icon to indicate warning, alert, and healthy status.</p>
Network Health	<p>This dashlet shows a pie chart with several segments to classify the network groups based on their network health. The segment size denotes the number of network groups with the status of that segment. The segment color represents the overall health status of the network group(s) in that segment.</p> <p>For example, if you hover over a red segment in the chart, it displays the name(s) of the network group(s) with major alerts and the total count of network groups with major alerts.</p> <p>The legend by the pie chart displays the names of all network groups. If you click on the name of a particular network group, that group is filtered out from the pie chart.</p>

Health Alert Dashlets

The following table describes the properties of the Health Alert dashlets.

Figure 48: Health Alert Dashlets



Dashlets	Description
Health Alert Severity	<p>The pie chart in this dashlet shows the total number of health alerts generated in Paragon Insights classified by the alert severity level.</p> <p>Each segment of the pie chart represents a severity level distinguished by a unique color.</p> <p>If you hover your cursor over the chart, the status halo of each segment displays the number of alerts with the severity level denoted by the segment color and the percentage share of the alerts in the total.</p> <p>The legend by the chart displays the severity levels of the alerts. If you click on the name of a severity level, the alerts marked with that severity level are filtered out from the pie chart.</p>
Health Alert Status	<p>The pie chart in this dashlet shows alerts generated in Paragon Insights classified by their status in the platform: open, closed, and expired.</p> <p>Each segment of the pie chart represents a status distinguished by a unique color.</p> <p>If you hover your cursor over the chart, the status halo of each segment displays the number of alerts with the status denoted by the segment color and the percentage share of the alerts in the total.</p> <p>The legend by the chart displays the three main statuses of alerts. If you click on the name of an alert status, the alerts marked with that status are filtered out from the pie chart.</p>

(Continued)

Dashlets	Description
Health Alert Summary	<p>The chart in this dashlet shows a time line view of alerts generated in Paragon Insights classified by their severity levels.</p> <p>Each severity level in the chart is distinguished by a unique color.</p> <p>If you hover your cursor over any time point in the chart, the number of alerts per severity level is displayed for that time.</p> <p>The legend by the chart displays the severity levels of alerts. If you click on the name of an alert severity level, alerts marked with that severity are filtered out from the chart.</p>

TSDB Dashlets

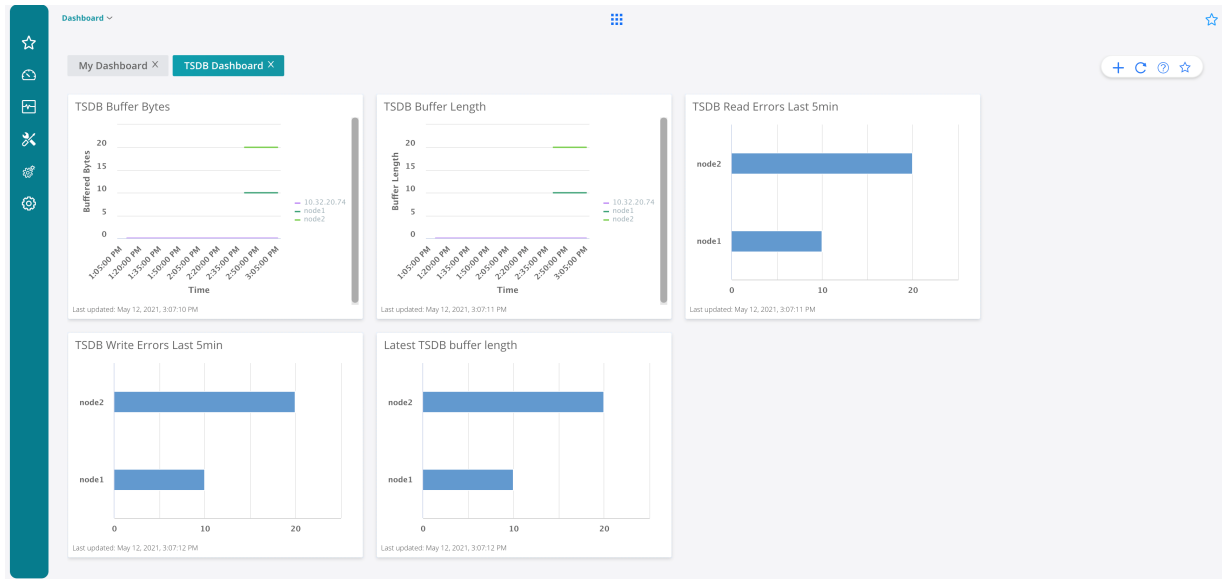
Each TSDB table row, that is also known as a point, contains data for a particular key at a given time. In TSDB, write or read requests are executed by grouping multiple points into a batch point.

Buffer Length denotes the number of batch points that are buffered per node at a given time. **Buffered Bytes** denotes the total size of the batch points buffered per TSDB node. In TSDB, a maximum of 1GB of batch point can be buffered per node.

Read or write errors occur when TSDB does not accept more requests as the buffer is full or because of issues in Kubernetes clusters. Errors can be minimized through database sharding. For more information, see ["Paragon Insights Time Series Database \(TSDB\)" on page 57](#).

[Figure 49 on page 181](#) shows a sample TSDB dashboard.

Figure 49: Paragon Insights TSDB Dashlets



The following table describes the main features of the time series database dashlet on the Dashboard. For information on how TSDB works, see ["Paragon Insights Time Series Database \(TSDB\)" on page 57](#).

Dashlets	Description
TSDB Buffer Bytes	<p>The chart in this dashlet shows a time line view of buffered bytes classified by the number of nodes.</p> <p>Each TSDB node in the chart is distinguished by a unique color.</p> <p>If you hover your cursor over any time point in the chart, the total buffered bytes per node for that time is displayed.</p> <p>The legend by the chart displays the nodes. If you click on the name of a node, buffered bytes data of that node is filtered out from the chart.</p> <p>When the chart refreshes, the vertical axis of buffered bytes is auto adjusted based on the data. The data in vertical axis is in bytes.</p> <p>To delete the dashlet, click the X at the top of the dashlet.</p>

(Continued)

Dashlets	Description
TSDB Buffer Length	<p>The chart in this dashlet shows a time line view of buffer length classified by the number of nodes.</p> <p>Each TSDB node in the chart is distinguished by a unique color.</p> <p>If you hover your cursor over any time point in the chart, the buffer length per node for that time is displayed.</p> <p>The legend by the chart displays the nodes. If you click on the name of a node, buffer length data of that node is filtered out from the chart.</p> <p>The vertical axis shows data of buffer length in terms of absolute number of batch points.</p> <p>To delete the dashlet, click the X at the top of the dashlet.</p>
TSDB Read Errors Last 5 Minutes	<p>The bar chart in this dashlet shows the number of TSDB read errors collected every 5 minutes classified by the number of nodes.</p> <p>The legend displays the nodes. If you click on the name of a node, the read error data of that node is filtered out from the chart.</p> <p>The horizontal axis shows read errors in absolute number.</p> <p>To delete the dashlet, click the X at the top of the dashlet.</p>
TSDB Write Errors Last 5 Minutes	<p>The bar chart in this dashlet shows the number of TSDB write errors collected every 5 minutes classified by the number of nodes.</p> <p>The legend displays the nodes. If you click on the name of a node, the write error data of that node is filtered out from the chart.</p> <p>The horizontal axis shows write errors in absolute number.</p> <p>To delete the dashlet, click the X at the top of the dashlet.</p>
Latest TSDB Buffer Length	<p>The bar chart in this dashlet shows the number of TSDB buffer length classified by the number of nodes.</p> <p>The legend displays the nodes. If you click on the name of a node, the buffer length data of that node is filtered out from the chart.</p> <p>The horizontal axis shows buffer length in absolute number.</p> <p>To delete the dashlet, click the X at the top of the dashlet.</p>

Health

IN THIS SECTION

- [Timeline View | 183](#)
- [Tile View | 185](#)
- [Table View | 187](#)
- [Time Inspector View | 190](#)

Use the Health page (**Monitor > Health**) to monitor and track the health of a single device, a device group, or a network. You can also troubleshoot problems. Select a device group using the entity type selectors (DEVICE, DEVICE GROUP, or NETWORK) located in the top left corner of the page. Once selected, you can then select individual devices or all of the devices from the group by clicking the **Select devices** pull-down menu. The page is divided into the following three main views that, when used together, can help you investigate the root cause of problems detected on your devices:

- ["Timeline View" on page 183](#)
- ["Tile View" on page 185](#)
- ["Table View" on page 187](#)
- ["Time Inspector View" on page 190](#)

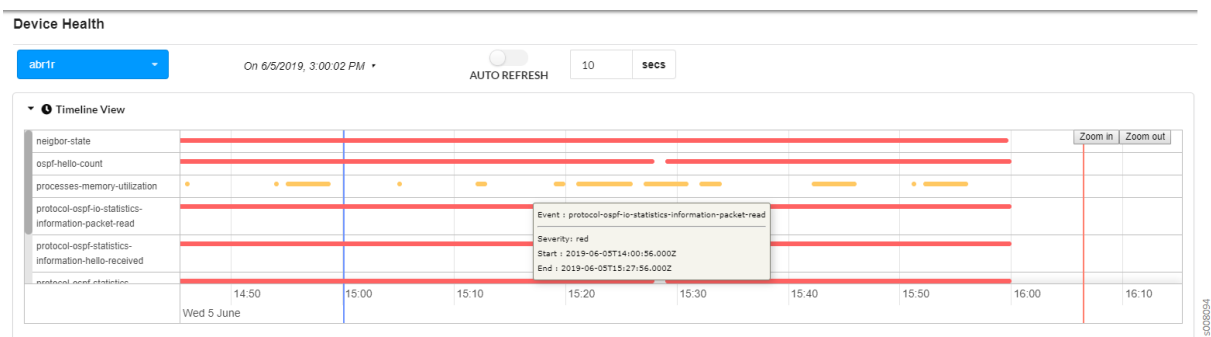
Timeline View

In timeline view, you can monitor real-time and past occurrences of KPI events flagged with a minor or major severity level health status. The general characteristics and behaviors of the timeline include (see [Figure 50 on page 184](#)):

- Clicking on the right caret next to the Timeline View heading expands or collapses the timeline.
- Each dot or line in the timeline represents the health status of a unique KPI event (also known as a Paragon Insights rule trigger) for a pre-defined KPI key with which Paragon Insights has detected a minor or major severity level issue. The name of each event is displayed (per device) directly to the left of its associated health status dot or line.
- The health status dot or line for each unique KPI event in the timeline can consist of several different KPI keys. Use tile view and table view to see the health status information for the individual KPI keys.

- Only minor or major severity level KPI events are displayed in the timeline. Yellow represents a minor event, and red represents a major one.
- A KPI event that occurs once (at only one point in time) and does not recur continuously over time is represented as a dot.
- A KPI event that occurs continuously over time is represented as a horizontal line.
- Timeline data is displayed for a 2-hour customizable time range.
- The red vertical line on the timeline represents the current time.
- The blue vertical line on the timeline represents the user-defined point of time for which to display data.

Figure 50: Timeline view



The following table describes the main features of the timeline:

Feature	Description
Display information about a dot or horizontal line in the timeline.	<p>Hover over the dot or horizontal line to display the associated KPI event name, device name, health status severity level, and event start and end times.</p> <p>Additional health status information about the KPI event can be found in tile view. For information about tile view, see the "Tile View" on page 185 section.</p>
For the displayed data, change the range of time (x-axis) that is visible on the page.	<p>Options:</p> <ul style="list-style-type: none"> • Click and drag the x-axis of the timeline to the left or to the right. • Click the Zoom In or Zoom Out buttons in the top right corner of the timeline.

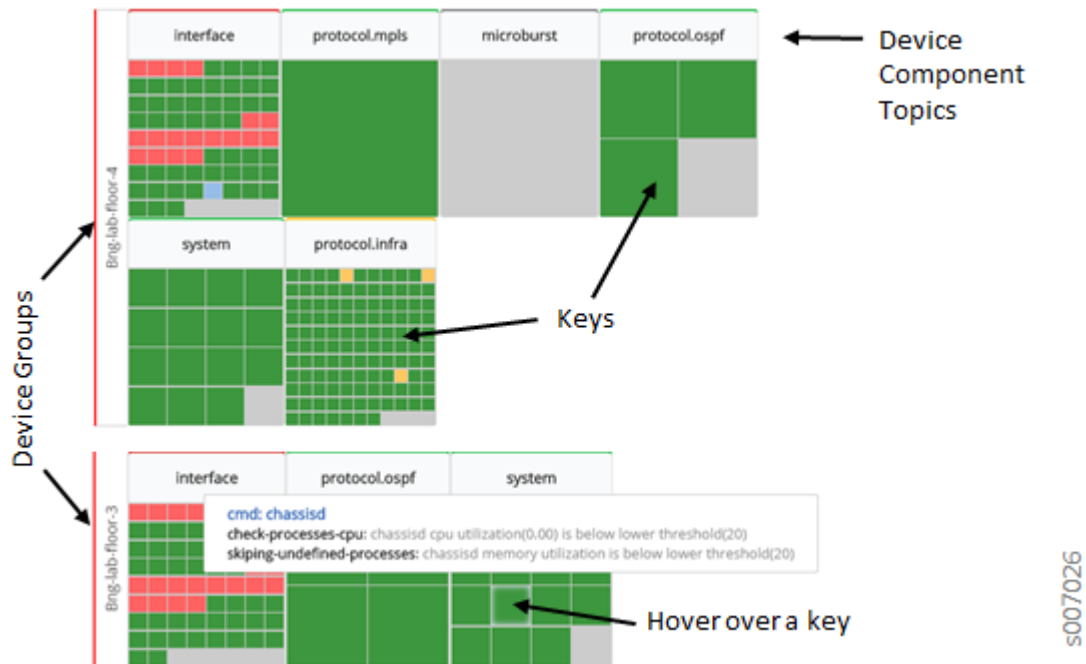
(Continued)

Feature	Description
Choose a different 2-hour time range of data to display.	<p>Use the blue vertical line to customize the time range of data to display. Options for enabling the blue vertical line:</p> <ul style="list-style-type: none"> • Click inside the timeline grid at the particular point in time you want to display data. • In the date/time drop-down menu (located above the timeline), select the particular point in time you want to display data .. <p>Data is generally displayed for 1 hour before and 1 hour after the blue line. Hover over the blue line to display the exact point in time that it represents. Drag the blue line left or right to adjust the time.</p> <p>NOTE: Auto-refresh is disabled whenever you enable the blue line. Re-enabling auto-refresh disables the blue line and resets the timeline to display the most recent 2-hour time range of data.</p>
Freeze the timeline (disable auto-refresh).	Toggle the auto-refresh switch to the left.
Unfreeze the timeline (enable auto-refresh).	Toggle the auto-refresh switch to the right.

Tile View

The tile view uses colored tiles to allow you to monitor and troubleshoot the health of a device. The tiles are organized first by device group, then by device component topic, and lastly by unique KPI key (see [Figure 51 on page 186](#)). By default, the tile view data corresponds to the most recent data collected. To customize the point in time for which data is displayed in tile view, select a particular point in time from the date/time drop-down menu (located above the timeline) or enable the blue vertical line in timeline view. For information about how to enable the blue vertical line, see the ["Timeline View" on page 183](#) section. The **Composite** toggle switch (not shown) at the upper right of the **TILE VIEW**, allows you to select data from more than one device component topic to be shown in the **Table View** and, thus, the Time Inspector View. This can be useful when topics must be combined to find root cause for an issue. For example, system memory usage could combine with output queue usage to create a performance issue in an overloaded system.

Figure 51: Tile View



The following table describes the meaning of the severity level colors displayed by the status tiles:

Color	Definition
Green	The overall health of the KPI key is normal. No problems have been detected.
Yellow	There might be a problem with the health of a KPI key. A minor problem has been detected. Further investigation is required.
Red	The health of a KPI key is severe. A major problem has been detected.
Gray	No data is available.

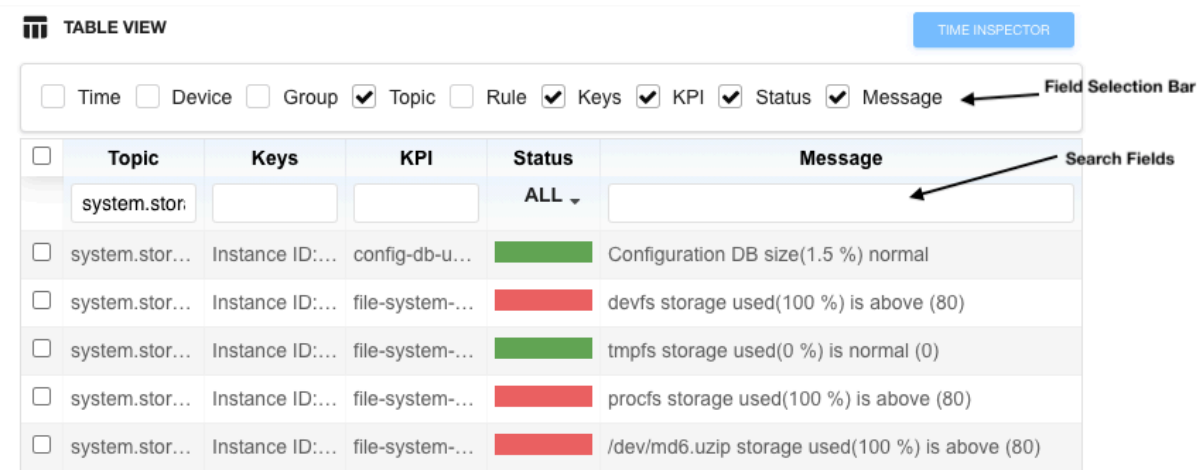
The following table describes the main features of the tile view:

Feature	Description
Display information about a status tile.	<p>Options:</p> <ul style="list-style-type: none"> • Hover over a status tile to display the name of the key, KPIs associated with the key, and the status messages associated with the KPIs. • Click on a status tile. Information about the status tile is displayed in table view. For information about table view, see the "Table View" on page 187 section. <p>Note: If the number of KPI keys exceeds 220, the keys are automatically aggregated and grouped.</p>
Display information in table view about the status tiles associated with a single device component topic.	Click on a device component topic name in tile view. For information about table view, see the "Table View" on page 187 section.
Composite Toggle	When active, users can click on specific keys within the tile groups. This allows you to pass multiple KPIs to the Time Inspector View.

Table View

The table view allows you to monitor and troubleshoot the health of a single device based on Paragon Insights data provided in a customizable table. You can search, sort, and filter the table data to find specific KPI information, which can be especially useful for large network deployments. To select which attributes are displayed in the table, check the appropriate check box in the field selection bar above the table (see [Figure 52 on page 188](#)). The checkbox on the left side of each row is used to help activate the Time Inspector view. Multiple rows can be selected at one time.

Figure 52: Table View



The following table describes the Paragon Insights attributes supported in table view:

Attributes	Description
Time	Time and date the event occurred.
Device	Device name.
Group	Device group name.
Topic	Rule topic name.
Keys	Unique KPI key name.
KPI	Key Performance Indicator (KPI) name associated with an event.
Status	Health status color. Each color represents a different severity level.
Message	Health status message.

The following table describes the meaning of the severity level colors displayed by the **Status** column:

Color	Definition
Green	The overall health of the KPI key is normal. No problems have been detected.
Yellow	There might be a problem with the health of a KPI key. A minor problem has been detected. Further investigation is required.
Red	The health of a KPI key is severe. A major problem has been detected.
Gray	No data is available.

The following table describes the main features of the table view:

Feature	Description
Sort the data by ascending or descending order based on a specific data type.	Click on the name of the data type at the top of the column by which you want to sort.
Filter the data in the table based on a keyword.	Enter the keyword in the text box under the name of a data type at the top of the table (see Figure 52 on page 188).
Navigate to a different page of the table.	Options: <ul style="list-style-type: none"> At the bottom of the table, click the Previous or Next buttons. At the bottom of the table, select the page number using the up/down arrows (or by manually entering the number) and then press Enter.
If the data in a cell is truncated, view all of the data in a cell.	Options: <ul style="list-style-type: none"> Hover over the cell. Resize the column width of the cell by dragging the right side of the title cell of the column to the right.
Row selection checkbox	Make this row's data available for Time Inspector view.

Time Inspector View

Time Inspector is a composite view that provides a timeline view of trigger conditions on KPI data that you selected in **Table View**. You can also drag and drop trigger conditions to view the conditions in one graph or as separate graphs. Time Inspector was initially available only when the entity type **DEVICE GROUP** is selected. However, starting with Paragon Insights Release 4.3.0, **Time Inspector View** is also available when you select **Device** or **Network** entity type. After you select an entity type, you can access time inspector view by clicking the **TIME INSPECTOR** button.

This view allows you to drill down into field-level data for specific triggers over a time line.

When the **Health** page is first accessed, the **TIME INSPECTOR** button is disabled. To activate the **TIME INSPECTOR** button, you must:

- Select the **Entity Type** from the **Health** page.

In releases earlier than Paragon Insights Release 4.3.0, **Time Inspector View** was available only when the entity type **DEVICE GROUP** is selected. However, starting with Paragon Insights Release 4.3.0, **Time Inspector View** is also available when you select **Device** or **Network** entity type.

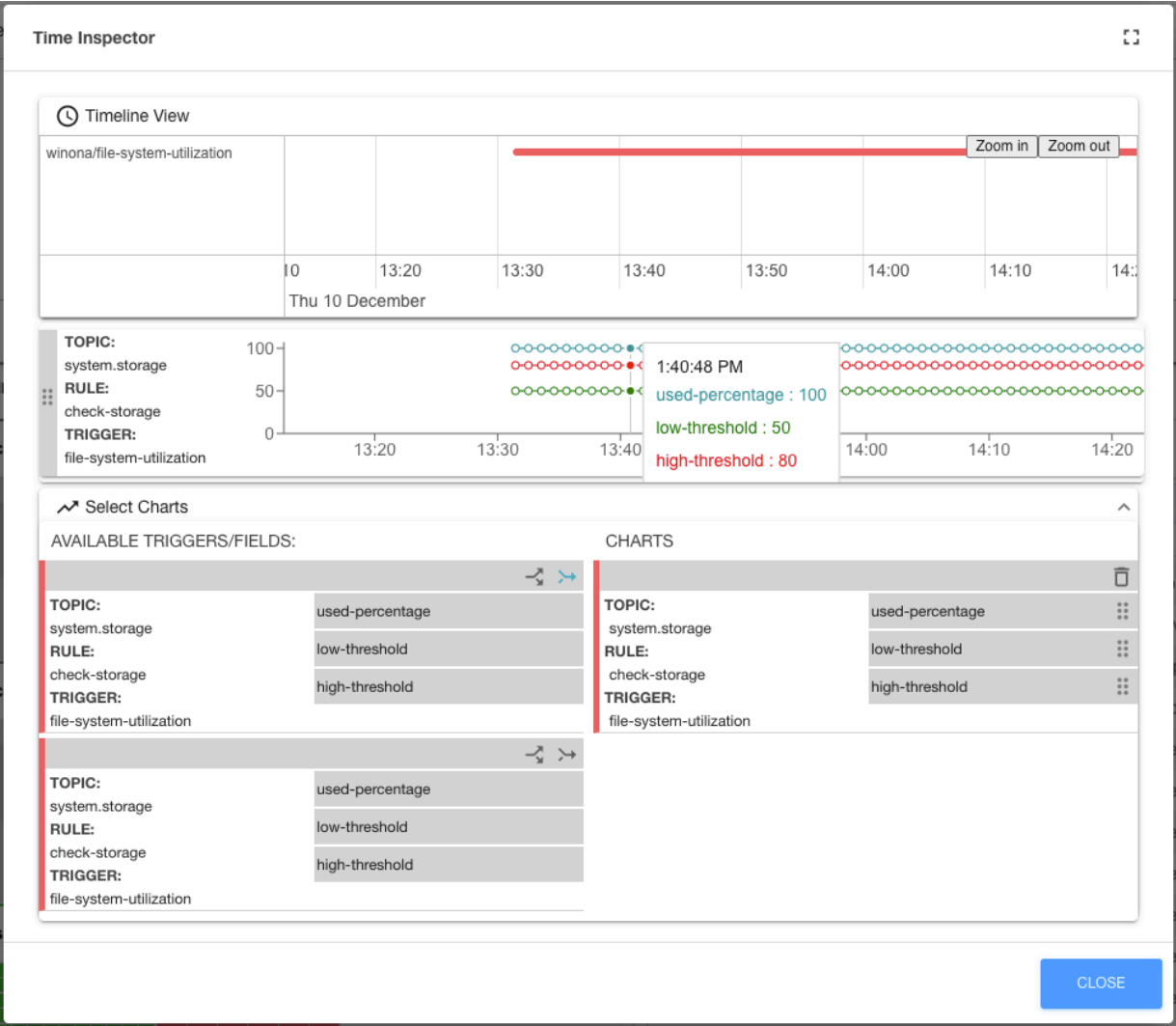
- Select one or more devices, device groups, or networks from the drop-down list next to the entity type that you have selected.
- Have valid data in at least one device, device group, or network component topic in **TILE VIEW**.

NOTE: Topics showing “no data” will not work for enabling the **Time Inspector** view.

- Have data appearing in the **TABLE VIEW** section. You can achieve this by clicking the device, device group, or network component topic header in **TILE VIEW**.
- Select the checkbox to the left of at least one of the rows in **TABLE VIEW**.

When clicked, the **TIME INSPECTOR** button opens a pop-up window above the Health page. [Figure 53 on page 191](#) below shows a time inspector window created from the system.storage usage topic for a specific device.

Figure 53: Time Inspector Window



As you can see, the Time Inspector window has a mini timeline at the top, an incremented line chart below, and a chart selector section at the bottom. This particular chart was created as a composite (indicated by the merging blue arrow) of a file-system-utilization in the check-storage rule of the system.storage topic.

Note that there are three fields in the check-storage rule: used-percentage, low-threshold, and high-threshold. Since the chart was created as a composite (fields charted together) there are three lines on the displayed chart. If the “chart fields separately” button (diverging arrows) were clicked instead, you would see 3 single-line charts showing the same data.

The more rules you select with the **TABLE VIEW** checkboxes, the more charts you can create in the **Time Inspector** view.

Network Health

Use the Network Health page (**Monitor > Network Health**) to monitor and track the health of a Network Group and troubleshoot problems. Select a Network Group using the drop-down list located in the top left corner of the page. Comparable to the Device Group Health page (see the ["Health" on page 183](#) section), the Network Health page is divided into three main views: timeline, tile, and table. The Network Health page provides similar features and functionality for a network group as the Device Group Health page provides for a single device.

Graphs Page

IN THIS SECTION

- [Graph Types | 195](#)
- [How to Create Graphs | 195](#)
- [Managing Graphs | 195](#)
- [Graph Tips and Tricks | 195](#)
- [Use Cases | 195](#)

You can use graphs to monitor the status and health of your network devices. Graphs allow you to visualize data collected by Paragon Insights from a device, showing the results of rule processing. Access the page from the left-nav panel **Monitor>Graphs>Charts**

NOTE: Graphs are refreshed every 60 seconds.

Figure 54: Example of Multiple Graph Panels on a Single Canvas



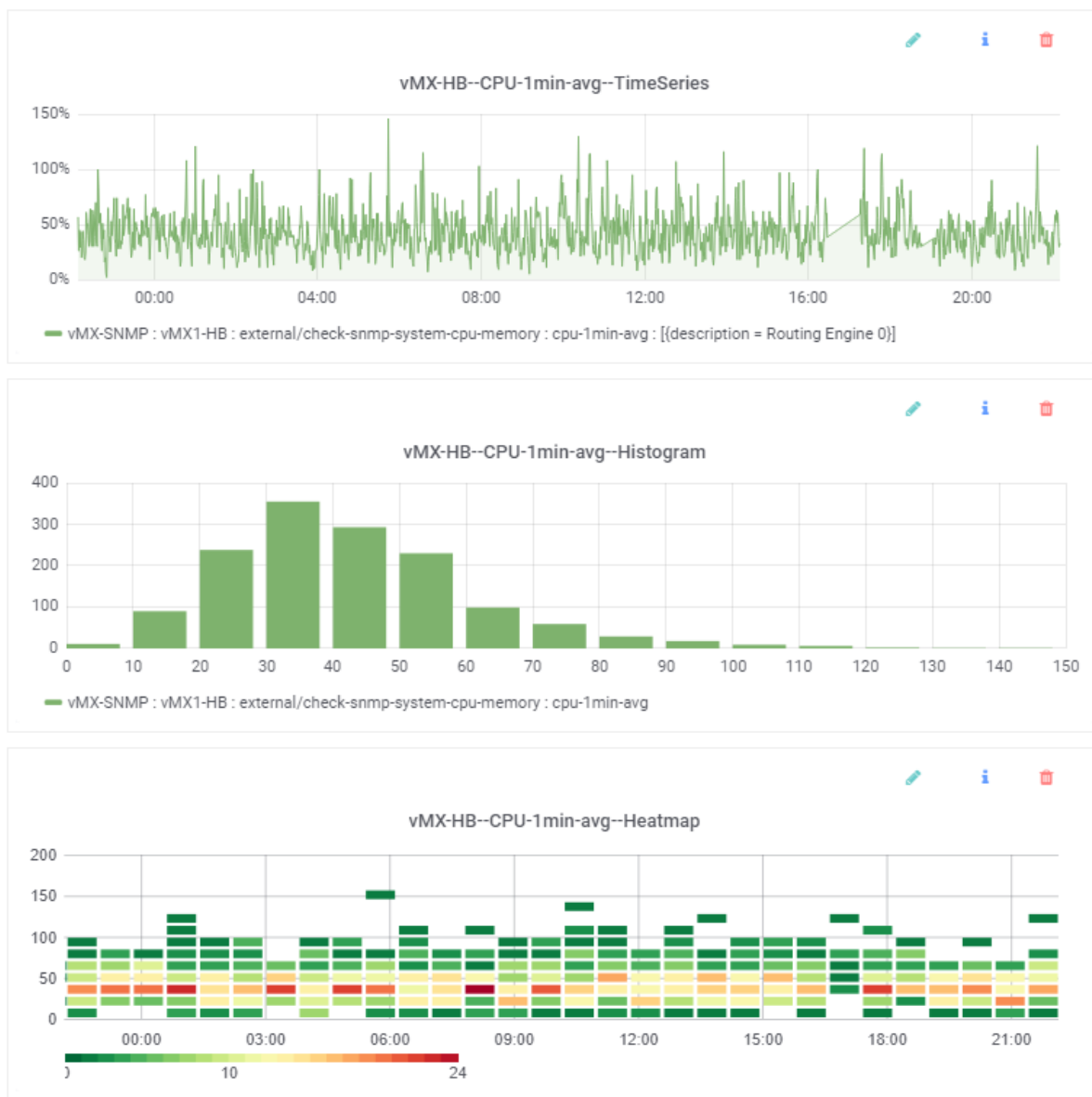
Graph types include time series graphs, histograms, and heatmaps.

Time series graphs are the kind you are used to, showing the data in a '2D' format where the x-axis indicates time while the y-axis indicates the value. Time series graphs are useful for real-time monitoring, and also to show historical patterns or trends. This graph type does not provide insight into whether a given value is 'good' or 'bad', it simply reports 'the latest value'.

Histograms work quite differently. Rather than show a continuous stream of data based on when each value occurred, histograms aggregate the data to show the *distribution* of the values over time. This results in a graph that shows 'how many instances of each value'. Histograms also show data in a '2D' format, however in this case the x-axis indicates the value while the y-axis indicates the number of instances of the given value.

Heatmaps bring together the elements above and provide a '3D' view to help determine the deviations in the data. Like a time series graph, the x-axis indicates time, while the y-axis indicates the value. Then the 'how many' aspect of a histogram is added in. Finally, the third dimension—*color*—is added. It is common to think of the colors as showing health, i.e., red means 'bad', yellow means 'OK', and green means 'good'. However, this is not correct; the color adds context. For each column, the bars indicate the various values that occurred. The color then indicates how often the values occurred relative to the neighboring values. Within each vertical set of bars, the values that occurred more frequently show as 'hotter' with orange and red, while those values that occurred less frequently show as shades of green.

To help illustrate these graph types, consider the graphs shown below.



All three graphs are showing the same data—the running 1-minute average of CPU utilization on a device over the last 24 hours. However, the way they visualize the data varies:

- The time series graph provides the typical view; each minute it adds the latest data point to the end of the line graph. Time moves forward along the x-axis from left to right, and the data values are indicated on the y-axis. What this graph doesn't show is how often each data point has occurred.
- The histogram groups together the values to show *how many* of each data point there are. Notice the tallest bar is the one between 30 and 40, which means the most common 1-minute CPU average value is in the 30-40% range. And how many times did this range of values occur? Based on the y-axis, there have been over 350 instances of values in this range. The next most frequently occurring values are in the 40-50% range (almost 300 occurrences), while the 0-10% range has almost no

occurrences, suggesting this CPU is rarely idle. What this graph doesn't show is how many of each data point occurred within a given time range.

- The heatmap makes use of elements from the other two graph types. Each small bar indicates that some number of instances occurred within the value range shown in the y-axis, at the given time shown in the x-axis. The color indicates which value ranges, for each given time, occurred *more* than others. To illustrate this, notice the vertical set of bars towards the right of the graph, at 18:00. In this example (at this zoom level), each column of vertical bars represents 12 minutes, and each small bar represents a bucket of 15 values. So the first (lowest) bar indicates that within this time range there were some values in the 0-14 range. The bar above indicates that within this time range there were some values in the 15-29 range, and so on. The color then indicates which bars have more values than others. In this example, the third bar is red indicating that for those 12 minutes most of the values fell into the 30-44 range (in this example the count is 21). By contrast, the first bar is the most green indicating that for those 12 minutes the least number of values fell into the 0-14 range (in this example the count is 1). This 'heat' information is also supported by the histogram; the most frequently occurring values were those in the 30-40 range, which indeed is the 'hotter' range in the heatmap.

The configuration model for graphs is to create graph panels and group them into one or more canvases.

To create a new graph panel on a canvas:

Graph Types

How to Create Graphs

Managing Graphs

Graph Tips and Tricks

Use Cases

1. Click the **Monitor > Graph** option in the left-nav bar.
2. Choose one of the following two options:
 - To create a graph in a new canvas, click the **+ New Canvas** button.
 - To create a graph within an existing canvas, select the desired canvas in the Saved Canvas and then click the **Add Graph +** button.

3. In the General section, provide the general descriptive information for the canvas (new canvas only) and graph:

Attribute	Description
Canvas Name	Name of the canvas
Description	(Optional) Description for the canvas.
Graph Name	Name of the graph panel.
Graph Type	Options include Time Series , Histogram , and Heatmap .
Time Range	Choose the time range for the graph. In real-time graphs, the time range sets the x-axis range. For example, selecting 12 hrs means the x-axis shows the last 12 hours of data.

4. Move down to the **Query** section. In the **FROM** section, define from where the data for the graph is coming:

Attribute	Description
Group	Choose the device group or network group.
Device	Choose a device from the group.
Topic / Rule	Choose the Paragon Insights topic/rule name. You can choose rolled-up measurements of fields you configured in rules. The rolled-up measurements are displayed in the list as topic/rule_roll-up-order. For example, my-rule_hourly and my-rule_yearly. The roll-up measurement's fields are listed as original-field-name_aggregation-type. For example, field-name_count and field-name_mean. Rolled-up measurements use aggregate functions on field data collected over a time to summarize the data into a single data point. You can configure how frequently the field data must be rolled-up or summarized. The default roll-up order (frequency) available in Paragon Insights are daily, hourly, monthly, weekly, or yearly.

5. In the **SELECT** section, select the data field and apply aggregation and transformation types to the data:

Attribute	Description
Field	Choose a field name. This list is derived based on the fields defined in the selected topic/rule.
Aggregation	(Optional) In the drop-down list, choose a data aggregation type.
Transformation	(Optional) In the drop-down list, choose a data transformation type.

6. In the **WHERE** section, filter data based on field and KPI key:

Attribute	Description
Tag Key / Field	(Optional) Choose a key or field. A key is an index field such as interface name. This list is derived based on the keys and fields defined in the selected topic/rule.

7. In the **GROUP BY** section, specify how to group the data based on time interval, fill, and KPI keys:

Attribute	Description
\$_interval	(Optional) Specify a time interval by which to group the data.
fill(null)	(Optional) Choose how to show a time interval when no data arrives from the device: null (default) Report the timestamp and null as the output value. none Report no timestamp and no output value. 0 Report 0 as the output value. previous Report the value from the previous time interval as the output value. linear Report the results of linear interpolation as the output value.

(Continued)

Attribute	Description
Tag Key (the + icon)	<p>(Optional) Choose a tag by which to group the data. A key is an index field such as interface name.</p> <p>This list is derived based on the keys and fields defined in the selected topic/rule.</p>

8. (Optional) Move down to the **Visualization** section, and define y-axis details.

9. Click **Save** to save the graph and display the data.

Managing Graphs

- To edit a graph, click the pencil icon located in the top right corner of the graph itself.
- To delete a graph, click the trash can icon located in the top right corner of the graph itself.
- To delete a canvas, click the trash can icon located in the top right corner of the canvas.

Graph Tips and Tricks

- To sort canvases on the Saved Canvas page, click on the column headings.
- To reorganize graphs on the screen, hover your mouse near the upper-left corner of a graph panel and click-and-drag it to the desired position.
- To resize a graph, hover your mouse over the lower-right corner of the graph panel and click-and-drag it to the desired size.
- To change the color of graph elements, click the color bar for the desired line item under the graph.
- To zoom in on a graph, click and drag across the desired section of the graph; to zoom out, double-click on the graph.
- To isolate an element on the graph, click its related line item under the graph; to view all elements again, click the same line item.

Use Cases

How do I monitor interface flaps for a single interface?

1. Playbook used: interface-kpis-playbook
2. Graph configuration

Edit Graph

General

Canvas Name*

TEST-canvas3

Description

Graph Name*

VMX1-ge-003-interface-flaps

Time Series

3 Hour

Query

FROM

VMX

VMX1

interface.statistics/check-interface-flaps

SELECT

flaps

mean()

Transformation

WHERE

interface-name

=

ge-0/0/3

AND

GROUP BY

\$_interval

fill(null)

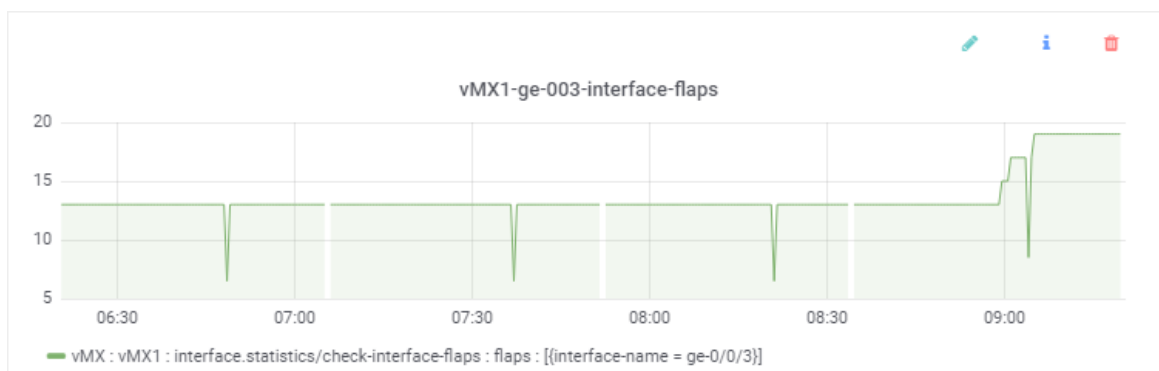
+

Visualization

Cancel

Save

3. Graph panel



How do I monitor interface flaps for all 'ge' interfaces on a device in a single graph?

1. Playbook used: interface-kpis-playbook
2. Graph configuration

Edit Graph

General

Canvas Name*

TEST-canvas3

Description

Graph Name*

vmx--interface-flaps

Time Series

3 Hour

Query

FROM

vmx

vmx1

interface.statistics/check-interface-flaps

SELECT

flaps

mean()

Transformation

WHERE

interface-name

== (matches with)

ge

AND

GROUP BY

\$_interval

fill(null)

interface-name

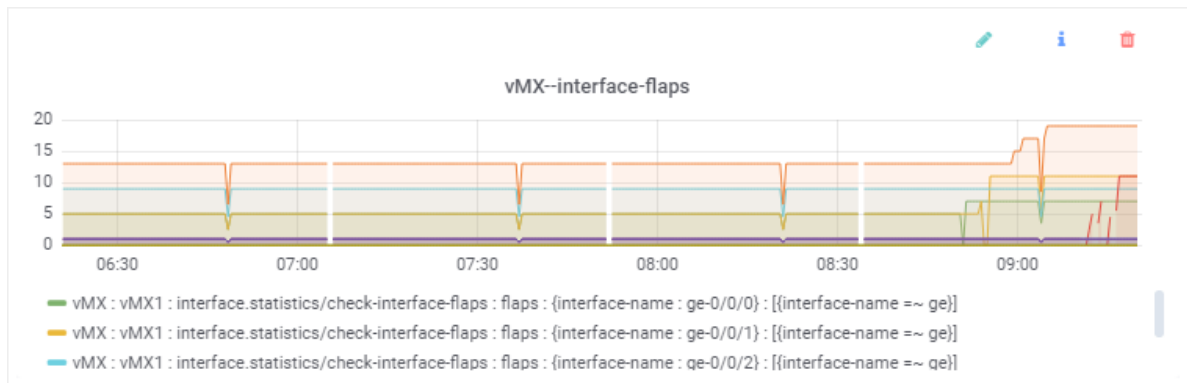
+

Visualization

Cancel

Save

3. Graph panel



How do I monitor system memory usage for all devices in a device group in a single graph?

1. Playbook used: system-kpis-playbook
2. Graph configuration

Edit Graph

▼ General

Canvas Name*

Canvas1

Description

Graph Name*

All-devices--system-memory

Time Series

▼

6 Hour

▼

▼ Query

FROM

VMX

▼

VMX1

▼

system.memory/check-system-memory

▼

SELECT

re-memory-buffer

▼

mean()

×

▼

Transformation

▼

WHERE

+

GROUP BY

1m

▼

fill(null)

▼

+

FROM

VMX

▼

VMX2

▼

system.memory/check-system-me...

▼

SELECT

re-memory-buffer

▼

mean()

×

▼

Transformation

▼

WHERE

+

GROUP BY

1m

▼

fill(null)

▼

+

FROM

VMX

▼

MX240

▼

system.memory/check-system-me...

▼

SELECT

re-memory-buffer

▼

mean()

×

▼

Transformation

▼

WHERE

+

GROUP BY

\$_interval

▼

fill(null)

▼

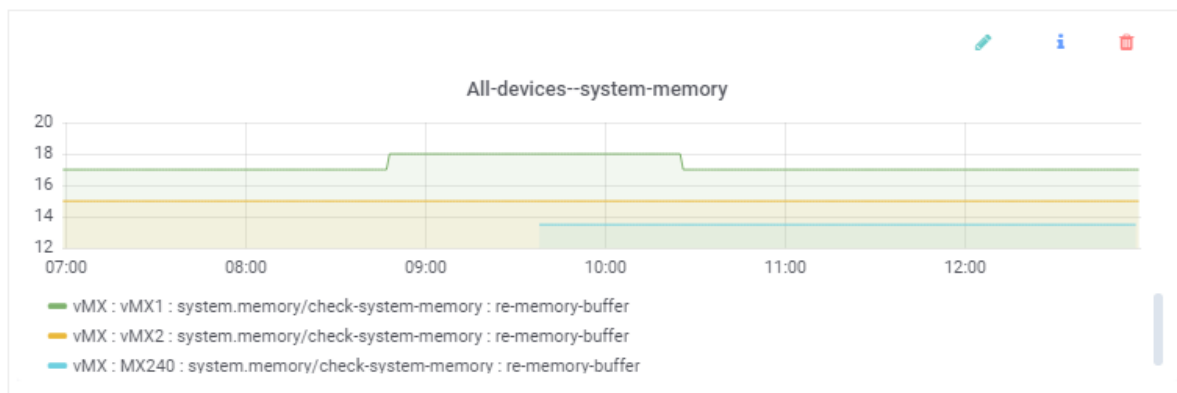
+

Visualization

Cancel

Save

3. Graph panel



How do I monitor RE CPU usage for multiple devices in a single graph?

1. Playbook used: system-kpis-playbook
2. Graph configuration

Edit Graph

▼ General

Canvas Name*

Canvas1

Description

Graph Name*

All-devices--CPU-utilization

Time Series

▼

6 Hour

▼

▼ Query

FROM

VMX

▼

VMX1

▼

system.cpu/check-system-cpu

▼

SELECT

re-cpu-utilization

▼

mean()

×

▼

Transformation

▼

WHERE

+

GROUP BY

1m

▼

fill(null)

▼

+

FROM

VMX

▼

VMX2

▼

system.cpu/check-system-cpu

▼

SELECT

re-cpu-utilization

▼

mean()

×

▼

Transformation

▼

WHERE

+

GROUP BY

1m

▼

fill(null)

▼

+

FROM

VMX

▼

MX240

▼

system.cpu/check-system-cpu

▼

SELECT

re-cpu-utilization

▼

mean()

×

▼

Transformation

▼

WHERE

+

GROUP BY

1m

▼

fill(null)

▼

+

+

▼ Visualization

Y-Axis

0

100

CPU utilization

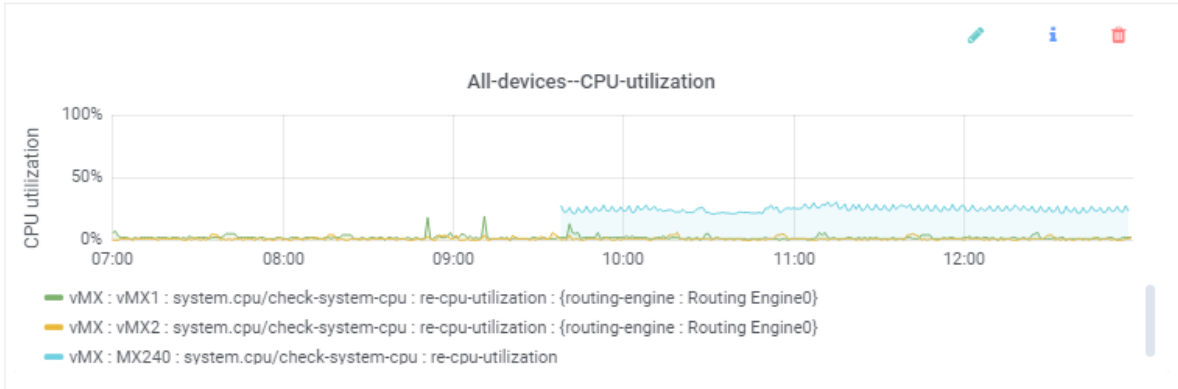
Percent(0 - 100)

▼

Cancel

Save

3. Graph panel



How do I monitor RE CPU usage for multiple devices side by side?

1. Playbook used: system-kpis-playbook
2. Graph configuration

Add Graph

General

Canvas Name* Canvas1 Description

Graph Name* vMX1-CPU-Utilization Time Series 6 Hour

Query

FROM vMX vMX1 system.cpu/check-system-cpu

SELECT re-cpu-utilization mean() Transformation

WHERE +

GROUP BY \$__interval fill(null) +

+ Visualization

Cancel Save

Edit Graph

General

Canvas Name*

Canvas1

Description

Graph Name*

MX240-CPU-Utilization

Time Series

6 Hour

Query

FROM

VMX

MX240

system.cpu/check-system-cpu

SELECT

re-cpu-utilization

mean()

X

Transformation

WHERE

+

GROUP BY

\$_interval

fill(null)

routing-engine

+

+

Visualization

Cancel

Save

3. Graph panel



Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.0.0	Paragon Insights Release 4.0.0 introduces GUI enhancements brought about by the migration to a new framework.

RELATED DOCUMENTATION

Manage Devices, Device Groups, and Network Groups	 121
Paragon Insights Rules and Playbooks	 141

Understand Resources and Dependencies

IN THIS SECTION

- [Resources](#) | [207](#)
- [Resource Dependencies](#) | [208](#)
- [Use Cases](#) | [209](#)

Paragon Insights monitors network resources — such as devices, interfaces, protocols, label switched paths — through rules deployed in the network. A rule can capture specific information about a key performance indicator (KPI) in a network and generate alerts when the KPI experiences an error event, but rules cannot decipher the root cause of the error event. You require a wider view to correlate an error event in one resource to a failure in another resource.

Paragon Insights tackles the need to connect events in a network through resources. A resource is a building block that creates a wider network view by forming single-level dependencies vis-à-vis other resources.

The following list has frequently used terms and concepts connected with resources:

Resource	<p>A resource is a specific component that constitute the network.</p> <p>For example, chassis, line card, protocols, system memory, interfaces, and so on.</p>
Resource instances	<p>Resources such as an interface, Flexible PIC Concentrators (FPC), router ids, or virtual private networks can have many instances. An instance is a specific realization of the resource. For example, an interface includes instances such as ge-0/0/1, et-1/0/0, xe-2/0/1, and so on.</p>
Property	<p>Properties define characteristics of a resource. A property is comparable to fields in rules.</p>

For example, *neighbor-id* and maximum transmission unit (MTU) are characteristics of routing-options resource and interface resource, respectively.

Key properties in resources	<p>A key property uniquely identifies all instances of a resource.</p> <p>For example, <i>interface-name</i> property uniquely identifies interface resource instance with name <i>ge-0/0/0</i> from other instances. MTU cannot be a unique property.</p>
Resource dependency	Defines the relationship between two resources.
Dependent resource	In a resource dependency, a dependent resource is the one that depends on another resource. In a single-level resource dependency, a dependent resource is a child resource of another resource.
Dependency resources	In a resource dependency, a dependency resource is the one that impacts another resource. In a single-level resource dependency, a dependency resource is a parent resource of another resource.

Resources

A resource can be part of a device or the network. Device resources can be an interface, line cards, chassis, OSPF, etc. Network resources are resources that span multiple devices in a network, such as IPSec tunnels, VPN, etc.

As with rules, you configure resources under the topic hierarchy in Paragon Insights. A resource and its properties are derived from rules. In this process, resources consolidate the following aspects in rule configurations.

- A resource, such as interface, is configured in different rules depending on which aspect of interface the rule monitors.
- Each rule has different field configuration for a resource property.

An interface name can be configured as *interface-name* in one rule and *intf-name* in another rule.

When you configure interface as a resource, you can choose the specific interface rules from which Paragon Insights detects instances of interface resource. The exact rules you select to identify a resource depend on your use case.

A resource property aggregates instances from referenced rules. When you configure a resource property, you can refer multiple rules where the field configurations are different for that property.

Resource Dependencies

Resource dependency defines the relationship between a dependent resource (child resource) and a dependency resource (parent resource). While configuring dependency, you begin with a dependent resource and refer dependency resources that the child resource depends on. A dependency configuration also has terms that contain the logic to map dependency between two resources.

There are three types of dependency based on the type of resources involved, as described in [Table 10 on page 208](#). You can define dependencies between resources in the same device (Local Device and Network), between resources in different devices (Other Device), between a network resource to another network resource (Other Network), and a network resource to a device resource (Other Device).

Table 10: Types of Resource Dependency

Local Device and Network	Other Device	Other Network
Interface → line card	Interface 1 (device 1) → interface 2 (device 2)	virtual private network (VPN)→ interface 1 (device 1)
Line card → chassis	OSPF (device 1) → OSPF (device 2)	VPN → IPSec tunnel

You can configure multiple single-level dependencies for a resource. Consider the following chains of dependencies:

- VPN → LSP → interface → Line card
- VPN → interface → Line card

When you configure VPN as a resource, you can define VPN's dependency on Label Switched Paths (LSPs) as one dependency and VPN's dependency on interface as a second dependency. For LSP as a resource, you can define its dependency on interface. For interface as a resource, you can define its dependency on line card.

A dependency term logic can involve checking parts of a dependent resource property with parts of dependency resource's property, checking all instances of the dependency resource, checking all devices that have dependency resource instances, checking all or specific device groups, and checking all or specific network groups.

Paragon Insights supports *matches-with* and user-defined functions in the dependency condition.

As dependency configurations can involve complex operations, Paragon Insights also allows you to execute such operations in a function. The function returns a Boolean value that can be used to check dependency.

Use Cases

Resource and dependency configurations serve the following use cases:

- Root cause analysis** You can understand the root cause of a failure in your network. Refer the resource dependencies and use the relationship between resources to diagnose alerts generated by triggers in rules.
- Smart alarms** Through smart alarms, Paragon Insights links several resources that have a failure to another resource that is the cause of the failure.

RELATED DOCUMENTATION

[About the Resources Page](#) | 209

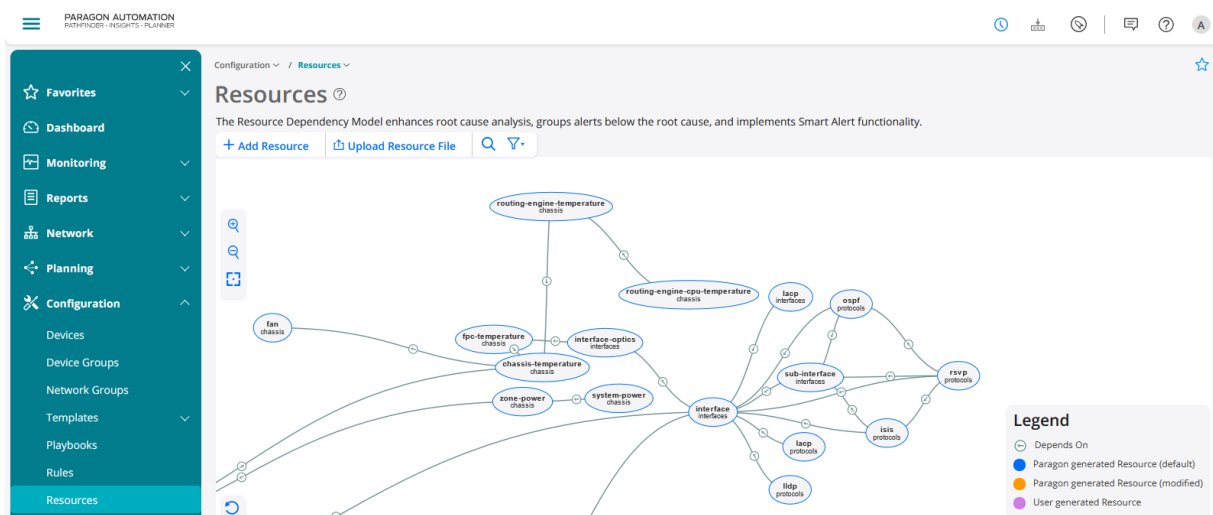
About the Resources Page

IN THIS SECTION

- [Tasks You Can Perform](#) | 210
- [Fields in Resources Table](#) | 211

You can access the Resources page from **Configuration > Resources**.

Figure 55: Resources and Dependency Visual Panel



The Resources page displays a visual representation of default resources in the Paragon application. It also shows that a dependent (child) resource depends on the preceding dependency (parent) resource in the model. The visualization is synchronous with Resources table so that you can work with resources from either the visual model or the table. Starting with Paragon Insights Release 4.3, you can reset the zoom level of the visual panel of resources using the reset button after zooming in or zooming out. You can also drag and change the position of resources on the visual panel. Paragon Insights retains any change to the position of resources, even after you leave the Resources page.

When you click on a resource on the visualization, Paragon Insights shows key properties of the resource, dependency resource (parent resource) of the current resource, and dependent resources of the current resource (child resources).

When you click on the dependency arrow on the visualization, Paragon Insights shows the terms and dependency type configured in the child resource.

Tasks You Can Perform

You can perform the following tasks in the Resources page:

- Add a resource from the resource visualization or table. See ["Add Resources for Root Cause Analysis" on page 212](#)
- Configure dependency resource. See ["Configure Dependency in Resources" on page 215](#)
- Example Dependency Configuration of OSPF protocol. See ["Example Configuration: OSPF Resource and Dependency" on page 221](#)

- Edit a resource from the resource visualization or table. See ["Edit Resources and Dependencies" on page 232](#)
- Upload a resource from the visual panel or the resource table. See ["Upload Resources" on page 235](#)
- Download a resource from the visual panel. See ["Download Resources" on page 236](#)
- Clone and modify resource configuration from the visual panel. See ["Clone Resources" on page 236](#)
- Delete a resource from the resource visualization or table. See ["Delete Resources and Dependencies" on page 237](#)
- Filter a resource from the resource visualization or table. See ["Filter Resources" on page 234](#)

Fields in Resources Table

[Table 11 on page 211](#) displays the fields on the Resource Dependency Model page.

Table 11: Fields on the Resource Dependency Model Page

Field	Description
Name	Names of topics that expands to resources within the respective topic.
Description	Description you enter for the resource during resource configuration.
Keys	When you expand the topics, you can view key properties you configured for each resource.
Dependency	Parent resource that impacts the resource.
Generated	Shows Paragon for system-generated (default) resources.

RELATED DOCUMENTATION

[Understand Resources and Dependencies](#) | 206

Add Resources for Root Cause Analysis

Resource configuration requires references to rules you want to link to the resource. Ensure that you configure the rules before you start resource configuration.

To configure a resource:

- 1. Select **Configuration > Resources**.
The Resources page appears.
- 2. Do one of the following:
 - Click **+Add Resource** on the model map.
 - Click **+** on the Resources table.

The Add Resource page appears.

- 3. Enter the details as described in [Table 12 on page 212](#).

Table 12: Fields of Properties and Functions in Resource Configuration

Field	Description
General Information	
A resource, like rules, is defined under the <i>topic</i> hierarchy.	
Resource Name	Enter name of the resource. The name can follow the regex pattern: [a-zA-Z][a-zA-Z0-9_-]* and can have up to 64 characters. For example, chassis, interface, or system.
Topic	Select the topic to which the resource belongs. For example, interface, chassis, or protocol.
Description	Enter a short description of the resource and the resources on which this resource is dependent.

Table 12: Fields of Properties and Functions in Resource Configuration *(Continued)*

Field	Description
Properties <p>Properties are characteristics such as name, MTU, neighbor-id etc. of a particular resource. A property of one resource can be matched with property of another resource to establish dependency.</p>	
Property name	<p>Enter name of the resource property. The name can follow the regex pattern: <code>[a-zA-Z][a-zA-Z0-9_-]*</code> and can have up to 64 characters.</p> <p>For example, neighbor-id and interface-name.</p>
Property type	<p>Select the type of property input.</p> <p>You can choose from string, integer, floating point, or unsigned.</p>
Add as a key	<p>If you enable a resource property as key, Paragon Insights uses this property to uniquely identify multiple instances of that property as a resource. You can mark more than one property as key.</p> <p>For example, in interface property that uses interface-name as a key, Paragon Insights identifies ge-0/0/1, ge-1/2/0, ge-1/0/0 as unique instances of interface resource.</p>
Source Configuration <p>Rules and fields are the sources from which Paragon Insights discovers a resource property.</p>	
Rule Name	<p>Add a rule name in the topic/rule format.</p> <p>For a resource property, you can add one or more rules as sources.</p>
Field Name	Select a field configured in the rule.

Table 12: Fields of Properties and Functions in Resource Configuration (Continued)

Field	Description
(Optional) Functions	
You can enter a function in conditional statements that check for resource dependency.	
Function name	Enter a function name that follows the [a-zA-Z][a-zA-Z0-9_-]* pattern.
Path to function	Select the file name where functions are defined. Paragon Insights supports only Python functions.
Method name	Select the name of the function you want to execute from the given file.
Description	Enter a description of the function.
Arguments (Add Item)	
Name	Add argument or parameters defined in the function.
Mandatory	Toggle the switch on if the function arguments you added must be included when Paragon Insights calls the function.

4. Click Save & Exit.

The Save Resource Configuration dialog appears.

5. Do one of the following:

- Click **Save and Deploy**.

You can save the resource configuration and deploy it.

- Click **Save**.

You can only save the resource configuration but not deploy it.

If you save your configuration by only entering the details in the table, you complete the resource configuration. You can see the new resource on the Resources page.

To connect the resource you add to another resource on the visual panel, you must configure dependency for this resource.

RELATED DOCUMENTATION

| [Configure Dependency in Resources](#) | 215

Configure Dependency in Resources

Before you configure dependency for a dependent (child) resource, you must complete configuring the dependency (parent) resources. See ["Add Resources for Root Cause Analysis" on page 212](#) for more information.

When you configure dependency for a child (dependent) resource, you configure terms that have the logical conditions to check for dependency. Starting with Paragon Insights Release 4.3, you can swap the order of dependency terms in the Dependency page. Paragon Insights executes the terms based on the sequence in which the terms appear in the GUI. The first term is evaluated first, followed by the second, the third, and so on.

To configure a resource dependency:

1. Select **Configuration > Resources**.
The Resources page appears.
2. Select a resource from Resources table and click edit (pencil icon).
The Edit Resource page appears.
3. Click **Next** until you view the Dependency page.
4. Enter the details as described in [Table 13 on page 216](#).
5. Click **Save**.
The Dependency page appears.
6. Click **Next**.
You can see visual representation of the resource and dependency configuration that you added.
7. (Optional) Click View Tree to view a collapsed view of the resource and the dependency configurations.
8. (Optional) Click **View JSON** to preview the JSON format of the configuration.
9. Click **Save & Exit**.
The Save Resource Configuration dialog appears.
10. Do one of the following:

- Click **Save and Deploy**.

Paragon Insights saves and deploys the configuration to form dependency between the resources and generate smart alarms.

- Click **Save**.

Paragon Insights saves the configuration to form dependency but does not generate smart alarms.

You can see the new resource and its dependency mapped in the Resources page.

Table 13: Fields in Dependency Configuration

Field	Description
<i>Dependent-resource-name</i> depends on	
Enter configuration details of dependency (parent) resource to establish dependency with dependent (child) resource.	
Resource Name	Select the dependency resource that impacts your dependent (child) resource.
Add Variables	
Variables in this section are used to extract parts of the child resource's property. Such variables are used to check dependency using conditional statements.	
Variables configured here can be used to check dependency in all terms.	
Name	Enter name for the variable. The name can follow the regex pattern: [a-zA-Z][a-zA-Z0-9_-]* and can have up to 64 characters. For example, interface-name-split.

Table 13: Fields in Dependency Configuration (*Continued*)

Field	Description
Expression	<p>Enter a regular expression to extract parts of a property value.</p> <p>See regex package for more information.</p> <p>For example, if you use the regular expression “.*(\d+)/(\d+)/\d+” on interface-name property of interface resource, you can extract line card number and PIC number.</p> <p>The example regex forms two capture groups to extract line card number and PIC number. The first capture group, which is line card number, can be referred as interface-name-split-1 and second capture group, which is PIC number, can be referred as interface-name-split-2.</p>
Resource label	Select the name of the dependent (child) resource.
Field	Select a property in the child resource on which regex expression should be applied.
Case sensitive	<p>Enable this field.</p> <p>If you enable this field, Paragon Insights ignores the case of resource properties when processing conditions.</p>

Add Terms (Terms > Add Item)

Terms contain the logic to check for a dependency relation between resources.

Term Name	Enter a term name that follows the [a-zA-Z][a-zA-Z0-9_-]* pattern.
Depends on Multiple Instances	<p>Enable this field if your dependency logic involves checking if a child resource property is dependent on multiple instances of a parent resource property.</p> <p>For example, you configured Aggregated Ethernet (link aggregation group) as a dependent (child) resource to interface resource. As AE depends on multiple links (interface resource), you must enable Depends on Multiple Instances knob.</p>

Table 13: Fields in Dependency Configuration (Continued)

Field	Description
Dependency Type	<p>Select the type of dependency.</p> <p>Local Device and Network dependency is the default option.</p> <p>If you select Other Device dependency, configure details in For Every Device section.</p> <p>If you select Other Network dependency, configure details in For Every Network Group section.</p> <p>See "Understand Resources and Dependencies" on page 206 for more information.</p>
Evaluate Next Term	<p>This field is disabled by default.</p> <p>Enable this field if your dependency logic involves checking condition in other terms.</p>
(Optional) Add Variable	<p>Enter the following details of a child resource property:</p> <ul style="list-style-type: none"> • Name • Regular Expression • Resource Name • Resource Property Field <p>You can configure a variable for a child resource property. Variable you configure here can only be used in conditional statements within the term.</p>
<p>For Every Device</p> <p>Selects a device from a list of devices in device groups.</p>	
Label As	<p>Enter a name for the dependency (parent) resource instance that is selected. The name must follow the [a-zA-Z][a-zA-Z0-9_-]* pattern.</p>
Across all device groups	<p>Enable this field if you want Paragon Insights to check devices in all device groups for dependency.</p>

Table 13: Fields in Dependency Configuration (Continued)

Field	Description
In device groups	Enter the names of device groups. The child resource property will be checked against devices in the specified device groups for dependency.
<p>For Every Network Group</p> <p>Instructs Paragon Insights to perform the dependency logic in all or select network groups.</p>	
Label As	Enter a name for the dependency (parent) resource instance that is selected. The name must follow the [a-zA-Z][a-zA-Z0-9_-]* pattern.
Across all network groups	Enable this field if you want Paragon Insights to check all network groups for dependency.
In network groups	Enter the names of network groups. The child resource property will be checked against instances of network resource property in the specified network groups.
<p>Locate Resource</p> <p>The locate resource iterates through all instances of a given resource.</p>	
Resource Name	<p>Select a resource.</p> <p>If the resource type is Local Device & Network, the resource in the list is of the format <i>topic-name/resource-name</i>.</p> <p>If the resource type is Other Device or Other Network, then the resource in the list is of the format <i>for every device/network label-as:topic-name/resource-name</i>.</p>
Label As	<p>Enter a label name.</p> <p>Paragon Insights stores each instance of the selected resource in the label. For example, instances such as interface instances or OSPF sessions.</p>

Table 13: Fields in Dependency Configuration (Continued)

Field	Description
(Optional) Add Variable	<p>Enter the following details of a resource property of the selected resource:</p> <ul style="list-style-type: none"> • Name • Regular Expression • Resource Name • Resource Property Field <p>You can configure a variable for a property of the selected resource. Variable you configure here can only be used in conditional statements within the term.</p>
Conditions	<p>Conditions in Locate Resource are used to identify if the selected resource is correct one or not. If the condition does not match for a particular instance it picks the next instance and checks the condition. This continues until we get an instance where conditions matches, or all the instances of the resources are exhausted.</p> <p>To check dependency, select a resource property in left-hand side (LHS), a remote resource property in RHS and the operator to check for a condition.</p> <p>Paragon Insights supports <i>matches-with</i> as a condition. You can also add functions in the LHS field of the conditional statement.</p>

RELATED DOCUMENTATION

Add Resources for Root Cause Analysis 212
Example Configuration: OSPF Resource and Dependency 221
Understand Resources and Dependencies 206

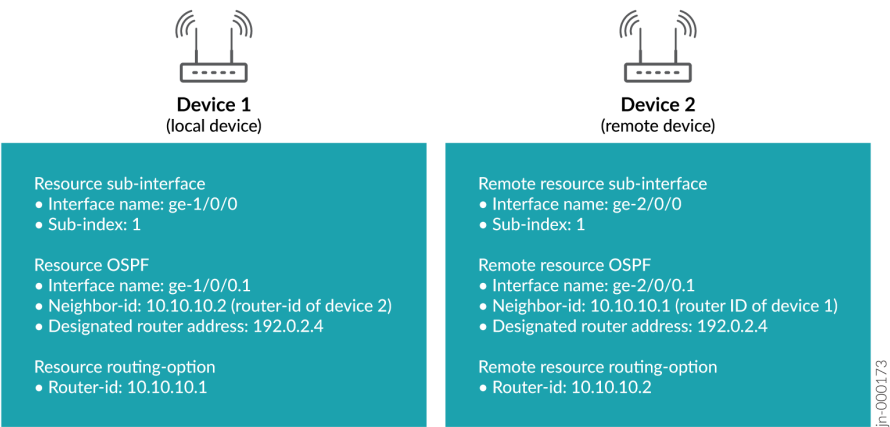
Example Configuration: OSPF Resource and Dependency

Before you begin configuring the OSPF resource and dependency, you must deploy the following configurations:

- Sub-interface as a resource with interface-name and sub-interface index as key properties.
- Routing options as a resource with router-id as a key property.

The example configurations create OSPF protocol dependency between two devices. The OSPF protocol runs on an interface between two devices. So, the protocol forms two types of dependencies in the example configuration. The first dependency is between OSPF and the device, known as Local Device and Network dependency in the configuration. The second dependency is between OSPF protocol and the remote device, known as Other Devices dependency in the configuration. Paragon Insights establishes the two dependencies using the following properties in device 1 and device 2 as shown in [Figure 56 on page 221](#).

Figure 56: Resources and Properties Used for OSPF Dependencies

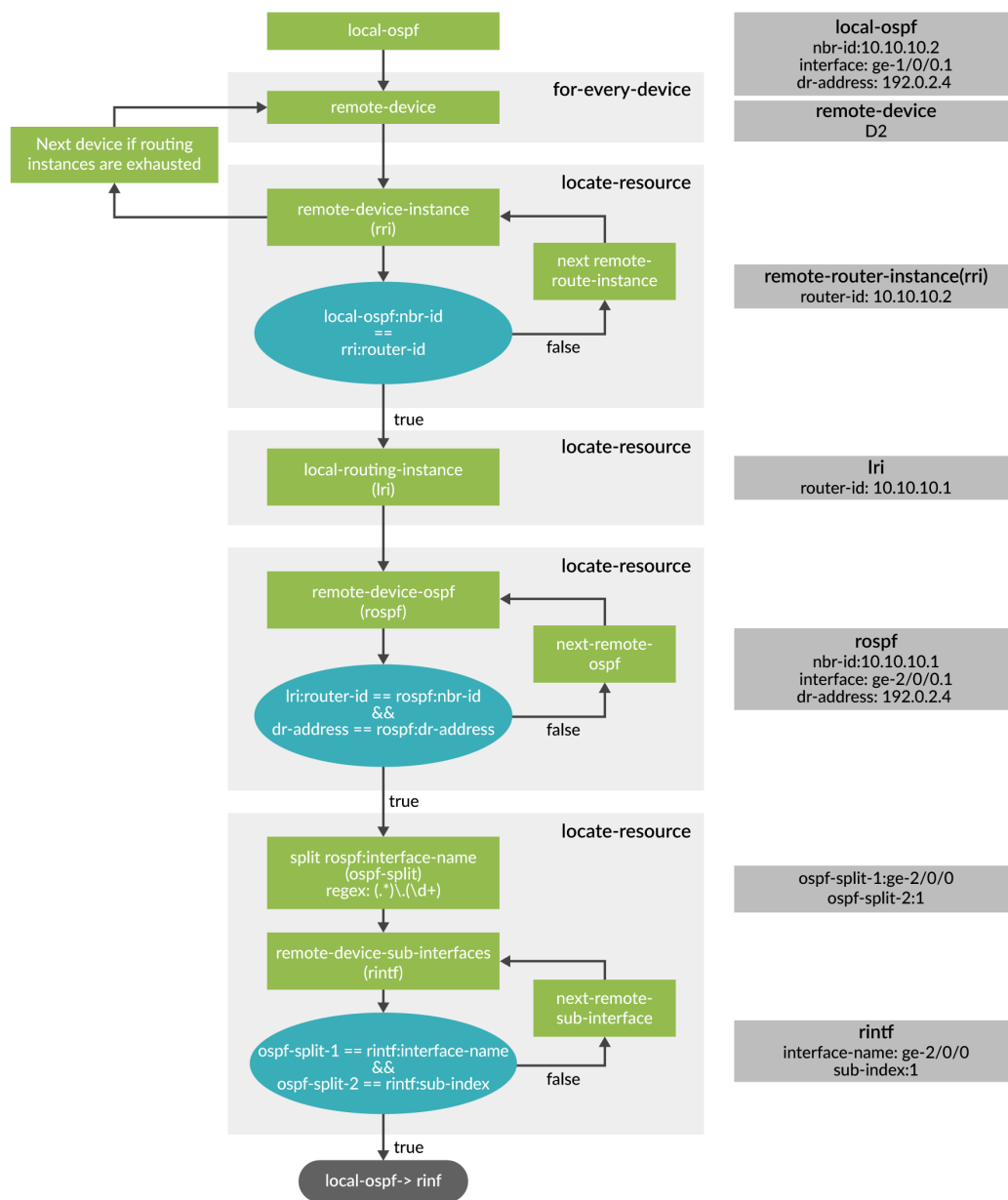


The sub-interface properties and OSPF interface names are used to create local dependency between a local device and the OSPF session at one end and between the remote device and the OSPF session at the other end.

The neighbor ID, designated router address and the device's router ID are used to create device-to-device (Other Device) dependency.

FIGURE2 shows how the iteration in locate resource configurations creates the device-to-device dependency.

Figure 57: Locate Resource Flowchart of OSPF Other Device Dependency



jn-000174

To configure OSPF resource and its dependencies:

1. Select **Configuration > Resources**.

The Resources page appears.

2. Select OSPF resource from the Resources table and click edit (pencil icon).

The Edit Resource page appears.

3. Enter the details as described in [Table 14 on page 223](#).

Table 14: Fields of Properties in OSPF Resource Configuration

Field	Description
-------	-------------

General Information

A resource, like rules, is defined under the *topic* hierarchy.

Resource Name	<p>Enter ospf.</p> <p>The name can follow the regular expression pattern: [a-zA-Z][a-zA-Z0-9_-]* and can have up to 64 characters.</p> <p>For example, the name can also be Ospf-1.</p>
Topic	Select protocol .
Description	Enter a short description of OSPF dependencies.

Property Designated Router Address

Properties are characteristics such as name, MTU, neighbor-id etc. of a particular resource. A property of one resource can be matched with the property of another resource to establish dependency.

Property name	<p>Enter dr-address.</p> <p>The name can follow the regular expression pattern: [a-zA-Z][a-zA-Z0-9_-]* and can have up to 64 characters.</p>
Property type	Select String .

Table 14: Fields of Properties in OSPF Resource Configuration *(Continued)*

Field	Description
Add as a key	<p>Not applicable.</p> <p>If you enable a resource property as key, Paragon Insights uses this property to uniquely identify multiple instances of that property as a resource. You can mark more than one property as key.</p> <p>For example, in interface property that uses the interface name as a key, Paragon Insights identifies ge-0/0/1, ge-1/2/0, ge-1/0/0 as unique instances of the interface resource.</p>
Source Configuration	
Rules and fields are the sources from which Paragon Insights discovers a resource property.	
Rule Name	Select protocol.ospf/check-ospf-neighbor-information .
Field Name	Select dr-address .
Property Interface Name	
Properties are characteristics such as name, MTU, neighbor-id etc. of a particular resource. A property of one resource can be matched with the property of another resource to establish dependency.	
Property name	<p>Enter interface-name.</p> <p>The name can follow the regex pattern: <code>[a-zA-Z][a-zA-Z0-9_-]*</code> and can have up to 64 characters.</p>
Property type	Select String .

Table 14: Fields of Properties in OSPF Resource Configuration *(Continued)*

Field	Description
Add as a key	<p>Enable this option.</p> <p>If you enable a resource property as key, Paragon Insights uses this property to uniquely identify multiple instances of that property as a resource. You can mark more than one property as key.</p> <p>For example, in interface property that uses the interface name as a key, Paragon Insights identifies ge-0/0/1, ge-1/2/0, ge-1/0/0 as unique instances of the interface resource.</p>
Source Configuration	
Rules and fields are the sources from which Paragon Insights discovers a resource property.	
Rule Name	Select protocol.ospf/check-ospf-neighbor-information .
Field Name	Select interface-name .
Property Neighbor ID	
Properties are characteristics such as name, MTU, neighbor-id etc. of a particular resource. A property of one resource can be matched with the property of another resource to establish dependency.	
Property name	<p>Enter neighbor-id.</p> <p>The name can follow the regular expression pattern: [a-zA-Z][a-zA-Z0-9_-]* and can have up to 64 characters.</p>
Property type	Select String .

Table 14: Fields of Properties in OSPF Resource Configuration (Continued)

Field	Description
Add as a key	<p>Enable this option.</p> <p>If you enable a resource property as key, Paragon Insights uses this property to uniquely identify multiple instances of that property as a resource. You can mark more than one property as key.</p> <p>For example, in interface property that uses the interface name as a key, Paragon Insights identifies ge-0/0/1, ge-1/2/0, ge-1/0/0 as unique instances of the interface resource.</p>

Source Configuration

Rules and fields are the sources from which Paragon Insights discovers a resource property.

Rule Name	Select protocol.ospf/check-ospf-neighbor-information .
Field Name	Select neighbor-id .

4. Click **Next** twice to go to the Dependency page.
The Dependency page appears.
5. Click **+** in the Dependency page.
A new Resource section appears.
6. Enter the details as described in [Table 15 on page 226](#).

Table 15: Local Device and Network Dependency Configuration

Field	Description
OSPF depends on	
Resource Name	Select interfaces/sub-interfaces .

Table 15: Local Device and Network Dependency Configuration *(Continued)*

Field	Description
Add Terms (Terms > Add Item) <p>Terms contain the logic to check for a dependency relation between resources.</p>	
Term Name	Enter ifl-dependency The name follows the [a-zA-Z][a-zA-Z0-9_-]* pattern.
Depends on Multiple Instances	Enable this option. Paragon Insights checks with multiple instances of the property interface name and the property sub-interface index configured in resource sub-interface.
Dependency Type	Select Local Device & Network . See "Understand Resources and Dependencies" on page 206 for more information on dependency types.
Add Variable	Name: Enter ospf-interface-split . Expression: Enter (.*)\.(d+) Field: Enter \$interface-name .
Locate Resource <p>Iterates over all instances of the interfaces and the sub-interface index in resource sub-interfaces to find local dependency.</p>	
Resource Name	Select interfaces/sub-interfaces
Label As	Enter ifl . Paragon Insights stores each instance of the interface resource to check for dependency.

Table 15: Local Device and Network Dependency Configuration (Continued)

Field	Description
<p>Conditions</p> <p>Conditions in Locate Resource are used to identify if the selected resource is the correct one or not. If the condition does not match a particular instance, Paragon Insights picks the next instance and checks for the condition. This continues until we get an instance where the condition matches, or all the instances of the resource are exhausted.</p>	<p>The first condition checks for a match between the OSPF resource's interface property and the interface resource's interface instances.</p> <p>In the left-hand side (LHS), select \$ospf-interface-split-1.</p> <p>Select <i>matches-with</i> as operator.</p> <p>In the right-hand side (RHS), select \$ifl:interface-name.</p> <p>Click + to add a second condition that checks for a match between the OSPF interface's sub-interface index and the resource sub interface's sub-interface index.</p> <p>In the LHS, select \$ospf-interface-split-2.</p> <p>Select <i>matches-with</i> as operator.</p> <p>In the RHS, select \$ifl:sub-interface-index.</p>

7. Click **Save**.

The Edit Resource page appears.

You can find the new term in Terms section. The OSPF's interface name and interface sub-index is checked against the sub interface resource's interface name and sub-index. When Paragon Insights finds a match, the interface from the OSPF resource monitored in a device forms a local dependency with the interface from the interface resource of the same device.

8. Click **+** in the Terms section.

You can add a second term to configure Other Device dependency for OSPF resource.

9. Enter the details as described in [Table 16 on page 228](#).

Table 16: Other Device Term Configuration

Field	Description
-------	-------------

Add Terms (Terms > Add Item)

Terms contain the logic to check for a dependency relation between resources.

Table 16: Other Device Term Configuration *(Continued)*

Field	Description
Term Name	<p>Enter ifl-remote-dependency.</p> <p>The name must follow the <code>[a-zA-Z][a-zA-Z0-9_-]*</code> pattern.</p>
Dependency Type	<p>Select Other Device dependency.</p> <p>See "Understand Resources and Dependencies" on page 206 for more information on dependency types.</p>
For Every Device <p>Selects a device from a list of devices in device groups.</p>	
Label As	<p>Enter remote-device.</p> <p>The name follows the <code>[a-zA-Z][a-zA-Z0-9_-]*</code> pattern.</p>
Locate Resource <p>Define condition to check if the local device's OSPF neighbor-id is the remote device's router-id.</p>	
Resource Name	Enter remote-device:protocols/routing-instance .
Label As	Enter remote-routing-instance .
<p>Conditions</p> <p>Conditions in Locate Resource are used to identify if the selected resource is the correct one or not. If the condition does not match a particular instance, Paragon Insights picks the next instance and checks for the condition. This continues until we get an instance where the condition matches, or all the instances of the resource are exhausted.</p>	<p>In the LHS, select \$neighbor-id.</p> <p>In the operator field, select <i>matches-with</i>.</p> <p>In the RHS, select \$remote-routing-instance:router-id.</p>

Table 16: Other Device Term Configuration *(Continued)*

Field	Description
Locate Resource	
Use resource Routing Instance to collect router-id of the local device.	
Resource Name	Enter protocol/routing-instance .
Label As	Enter label name as local-routing-instance .
Locate Resource	
Define condition to check if the remote device's OSPF neighbor-id matches with the router-id of the local device.	
Resource Name	Enter remote-device: protocols/ospf .
Label As	Enter remote-ospf .
<p>Conditions</p> <p>Conditions in Locate Resource are used to identify if the selected resource is the correct one or not. If the condition does not match a particular instance, Paragon Insights picks the next instance and checks for the condition. This continues until we get an instance where the condition matches, or all the instances of the resource are exhausted.</p>	<p>In the LHS, enter \$local-routing-instance:router-id.</p> <p>In the operator field, select <i>matches-with</i>.</p> <p>In the RHS, enter \$remote-ospf:neighbor-id.</p> <p>Click + to add a second condition that checks for a match between local device OSPF's designated router address and remote OSPF's designated router address.</p> <p>In the LHS, enter \$dr-address</p> <p>In the operator field, select <i>matches-with</i>.</p> <p>In the RHS, enter \$remote-ospf:dr-address</p>
Locate Resource	
Define a condition to check if interface name and sub-index in the remote OSPF matches with interface name and sub-index of the remote device's interface.	

Table 16: Other Device Term Configuration (Continued)

Field	Description
Resource Name	Enter remote-device: interfaces/sub-interface .
Label As	Enter remote-ifl .
Conditions	<p>The first condition checks for a match between remote OSPF resource's interface property and remote interface resource's interface instances.</p> <p>In the LHS, select \$ospf-remote-interface-split-1.</p> <p>Select <i>matches-with</i> as operator.</p> <p>In the RHS, select \$remote-ifl:interface-name.</p> <p>Click + to add a second condition that checks for a match between remote OSPF interface's sub interface index and remote interface instance's sub interface index.</p> <p>In the LHS, select \$ospf-remote-interface-split-2.</p> <p>Select <i>matches-with</i> as operator.</p> <p>In the RHS, select \$remote-ifl:sub-interface-index.</p>

10. Click *Next*.

You can see a collapsed view of the resource and the dependency configuration you added.

11. (Optional) Click *View JSON* to preview the JSON format of the configuration.

12. Click *Save & Exit*.

The Save Resource Configuration dialog appears.

Do one of the following:

a. Click *Save and Deploy*.

Paragon Insights saves and deploys the configuration to generate smart alarms.

b. Click *Save*.

Paragon Insights saves the configuration but does not generate smart alarms based on the new resource and dependency configuration you add.

You can see the OSPF resource and its dependency in the visual panel of the Resources page.

RELATED DOCUMENTATION

[Configure Dependency in Resources | 215](#)

[Add Resources for Root Cause Analysis | 212](#)

Edit Resources and Dependencies

IN THIS SECTION

- [Edit a Resource | 232](#)
- [Edit Resource Dependency | 233](#)

Users can edit Paragon Insights-generated (system) resources, user-generated resources, and dependencies from the Resources page (**Configuration > Resources**). If you want to restore original configuration of system resources, you can restore by uploading the configuration files. Configuration files for system resources are available on Paragon Insights server and [GitHub](#). See "[Upload Resources](#)" on [page 235](#) for more information.

NOTE: If you edit a system resource and later restore it to the original configuration, Paragon Insights continues to display the resource status as modified.

Edit a Resource

To edit a resource:

1. Select **Configuration > Resources**.
The Resources page appears.
2. Do one of the following:
 - Select the resource from Resources table and click edit (pencil icon).
 - Select the resource on the visual panel and click **Edit** in the information pane of the resource.

The Edit Resources page appears.

3. Make changes to the configuration.

See ["Add Resources for Root Cause Analysis" on page 212](#) for editing resource configuration.

See ["Configure Dependency in Resources" on page 215](#) for editing dependency configuration.

4. Click **Save & Exit**.

Do one of the following:

- Click **Save and Deploy**.

Paragon Insights saves and deploys the configuration to generate smart alerts.

- Click **Save**.

Paragon Insights saves the configuration but does not generate smart alerts based on the edits in resource configuration.

You can view changes in the information pane when you click on the edited resource.

SEE ALSO

[Delete Resources and Dependencies | 237](#)

Edit Resource Dependency

To edit a resource:

1. Select **Configuration > Resources**.

The Resources page appears.

2. Click on the arrow that shows dependency from the resource you want to edit to a parent resource.

The dependency page of the child resource appears.

3. Edit the dependency configuration.

See ["Configure Dependency in Resources" on page 215](#) for editing dependency configuration.

4. Click **Save & Exit**.

Do one of the following:

- Click **Save and Deploy**.

Paragon Insights saves and deploys the configuration.

- Click **Save**.

Paragon Insights saves the configuration but does not deploy it.

If you changed the dependency type or Term name, you can view the changes in dependency information when you hover over the arrow. If you changed the depends on (parent) resource, you can view the edited resource connected to the new parent resource.

RELATED DOCUMENTATION

| [Delete Resources and Dependencies](#) | 237

Filter Resources

The Resource Dependency map (visual panel) has many system resources. The visual panel expands as you add other resources. You can focus on select resources and their dependencies through a keyword search for a resource in the Resources page and by filtering resources.

When you filter or search a resource, the other resources gray out on the visual panel and in the Resources table.

To filter a resource:

1. Select **Configuration > Resources**.
The Resources page appears.
2. Click the filter icon (funnel) on the visual panel.
3. Click **Add Filter** in the menu.
The Add Criteria page appears.
4. Enter the details as described in [Table 17 on page 235](#).
You can see the filtered resources in the visual panel and the resources table.
5. (If you added multiple criteria) Do one of the following:
 - a. Select **AND** if you want Paragon Insights to filter resources based on both the criteria.
 - b. Select **OR** if you want Paragon Insights to filter resources based on either criterion.
6. Click **Save** if you want to reuse the filter you set.
The Save Filter page appears.
7. Enter a name for the filter.
8. Toggle the default switch on if you want that filter to appear first on the saved filter list.
9. Click **OK**.
10. Click the filter icon (funnel) icon to view all your saved filters.

If you click the filter icon (funnel) after saving filters, you can view only the saved filters and not the Add Filter menu. You must hide filters to view the **Add Filter** menu.

Table 17: Attributes in Add Criteria Page

Field	Description
Field	Select Topic Name, Resource Name, or Keys to filter the resources.
Condition	You can enter if the filter criterion includes, equal to, or not equal to your input in Value field.
Value	Enter a topic's name, resource's name, or a key name. The entry in the Value field depends on your selection in the Field.

RELATED DOCUMENTATION

[Edit Resources and Dependencies](#) | 232

Upload Resources

You can upload resource configuration from the Resources page (**Configuration > Resources**). You can upload one resource file at a time but add multiple resource configurations in a resource file. Ensure that .resource is the file extension of a resource file.

If you added different resource configurations in a resource file, each configuration appears as a distinct resource on the Resources page. If a resource you add in the resource file already exists in Paragon Insights, the new configuration of that resource overrides the existing configuration when you upload the .resource file.

NOTE: The name of each resource inside a resource configuration file (.resource file) must be unique.

To upload a resource file:

1. Click **Configuration > Resources**.

The Resources page appears.

2. On the visual panel, click **Upload Resource File**.

An Upload Resource File page appears.

3. Click **Browse** and select the resource file saved in your local system.

4. Click **OK**.

You can see a confirmation message after the resource uploads successfully. The new resource or resources appear on the Resources page (visual panel).

RELATED DOCUMENTATION

[Download Resources | 236](#)

[Clone Resources | 236](#)

Download Resources

To download a resource configuration:

1. Click on the resource that you want to download.

A page that shows details of the resource appears.

2. On the page, click the download (down arrow) icon.

3. Click **Save File** and **OK** in the pop-up window to save the resource.

The resource configuration is downloaded to your local system as a compressed tar file.

RELATED DOCUMENTATION

[Upload Resources | 235](#)

[Clone Resources | 236](#)

Clone Resources

To clone a resource:

1. Click on the resource that you want to clone.

A page that shows details of the resource appears.

2. On the page, click the copy icon.

The Clone Resource page appears with the resource and dependency configurations of the resource you cloned.

3. Enter a name for the new resource in the Resource Name field.

4. Modify resource configuration details, if necessary.

See ["Add Resources for Root Cause Analysis" on page 212](#) for details about the resource configuration fields.

5. Modify resource dependency details, if necessary.

See ["Configure Dependency in Resources" on page 215](#) for details about the dependency configuration fields.

6. Do one of the following:

- Click **Save and Deploy**.

Paragon Insights saves and deploys the resource and dependency configuration to generate smart alarms.

You can see the new resource on the Resources page.

- Click **Save**.

Paragon Insights saves the resource and dependency configuration. When you only save the configurations, smart alarms are not generated.

You can see the new resource on the Resources page.

RELATED DOCUMENTATION

[Upload Resources | 235](#)

[Download Resources | 236](#)

Delete Resources and Dependencies

IN THIS SECTION

● [Delete a Resource | 238](#)

NOTE: You cannot delete default (system-generated) resources.

Delete a Resource

To delete a resource:

1. Select **Configuration > Resources**.

The Resources page appears.

2. Do one of the following:

- a. Select the resource from Resources table and click delete (trash icon).

- b. Select the resource you want to delete on the resource visualization.

The information pane of the resource page appears.

- Click **Delete**.

A delete confirmation message appears.

3. Do one of the following:

- a. Click **Delete**.

Paragon Insights deletes the resource.

- b. Click **Delete and Deploy**.

Paragon Insights deletes the resource and deploy the configuration changes that occur as a result of the deletion.

SEE ALSO

| [Edit Resources and Dependencies](#) | 232

Delete Resource Dependency

To edit a resource:

1. Select **Configuration > Resources**.

The Resource Dependency Model page appears.

2. Click the resource dependency you want to delete on the resource visualization.

The information pane of the dependency appears.

3. Click **Delete**.

A delete resource dependency confirmation dialog appears.

4. Click **OK**.

Paragon Insights removes the dependency you selected.

RELATED DOCUMENTATION

[Edit Resources and Dependencies](#) | 232

Monitor Network Device Health Using Grafana

IN THIS SECTION

- [Grafana Overview](#) | 239
- [Access the Grafana UI](#) | 240
- [Run a Query](#) | 240
- [View Prepopulated Graphs](#) | 242
- [Back Up and Restore Grafana Data](#) | 243

Grafana Overview

Starting with Paragon Insights (formerly HealthBot) Release 4.1.0, you can use the Grafana user interface (UI) to view data of your network devices. Grafana UI renders data from Paragon Insights time

series database (TSDB). You can view this data in the form of charts, graphs, histograms, and heat maps. You can also query data and view the results from the Grafana UI.

In releases earlier than Paragon Insights Release 4.1.0, you could only create and view graphs from the **Charts** page of the Paragon Insights UI.

Grafana is an open-source data visualization tool. You use Grafana to create and view charts, graphs, and other visuals to help organize and understand data. Paragon Insights uses Grafana to render graphs that you create from the **Charts** page of the UI. With Release 4.1.0, you can create graphs and other visuals directly from the Grafana UI.

For more information on Grafana, see [Grafana Documentation](#).

Access the Grafana UI

You can access Grafana UI by selecting **Monitor > Graphs > Grafana** from the Paragon Insights UI.

You can access Grafana by logging into the Paragon Insights UI. All logged in users can access and use Grafana and its features by clicking **Monitor > Graphs > Grafana**.

You use dashboards to monitor the status and health of devices. Follow these steps to view existing dashboards or create new dashboards from the Grafana UI:

- To create a new dashboard from the Grafana UI, click the plus (+) icon > **Create > Dashboards > Add Panel** from the right-nav panel.
- To view existing dashboards from the Grafana UI, click **Dashboards** from the right-nav panel.

Run a Query

You can run queries from the Grafana UI. For more information, see [Query a Data Source](#).

When you create a chart, the data is rendered from the TSDB database, an open-source time series database. Information sent to and received from the TSDB database is in the influxDB format. With Release 4.1.0, Paragon Insights also supports Grafana's open-source influxDB plugin. After you run a query from the Grafana UI, the open-source influxDB sends a backend request to the Paragon Insights TSDB. The Paragon Insights TSDB sends the response in influxDB format.

In Paragon Insights Release 4.1.0, when you run a query from the Grafana UI using influxDB as a data source, you can only view data of a specific device of a device group. You use syntax parameters such as FROM and SELECT that the influxDB recognizes. You cannot view aggregate data of a device group, or view

aggregate data of multiple device groups using the influxDB plugin. However, with Release 4.2.0, Paragon Insights supports the Juniper Paragon Insights TSDB plugin.

The Juniper Paragon Insights TSDB plugin is based on the influxDB plugin but also supports the ability to view aggregate data of a device group, or view aggregate data of multiple device groups. You can view aggregate data by using Syntax parameters such as DEVICE, DEVICE GROUPS, and MEASUREMENT. The default data source for Paragon Insights in Grafana is based on the Juniper Paragon Insights TSDB plugin.

For more information on supported syntax parameters, see [Table 18 on page 241](#).

Figure 58: Querying from Grafana UI

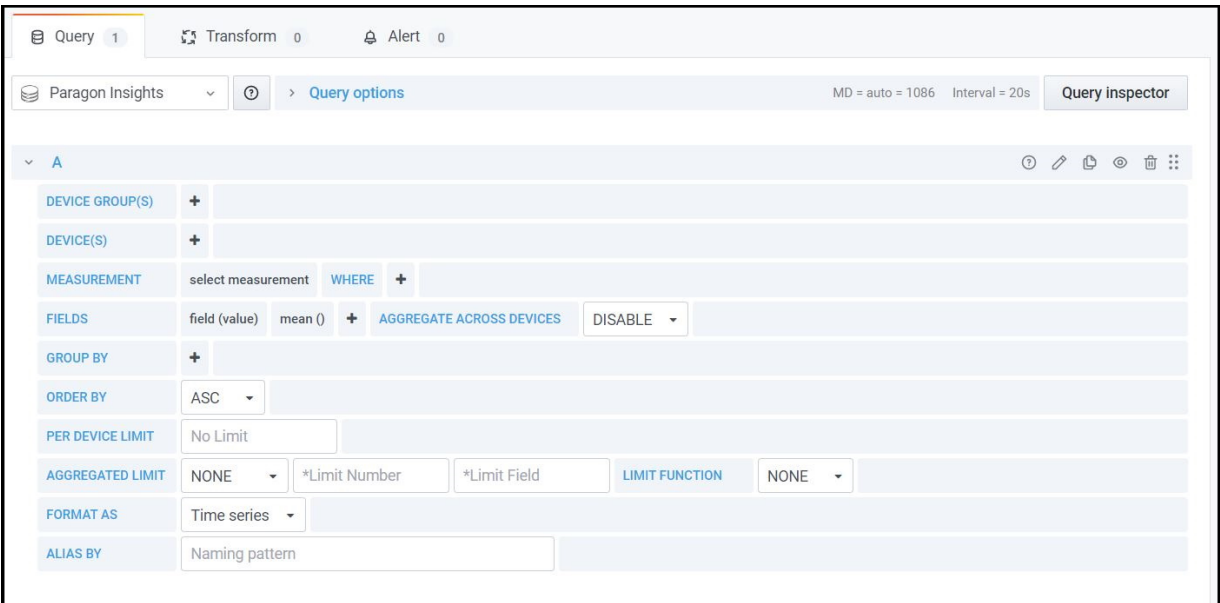


Table 18: Examples of InfluxDB and TSDB Syntax Parameters

InfluxDB Querying Options	TSDB Querying Options
FROM—Select the data source (device, device group, topic/rule).	DEVICE—Select one or more than one device for this query.
SELECT—Select the data field, and apply aggregation and transformation types to the data.	DEVICE GROUP—Select one or more than one device group for this query.

Table 18: Examples of InfluxDB and TSDB Syntax Parameters *(Continued)*

InfluxDB Querying Options	TSDB Querying Options
GROUP BY—Specify how to group data based on KPI keys.	MEASUREMENT—Select the Paragon Insights topic or rule name.
FORMAT AS—Specify if you want to format as table, time series, or log.	WHERE—Filter data based on tags and fields.
	FIELDS—Select the fields to which you want to apply the aggregation.
	GROUP BY—Specify how to group data based KPI keys.
	FORMAT AS—Specify if you want to format as table, time series, or log.
	PER DEVICE LIMIT—Set the per-device limit of the base query for each device (database) that you have selected.
	AGGREGATED LIMIT—Set the aggregate limit for the device (databases) after the base query is run.
	ALIAS BY—Rename a field. For example, if you select mean as an aggregation function for a field named bps , you can give it an alias name called mean_bps .

View Prepopulated Graphs

Starting with Paragon Insights Release 4.3.0, you can view prepopulated graphs from the Grafana dashboard.

You can view prepopulated graphs from the **Dashboards** section of the Grafana **Home** page. You can also view these prepopulated graphs by clicking **Dashboard > Manage > Paragon Insights Cluster Health**. With Paragon Insights Release 4.3.0, the following graphs are available by default:

- **CPU Usage**—View cluster-wise or node-wise information on CPU (%) usage at different levels.

- **Disk Read Usage (r_await)**—View cluster-wise or node-wise information on average time taken for disk reads to be served.
- **Disk Write Usage (w_await)**—View cluster-wise or node-wise information on average time taken for disk writes to be served.
- **Node Memory Available**—View information on free memory available on a node in a selected cluster.

Click the name of a graph to view more information of that graph. You can view a prepopulated graph of all nodes in a cluster at a time, and also view prepopulated graphs of one or more nodes in a cluster.

Back Up and Restore Grafana Data

Back Up Grafana Data

1. Click **Administration > Backup**.
The **Backup** page appears.
2. Click **Backup Grafana** to backup Grafana data. The **Confirm Backup** pop-up appears.
3. Click **Ok** to confirm.

Restore Grafana Data

1. Click **Administration > Restore**. The **Restore** page appears.
2. Click **Choose file** and select the backed-up Grafana data file from your local system; click **Ok** to confirm selection.
3. Click **Restore Grafana** to restore Grafana data.
4. Click **Ok** to confirm.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.3.0	Starting with Paragon Insights Release 4.3.0, you can view prepopulated graphs from the Grafana dashboard.
4.1.0	Starting with Paragon Insights (formerly HealthBot) Release 4.1.0, you can use the Grafana user interface (UI) to view data of your network devices.

RELATED DOCUMENTATION

| [Monitor Device and Network Health](#) | 172

Understanding Action Engine Workflows

NOTE: Action engine workflow is a Beta feature.

When creating rules, Paragon Insights includes the ability to run user-defined actions (UDAs) as part of a trigger. UDAs are Python scripts that can be configured to be triggered by a Paragon Insights rule.

You cannot track and manage UDAs, pause an action, retry a failed action, and resume a paused action. However, with Release 4.1.0, Paragon Insights supports action engine workflow monitoring. An action engine workflow is an action engine that you can use to configure a set of tasks (instances). You can configure action engine workflows, monitor existing action engine workflows, and manage action engine workflow instances from the Paragon Insights GUI.

You can view action engine workflows that you created by using the CLI in the Paragon Insights GUI. You can perform the following actions from the CLI:

- run NETCONF command
- run arbitrary executable files such as Python, Bash, Ruby
- run a command to send notification messages

RELATED DOCUMENTATION

| [Manage Action Engine Workflows](#) | 244

Manage Action Engine Workflows

NOTE: Action engine workflow is a Beta feature.

See "[Understanding Action Engine Workflows](#)" on page 244 for more information on action engine workflows.

You can add new workflows from the **Configuration > Action Engine** page of the Paragon Insights GUI. You can manage existing action engine workflows, and action engine workflow instances from the **Monitor > Action Engine** page.

Add an Action Engine Workflow

Follow these steps to add an action engine workflow:

1. Click **Configuration > Action Engine**.

The **Workflows** page appears.

2. Click plus (+) icon to add an action engine workflow.

The **Add New Workflow** page appears. The **General** tabbed page is displayed by default.

3. Enter the following information in the **General** tabbed page :

- a. Enter a name for the action engine workflow in the **Name** text box.
- b. Enter a description for the action engine workflow in the **Description** text box.
- c. Select an entry task from the **Entry Task** drop-down list.

NOTE: You have to add a task before you can select the task from the **Entry Task** drop-down list. To add a task, see Step 4.

An entry task is the first task that is executed when you run an action engine workflow.

- d. Select an exit task from the **Exit Task** drop-down list.

NOTE: You have to add a task before you can select the task from the **Exit Task** drop-down list. To add a task, see Step 4.

An exit task is the last task (for example, a clean up task) that is executed at the end of an action engine workflow sequence.

4. Click **Tasks** to view the **Task** tabbed page.

5. Click (+) to add a task.

Enter the following information:

- a. Enter a name for the task in the **Name** text box.
- b. Enable or Disable the **Parallel** toggle button.

Enable the **Parallel** toggle button to run all steps in a task simultaneously.

c. Follow these steps for Paragon Insights Release 4.1.0:

- i. Click (+) to add a new step.

The **Add Step** overlay page appears.

- ii. Enter a name for the step in the **Name** text box.
- iii. Enter a value for **Command Tag** and **Command Field** in the **CLI Command Settings** section.
- iv. Click **Ok** to confirm

Follow these steps for Paragon Insights Release 4.2.0:

- i. Click the (+) icon to add a step.

A row is added to the **Steps** section.

In the row that is added,

1. Enter a name for the step in the **Name** text box.
2. Enter a description for the step in the **Description** text box.
3. Select dependencies from the **Dependencies** drop-down list.
4. Select a step type from the **Type** drop-down list.

- ii. Click **Edit Commands** to add a new command.

The **ADD/EDIT COMMANDS** pop-up appears.

Enter the following information:

1. Click **+Add New Command** to add a new command.

The **New Command** section appears in the **ADD/EDIT COMMANDS** pop-up.

2. Enter a value for the command tag in the **Command Tag** text box.
3. Enter a value in the **Commands** list box. You can select more than one command.

Click **X** to remove the command that you selected.

4. Enter a value in the **Arguments** list box. You can select more than one argument.

Click **X** to remove the argument that you selected.

5. Enter a value in the **Device** list box. You can select more than one device.

Click **X** to remove the device that you selected.

6. Enter a value in the **Device Group** list box. You can select more than one device group.

Click **X** to remove the device group that you selected.

7. Enter a value in the **Environment** list box.

8. Select an output from the **Output Type** list box.

9. Enable or Disable the **Ignore** toggle button.

Enable the **Ignore** toggle button to ignore steps.

10. Set repeat parameters in the **Repeat** field.

You can determine if you want to repeat a failed step or not.

11. The default delay value displayed is 10 seconds.

After you have set repeat parameters to repeat a step that has failed, there is a delay of 10 seconds before the step is repeated again.

12. Click **OK** to confirm.

The new command is added.

- iii. Click **Edit Conditions** to add new conditions.

The **ADD/EDIT CONDITIONS** pop-up appears.

Follow this procedure to set conditions to run a step:

1. Enter the conditions in the **Conditions** text box.
2. Select a condition type from the **Conditions Type** list box.
3. Enter a description for the condition in the **Condition Description** text box.
4. Click **OK** to confirm.

The new conditions are added

- iv. Click **Edit Inputs** to add new inputs.

The **ADD/EDIT INPUTS** pop-up appears.

Enter the following information:

1. Click the (+) icon to add new input.
2. Enter a name for the input in the **Name** field.

- 3. Enter a value for the input in the **Value** field.
- 4. Click **OK** to confirm.

The **operation is successful** message is displayed in the **ADD/EDIT INPUTS** pop-up.

- 5. Click **Close** to close the **ADD/EDIT INPUTS** pop-up.

- v. Click **Edit Output** to add new outputs.

The **ADD/EDIT OUTPUT** pop-up appears.

Enter the following information:

- 1. Click the **(+)** icon to add new input.
- 2. Select a name for the output from the **Name** drop-down list.
- 3. Enter a description for the output in the **Description** text box.
- 4. Enter a value for the command tag in the **Command Tag** field.
- 5. Select output type from the **Output Type** drop-down list. See [Table 19 on page 248](#).
- 6. The field displayed depends on the output type that you have selected. See [Table 19 on page 248](#).

Table 19: Output Type and Corresponding Fields

Output Type	Field
Grok	Pattern
XML	XPath
JSON	JQ path
Artifact	Path
Regex	Pattern
Result	

7. Click **OK** to confirm.

The **operation is successful** message is displayed in the **ADD/EDIT OUTPUT** pop-up.

8. Click **Close** to close the **ADD/EDIT OUTPUT** pop-up.
- d. Click the **✓** icon to add this row to the **Steps** section.
6. Click the **Arguments** tab.
7. On the **Arguments** tabbed page, click the plus (+) icon to add a new argument.
Enter the following information:
 - Enter a name for the argument in the **Name** text box.
 - Click **Ok** to confirm.
8. Do any one of the following:
 - Click **Save** to save the action engine workflow.
 - Click **Save & Deploy** to save and deploy the action engine workflow.

You have now added and deployed an action engine workflow. To monitor the action engine workflows that you have added, see **Monitor > Action Engine**.

Run an Action Engine Workflow

After you add an action engine workflow from the **Configuration > Action Engine** page of the UI, you can run the action engine workflow by following these steps:

1. Click **Monitor > Action Engine**.

The **Workflows Monitor** page appears.

2. Select the action engine workflow you want to run by selecting the check box next to the name of the action engine workflow.
3. Click **Run Workflow**.

The **Run Workflow <workflow name>** pop-up appears.

4. In the **Run Workflow <workflow name>** pop-up that appears, you can:
 - View the list of preconfigured arguments for the action engine workflow.
 - Configure additional arguments.

To configure additional arguments:

- a. Click (+).

The **Additional Arguments** fields that you can configure are displayed.

- i. Enter a name in the **Name** text box to identify this additional argument.

The name you enter must be in the `[a-zA-Z][a-zA-Z0-9_-]*$` regular expression format. This format states that the first character of the name can start with **a-z** or **A-Z**. The name cannot start with a number or a special character. However, you can use numbers, `_`, and `-` within the name. Maximum length is 64 characters.

- ii. Select an additional argument type from the **Type** drop-down list.

Available options: string, list, password, device, device-group, network-group

- iii. Select a value from the options available.

The options you can choose from depend on the additional argument **Type** that you have selected. You can add one or more than one arguments.

5. Click **OK** to confirm settings and run the action engine workflow.

Stop an Action Engine Workflow Instance

You can stop an action engine workflow instance that is currently running.

NOTE: You cannot resume (restart) an instance that you have stopped.

To stop an instance:

1. Click **Monitor > Action Engine**.

The **Workflows Monitor** page appears.

2. Click a action engine workflow to view the instances listed under it.
3. Select the instance that is currently running by selecting the check box next to the name of the instance.
4. Click **Stop** to stop the instance.

Resume a Suspended Action Engine Workflow Instance

You can resume (restart) an action Engine workflow instance that has been suspended. To resume a suspended instance:

1. Click **Monitor > Action Engine**.

The **Workflows Monitor** page appears.

2. Click an action engine workflow to view the instances listed under it.

3. Select a suspended instance by selecting the check box next to the name of the instance.
4. Click **Resume** to restart the instance.

Filter Action Engine Workflow Instances

To filter action engine workflow instances within an action engine workflow:

1. Click **Monitor > Action Engine**.

The **Workflows Monitor** page appears.

2. Click **Filter**, and then click **Add Filter** from the **Filter** drop-down list.

The **Add Criteria** pop-up appears.

3. Enter the following information in the **Add Criteria** pop-up.

- a. Select the field that you want to apply the filter to, from the **Field** drop-down list .
- b. Select the conditions that you want to apply to the field, from the **Condition** drop-down list .
- c. Enter start and/or finish time that you want to apply to the filter, in the **Value** text box.

4. Click **Add** to apply the filter.

Delete an Action Engine Workflow

To delete an action engine workflow:

1. Click **Configuration > Action Engine**.

The **Workflows** page appears.

2. Select the action engine workflow you want to delete by selecting the check box next to the name of the instance.

3. Click the **Delete** icon.

The **Delete Workflow** pop-up appears.

4. In the **Delete Workflow** pop-up that appears, click **Ok** to delete the action engine workflow.

RELATED DOCUMENTATION

| [Understanding Action Engine Workflows](#) | 244

Alerts and Notifications

IN THIS SECTION

- [Generate Alert Notifications | 252](#)
- [Manage Alerts Using Alert Manager | 260](#)
- [Stream Sensor and Field Data from Paragon Insights | 264](#)

Generate Alert Notifications

IN THIS SECTION

- [Configure a Notification Profile | 253](#)
- [Enable Alert Notifications for a Device Group or Network Group | 259](#)

Paragon Insights (formerly HealthBot) generates alerts that indicate when specific KPI events occur on your devices. To receive Paragon Insights notifications for these KPI events, you must first configure a notification profile. Once configured, you can enable alert notifications for specific device groups and network groups.

Paragon Insights supports the following notification delivery methods:

- Web Hook
- Slack
- Kafka Publish
- Microsoft Teams (HealthBot Release 2.1.0 and later)
- Email (HealthBot Release 2.1.0 and later)
- Advanced Message Queuing Protocol (AMQP) Publish (Paragon Insights Release 4.0.0 and later)

This section includes the following procedures:

Configure a Notification Profile

A notification profile defines the delivery method to use for sending notifications.

1. Click the **Settings > System** option in the left-nav bar.
2. Click the **Notification** tab on the left of the window. click the add notification button (**+ Notification**).
3. Click the **+ Notification** button
4. In the **Add Notification** window that appears, configure the notification profile:

Attributes	Description
Name	Enter a name.
Description	(Optional) Enter a description.
Notification Type	<p>Select a notification type:</p> <ul style="list-style-type: none"> • Web Hook • Slack • Kafka Publish • Microsoft Teams (HealthBot 2.1.0 and later) • EMails (HealthBot 2.1.0 and later) • Advanced Message Queuing Protocol (AMQP) Publish (Paragon Insights Release 4.0.0 and later) <p>Notification type attributes vary based on notification type selected. See below for details.</p>

5. Click **Save and Deploy**.

NOTIFICATION TYPE DETAILS

Web Hook

- **URL** URL at which the Web Hook notification should be posted.
- **Username** (Optional) Username for basic HTTP authentication.

- **Password** (Optional) Password for basic HTTP authentication.

Slack

- **URL** URL at which the Slack notification should be posted. Different from your Slack workspace URL. Go to <https://slack.com/services/new/incoming-webhook> and sign in to your Slack workspace to create a Slack API endpoint URL.
- **Channel** Channel on which the notification should be posted.

Kafka Publish

- **Bootstrap Servers** Add Kafka host:port pairs from the drop-down list to establish the initial connection to the Kafka cluster.
- **Topic** (Optional) Name of the Kafka topic to which data will be published. By default, the Kafka topic naming convention for device group alert notifications is *device-group.device-id.topic.rule.trigger*.

Depending on the authentication protocols being used, the required authentication parameters are as follows:

Protocol	Required Parameters
SASL/SSL	Username, password and certificate
SASL/Plaintext	Username and password
SSL	Certificate
Plaintext	None

- **Username** Username for SASL/SSL or SASL/plaintext authentication.
- **Password** Password for SASL/SSL or SASL/plaintext authentication.

- **Certificate** Kafka server's CA certificate. Choose file from the drop-down list.
- **Upload Certificate** Location from where the Kafka server's CA certificate will be uploaded. Click **Choose files** and navigate to the file location. File should be in Privacy Enhanced Mail (PEM) format.

Microsoft Teams

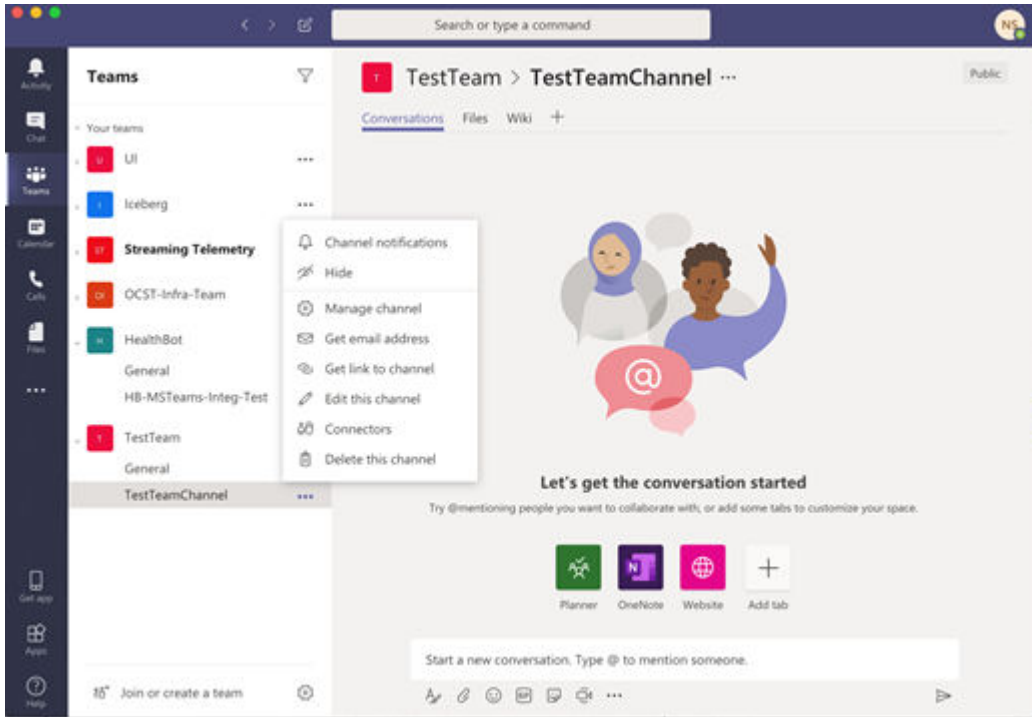
As of HealthBot Release 2.1.0, you can send Paragon Insights (formerly HealthBot) notifications to Microsoft Teams. Teams can provide a connector which you can add to Paragon Insights to enable the connection.

Configuration workflow:

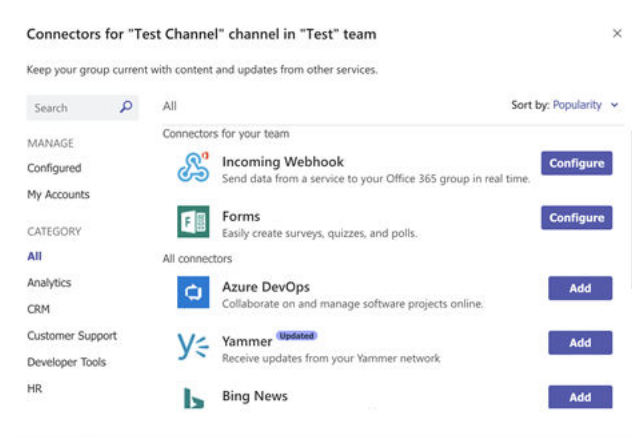
- In Teams, create a new connector set as an incoming webhook.
- Copy the URL provided by Teams.
- In Paragon Insights, configure a notification profile that sends to Microsoft Teams.
- Apply the notification profile to a device group.

To configure MS Teams notifications:

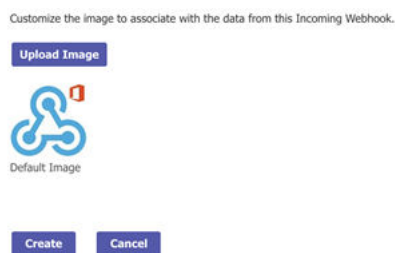
1. In Teams, select the desired channel and click the ellipsis (...).
2. In the menu that appears, click **Connectors**.



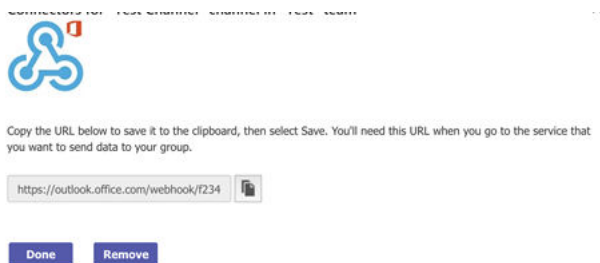
3. Use the Incoming Webhook option and click **Configure**.



4. On the next page, click **Create**.



5. Once the web hook is successfully created, copy the provided URL.



6. In Paragon Insights, go to the **Settings > System** page select the Notification tab.
7. Click the **+ Notification** button.
8. Configure the notification profile as follows:
 - Name - Enter a profile name.
 - Notification Type - select **Microsoft Teams**.
 - Channel - Paste the URL provided by the Teams UI above.
9. Click **Save and Deploy**.
10. Apply the notification profile to a device group or network group as shown in ["Enable Alert Notifications for a Device Group or Network Group" on page 259](#)

EMails

As of HealthBot Release 2.1.0, you can send Paragon Insights (formerly HealthBot) notifications by email. By default, email notifications cover all running playbooks and rules for the device group or network group to which they are applied, however you can narrow the focus by selecting specific rules.

NOTE: Paragon Insights includes its own mail transfer agent (MTA), so no other mail server is required.

Configuration workflow:

- In Paragon Insights, configure a notification profile that sends to email.
- Apply the notification profile to a device group.

To configure email notifications:

1. In Paragon Insights, go to the **Settings > System** page.
2. Select the **Notification** tab and click the **+ Notification** button.

3. Configure the notification profile as follows:

- Name - Enter a profile name.
- Notification Type - Select **Emails**.
- Email Addresses - Enter an email address and click **Add <email-address>**; repeat for more email addresses.
- (Optional) Rule filters - To narrow the scope of what triggers an email, define rule filters. Enter a filter and click **Add <rule-filters>**; repeat for more filters.
 - Format is topic/rule; can use regular expressions
 - Example: **interface.statistics/check-interface-flaps** sends notifications only for the rule check-interface-flaps.
 - Example: **system.processes/.***, **system.cpu/.***, and **interface.statistics/.*** sends notifications for all rules under the topics system.processes, system.cpu, and interface.statistics.

4. Click **Save and Deploy**.

5. Apply the notification profile to a device group or network group as shown in ["Enable Alert Notifications for a Device Group or Network Group" on page 259](#)

AMQP Publish

If you select AMQP Publish as the notification type, you have to specify the following:

- Host (mandatory)—Specify a valid hostname or the IP address of the AMQP server.
- Port (mandatory)—Specify the listener port of the AMQP server.
- Exchange(mandatory)—Specify the name of the exchange or the routing agent of the AMQP server on which the connection must be instantiated.
- Virtual Host(optional)—Specify the virtual host of the AMQP server on which the connection must be instantiated. If you do not specify, the default value(/) is used.
- Routing Key(optional)—Specify the routing key. The routing key is a message attribute that the exchange refers to when deciding how to route the message to the queue.

NOTE: If you have not configured the routing key, the following are the default value:

- For sensor or raw data, *<device-group>.<device>.sensors*

- For field data, *<device/network-group>.<device>.<topic>.<rule>.fields*
- For trigger/alert data, *<device/network-group>.<device>.<topic>.<rule>.<trigger>*

In case of a network group, *<device>* is rendered as "-".

- Username—Specify the username for the Simple Authentication Security Layer (SASL) authentication.
- Password—Specify the password for the SASL authentication.
- CA Profile—Select the CA profile from the drop-down list. For more information on CA Profiles and local certificates, see ["Configure a Secure Data Connection for Paragon Insights Devices" on page 286](#).
- Local Certificate—Select the local certificate from the drop-down list. For more information on CA Profiles and local certificates, see ["Configure a Secure Data Connection for Paragon Insights Devices" on page 286](#)
- Server Common Name—Specify the server common name that is used while creating a certificate.

Enable Alert Notifications for a Device Group or Network Group

To enable alert notifications for a device group or network group:

1. For Device Groups, select the **Configuration > Device Group** page from the left-nav bar.
For Network Groups, select the **Configuration > Network** page from the left-nav bar.
2. Click the name of the device group or network group for which you want to enable alert notifications.
3. Click the **Edit (Pencil)** icon.
4. Scroll down to the **Notification** section in the pop-up window and click the caret to expand that section.
5. Select a destination for any alert level (Major, Minor, or Normal) that you want. Notification can be sent to zero or more defined destinations for each alert level.
6. Click **Save and Deploy**.

Manage Alerts Using Alert Manager

IN THIS SECTION

- [Viewing Alerts | 260](#)
- [Manage Individual Alerts | 262](#)
- [Configure Alert Blackouts | 263](#)

You can use the Alert Manager feature to organize, track, and manage KPI event alert notifications received from Paragon Insights devices. The Alert Manager does not track alerts by default; it is populated based on which device groups or network groups are configured to send the notifications.

Viewing Alerts

To view the alerts report table, go to the **Monitor > Alerts** page in the left-nav bar. Note that Alert Manager consolidates duplicate alerts into one table entry and provides a count of the number of duplicate alerts it has received.

Starting with release 4.2.0, Paragon Insights generates smart alerts if you configured resources and dependencies. To configure resources, click Resource Discovery at the top right corner of the Alerts page.

Smart alerts combine alerts from different rules into a collapsible tree structure. The main alert in the tree displays the root cause that triggered the other alerts in the tree. See "[Understand Resources and Dependencies](#)" on [page 206](#) for more information.

The following table describes the alerts report table attributes.

Attributes	Description
Severity	Severity level of the alert. Options include: <ul style="list-style-type: none">● Major● Minor● Normal

(Continued)

Attributes	Description
Status	Management status of the alert entry. Options are Open, Active, Shelved, Closed, and Ack. The statuses available in the Status pull-down menu in the top row of the table only include statuses of alerts visible in the table and those allowed by the status filter above the table.
Last Received	Time the alert was last received.
Dupl.	Duplicate count. Number of times an alert with the same event, resource, environment, and severity has been triggered.
Topic	Device component topic name.
Resource	Device name.
Event	Name of the rule, trigger or field, and event with which the alert is associated.
Text	Health status message.

The following table describes the main features of the alerts report table:

Feature	Description
Sort the data by ascending or descending order based on a specific attribute.	Click on the name of the data type at the top of the column by which you want to sort.
Filter the data based on the device group.	In the drop-down list at the top left corner of the page, select a device group by which to filter.

(Continued)

Feature	Description
Filter the data based on the alert status.	<p>Two options:</p> <ol style="list-style-type: none"> 1. In the drop-down list above the table at the top of the page, select one or more status types on which to filter. Options are open, active, shelved, closed, and ack. You can filter on multiple status types. 2. In the drop-down list at the top of the Status column, select a status type by which to filter. Note that if there are status types shown in the filter list at the top of the report, then the status column can only show those status types.
Filter the data based on the severity, topic, or resource	In the associated drop-down list for each attribute at the top of the table, select an option by which to filter.
Filter the data based on a keyword.	In the associated text box under the Event or Text attribute name at the top of the table, enter the keyword on which to filter.
Filter the data based on date or time received.	In the Last Received field, enter a date and time in the format: <Day> <DD> <Mon> <HH:MM>
Navigate to a different page of the table.	<p>Two options:</p> <ol style="list-style-type: none"> 1. At the bottom of the table, click the Previous or Next buttons. 2. At the bottom of the table, select the page number using the up/down arrows (or by manually entering the number) and then press Enter.
Change the number of rows displayed.	At the bottom of the table, choose the number of rows to display in the drop-down list. The table displays 20 rows by default.
If the data in a cell is truncated, view all of the data in a cell.	Resize the column width of the cell by dragging the right side of the title cell of the column to the right.

Manage Individual Alerts

You can view detailed information about each alert in the alerts report table. You can also assign a management status (such as open, ack, and close), and apply simple actions (such as shelve and delete) to each alert.

To manage individual alerts:

1. Go to the **Monitor > Alerts** page from the left-nav bar to open the alert report table.
2. Click on a single alert entry in the table. The Alert Details pane displays detailed information about the alert.

The following table describes the set of buttons at the top of the Alert Details pane:

Button	Description
Open	Changes the status of the alert to Open.
Shelve	Removes the alert from the table for a set amount of time. Time options are 1, 2, 4 and 8 hours. Click Unshelve to disable this feature.
Ack	Changes the status of the alert to Ack. The Ack status removes the alert from the table, but the alert still remains active.
Close	Changes the status of the alert to Closed. The Closed status indicates that the severity level of the alert is now Normal.
Delete	Deletes the alert from the table.

Configure Alert Blackouts

You can configure blackout periods to suppress or mute alerts during, for example, scheduled downtimes.

To configure blackouts:

1. Click the **Settings > System** page from the left-nav bar.
2. Select the **Alert** tab on the left side of the page.
3. In **Alert Blackout Settings**, click the **+ Alert Blackout** button.
4. Enter the necessary values in the text boxes for the blackout configuration.

The following table describes the attributes in the **Add an Alert Blackout** pane:

Attributes	Description
Duration	Select a start and end date and time for the blackout.
Device Group	Select a device group from the drop-down list to which to apply the blackout configuration.
Attribute	(Optional) Specify an attribute from the drop-down list to which to apply the blackout configuration.
Value	<p>(Optional) If a blackout attribute is specified, provide an associated value (as shown in the alerts report table). Only the alerts that match this attribute value exactly will be suppressed from the alerts report table.</p> <p>NOTE: For the Resource-Event attribute, you must specify a resource from the drop-down list, as well as specify an Event value. Only the alerts generated by the specified resource that match this Event value exactly will be suppressed from the alerts report table.</p>

5. Click **Save** to save the configuration.
6. (Optional) Click the **Delete** button to delete a blackout configuration.

Stream Sensor and Field Data from Paragon Insights

IN THIS SECTION

- [Configure the Notification Type for Publishing | 264](#)
- [Publish Data for a Device Group or Network Group | 267](#)

You can configure Paragon Insights to publish Paragon Insights sensor and field data for a specific device group or network group. You must first configure the notification type for publishing and then specify the fields and sensors that you want published.

Configure the Notification Type for Publishing

Paragon Insights supports Apache Kafka and AMQP for publishing sensor and field data.

You must first configure a Kafka publishing profile before you can start publishing sensor and field data for a specific device group or network group.

To configure a Kafka publishing profile:

1. Select the **Settings > System** page from the left-nav bar.
2. Click the **Notification** tab on the left part of the page.
3. In **Notification Settings**, click the **+ Notification** button.
4. Enter the necessary values in the text boxes and select the appropriate options for the Kafka publishing profile.

The following table describes the relevant attributes in the **Add a Notification Setting** and **Edit Notification Configuration** panes:

Attributes	Description
Name	Name of the notification.
Description	(Optional) Description of the notification.
Notification Type	Click the Kafka publish radio button.

(Continued)

Attributes	Description
Kafka Publish	<ul style="list-style-type: none"> • Bootstrap Servers Add Kafka host:port pairs from the drop-down list to establish the initial connection to the Kafka cluster. • Topic (Optional) Name of the Kafka topic to which data will be published. By default, the Kafka topic naming conventions are: <ul style="list-style-type: none"> • For device group field data, <i>device-group.device-id.topic.rule.fields</i> • For network group field data, <i>network-group.topic.rule.fields</i> • o For device group sensor data, <i>device-group.device-id.sensors</i> <p>Depending on the authentication protocols being used, the required authentication parameters are as follows:</p> <ul style="list-style-type: none"> • SASL/SSL—Username, password and certificate • SASL/Plaintext—Username and password • SSL—Certificate • Plaintext—None <p>Required authentication parameters are:</p> <ul style="list-style-type: none"> • Username Username for SASL/SSL or SASL/plaintext authentication. • Password Password for SASL/SSL or SASL/plaintext authentication. • Certificate Kafka server's CA certificate. Choose file from the drop-down list. • Upload Certificate Location from where the Kafka server's CA certificate will be uploaded. Click Choose files and navigate to the file location. File should be in Privacy Enhanced Mail (PEM) format.

5. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.
6. Apply the Kafka publishing profile to a device group or network group. For more details, see the ["Publish Data for a Device Group or Network Group" on page 267](#) section.

Publish Data for a Device Group or Network Group

To publish Paragon Insights sensor or field data for a device group or network group:

1. For Device Groups, select the **Configuration > Device Group** page from the left-nav bar.
For Network Groups, select the **Configuration > Network** page from the left-nav bar.
2. Click the name of the device group or the network group to which you want to publish data.
3. Click the **Edit (Pencil)** icon.
4. Under **Publish**, select the appropriate Destinations, Field, or Sensor from the drop-down lists for the data you want to publish. To publish field or sensor data, you must configure a destination.

Parameter	Description
Destinations	<p>Select the publishing profiles that define the notification type requirements (such as authentication parameters) for publishing the data.</p> <p>To edit or view details about saved publishing profiles, go to the System page under the Settings menu option in the left-nav bar. The publishing profiles are listed under Notification Settings.</p> <p>NOTE: Only Kafka and AMQP publishing are currently supported.</p>
Field	Select the Paragon Insights rule topic and rule name pairs that contain the field data you want to publish.
Sensor	(Device group only) Select the sensor paths or YAML tables that contain the sensor data you want to publish. No sensor data is published by default.

5. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
2.1.0	As of HealthBot Release 2.1.0, you can send Paragon Insights (formerly HealthBot) notifications by email.

RELATED DOCUMENTATION

[Monitor Device and Network Health](#) | 172

Generate Reports

IN THIS SECTION

- [Generate On-Demand Reports | 269](#)
- [Generate Scheduled Reports | 270](#)
- [Create a Schedule Profile | 271](#)
- [Create a Report | 272](#)
- [Associate the Report to a Device Group or Network Group | 273](#)
- [View Reports | 273](#)
- [Create a Field Snapshot | 276](#)
- [Compare \(Diff\) Reports | 278](#)

You can generate Paragon Insights (formerly HealthBot) reports for device groups and network groups. These reports include alarm statistics, device or network health data, as well as device-specific information (such as hardware and software specifications).

Paragon Insights's reporting functionality allows you to:

- Send reports by email, save them on the Paragon Insights server, or download them to your local machine
- Schedule reports to run at regular intervals, or for a specific time
- Generate reports on demand (HealthBot Release 2.1.0 and later)
- Compare (diff) two reports (HealthBot Release 2.1.0 and later)
- Capture a snapshot of a specific set of fields at a given point in time (HealthBot Release 3.1.0 and later)

This section includes the following procedures:

- ["Generate On-Demand Reports" on page 269](#)
- ["Generate Scheduled Reports" on page 270](#)
- ["View Reports" on page 273](#)
- ["Create a Field Snapshot" on page 276](#)

- ["Compare \(Diff\) Reports" on page 278](#)

Generate On-Demand Reports

You can generate and download a report on demand for a device group or network group. As with regular report generation, formats supported include HTML or JSON, and you can download the report or receive it by email.

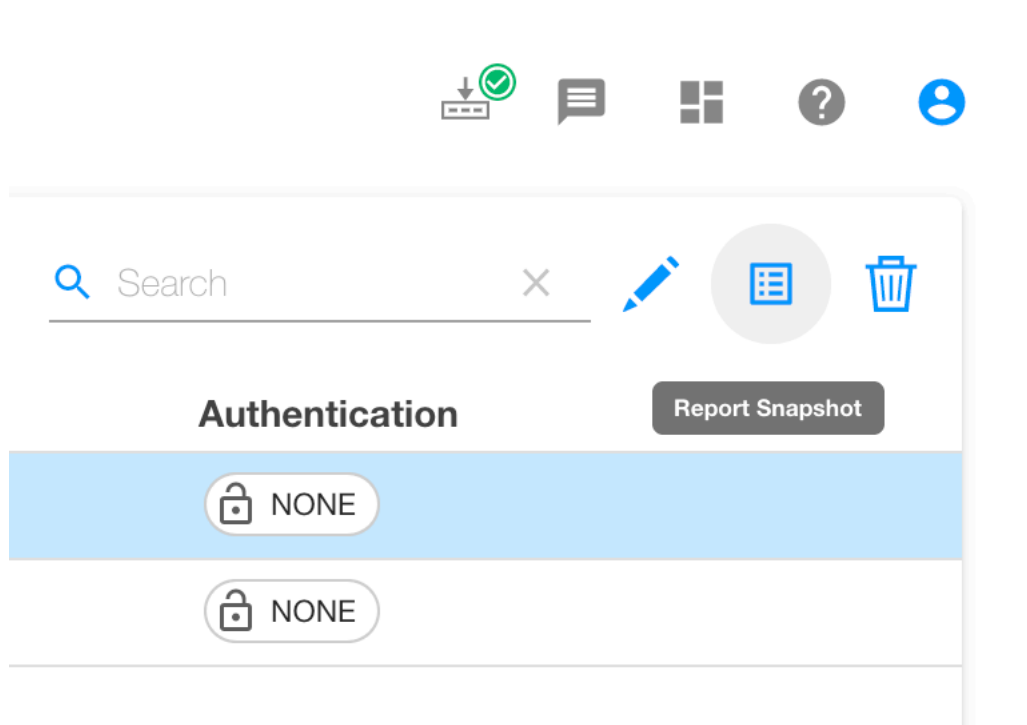
Once generated, you can re-download on-demand reports from the Reports page. These reports have the report name **HB_MANUAL_REPORT**.

1. For a device group, navigate to the **Configuration > Device Group** page from the left-nav bar and select the device group name from the list.

For a network group, navigate to the **Configuration > Network** page from the left-nav bar and select the network group name from the list.

2. Click the **Report Snapshot (Page)** icon in the upper right part of the page as shown in

Figure 59: Report Snapshot Button



3. On the page that appears, enter report generation details, including:

- Format - **HTML** or **JSON**
 - Destination Type - save to your computer (**disk**) or send to **email** (which also saves to disk)
 - If disk, specify the number of reports to save on the server before deleting older reports
 - If email, add target email address(es)
 - (Optional) Select graph canvases to include in the report.
 - (Optional) Select the desired graph panels to include in the report.
4. Click **Submit**.
 5. A dialog box appears allowing you to download the file. Additionally, if the destination is disk, Paragon Insights stores a copy of the report on the server. If the destination is email, Paragon Insights sends the report to the specified account.

Generate Scheduled Reports

The workflow to configure and generate scheduled reports is as follows:



Create a Destination Profile

1. Select the **Settings > System** page from the left-nav bar.
2. Click the **Destination** tab on the left of the page.
3. In the **Destination Settings** section, click the **+ Destination** button.
4. Specify the destination profile settings as appropriate.

The following table describes the attributes in the **Add a destination** and **Edit a destination** panes:

Attributes	Description
Destination Name	Enter a name. The name cannot be changed once saved.

(Continued)

Attributes	Description
Destination Type	Options include Email or Disk .
Email > Email Id	Enter an email address to which the report will be sent.
Disk > Maximum Reports	Specify how many versions of this report will be stored on the server. Older reports are deleted as newer reports are generated and saved.

NOTE: Using the email option also saves a copy of the report to disk.

5. Click **Save and Deploy**.

Create a Schedule Profile

1. Click the **Scheduler** tab on the left side of the page.
2. In the **Scheduler Settings** section, click the **+ Scheduler** button.
3. Specify the schedule profile settings as appropriate.

The following table describes the attributes in the **Add a scheduler** and **Edit a scheduler** panes:

Attributes	Description
Name	Enter a name .
Scheduler Type	Select continuous .
Start On	Select the date and time for the first report to be generated.
Run for	Not applicable.

(Continued)

Attributes	Description
End On	(Optional) Select the date and time to stop generating reports. Leave blank to generate the report indefinitely.
Repeat	Select one of the following: <ul style="list-style-type: none"> • The frequency (every day, week, month, or year) at which you want the report to be generated. • Never—generate the report only once. • Custom—select and use the Repeat Every fields to configure a custom frequency.

4. Click **Save and Deploy**.

Create a Report

1. Click the **Report** tab on the left side of the page.
2. In the **Report Settings** section, click **+ Report** button.
3. Specify the schedule profile settings as appropriate.

The following table describes the attributes in the **Add a report setting** and **Edit a report setting** panes:

Attributes	Description
Name	Enter a name.
Format	Options include HTML and JSON .
Schedule(s)	Select the schedule profile you created above.

(Continued)

Attributes	Description
Destination(s)	Select the destination profile you created above.
Canvas(es)	<p>(Optional) Select graph canvases to include in the report. The list of graph panels in the Panel(s) drop-down list changes based on the canvas selected.</p> <p>For information on creating graphs, see Graph Page.</p>
Panel(s)	<p>(Optional) Select the desired graph panels to include in the report.</p> <p>NOTE: JSON reports include the raw time series data only, no graphs.</p>

4. Click **Save and Deploy**.

Associate the Report to a Device Group or Network Group

1. For a device group, select the **Configuration > Device Groups** page from the left-nav bar.
For a network group, select the **Configuration > Network** page from the left-nav bar.
2. Click the name of the device group or network group for which you want to generate reports.
3. Click the **Edit (Pencil)** button.
4. In the **Reports** section, click the caret to expand the menu.
5. Click the name(s) of the reports that you want to associate with this group.
6. Click **Save and Deploy**.

View Reports

To view reports for a device group or network group:

1. If the report's notification parameters are set to use email, check the email box of the specified account for the report and open the attachment.

2. If the report's notification parameters are set to save to the server's disk (or even if set to email), select the **Monitor > Reports** page from the left-nav bar. The reports are organized by the date and time at which they were generated. The most recent report is listed at the top of the table.
3. Find the report you wish to download. To help find the desired report:
 - Click the column headings to sort based on that column.
 - Search within a column using the text box under the column heading.
 - Use the bottom of the page to view more rows or change pages.
4. Click on the name of the report to download it to your system.

The following is a sample report:

Healthbot Report for Device group "vSRX"

Report Name: "TEST-sched-report-TEST-report-vSRX"

Generation Time: "2019-11-19T19:10"

- [Report 1: Dashboard](#)
- [Report 2: Device State](#)
- [Report 3: Alert State](#)
- [Report 4: Device Inventory](#)

Dashboard

Device State view

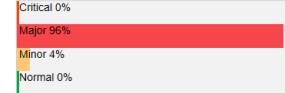
Total : 1 Devices
Risk : 0 Devices
Check : 1 Devices
No Data : 0 Devices
Good : 0 Devices



[Top of page](#)

Alert State View

Total : 27
Critical : 0
Major : 26
Minor : 1
Normal : 0



Alerts of top 5 devices

vSRX : 27

vSRX 100%

Alerts of top 5 topics

system.cpu/check-system-cpu-load-average : 4
interface.statistics/check-interface-status : 2
system.memory/check-system-memory : 2
interface.statistics/check-neighbor-state : 2
interface.statistics/check-out-traffic : 2

system.cpu/check-system-cpu-load-average 33%
interface.statistics/check-interface-status 17%
system.memory/check-system-memory 17%
interface.statistics/check-neighbor-state 17%
interface.statistics/check-out-traffic 17%

Device State

Total Devices Displayed: 1

Page 1 of 1

State	Device Name
system	vSRX

[Top of page](#)

Alert State

Total Alerts Displayed: 20

Page 1 of 2

Severity	Status	Last Receive Time	Duplicate Count	Environment	Topic	Resource	Event	Text
major	open	2019-11-19T12:55:28.862Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-ifl-out-traffic, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:55:28.801Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-out-traffic, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:56:58.367Z	124	Production vSRX	system.cpu	vSRX	Rule=system.cpu/check-system-cpu-load-average, Sensor=CPUUtilizationTable, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:56:58.784Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-interface-flaps, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:55:28.767Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-interface-status, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:55:28.751Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-neighbor-state, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:55:28.681Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-in-traffic, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:56:58.796Z	124	Production vSRX	system.cpu	vSRX	Rule=system.cpu/check-system-cpu, Sensor=components/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:55:28.726Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-ifl-in-traffic, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:56:58.755Z	124	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-in-errors, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:55:28.710Z	249	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-out-errors, Sensor=interfaces/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:56:58.683Z	124	Production vSRX	system.memory	vSRX	Rule=system.memory/check-system-memory, Sensor=components/, event=no-data-received	No data is being received on Sensor
major	open	2019-11-19T12:56:58.055Z	93	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-in-errors, Trigger=in-errors, event=no-data-received	No data present for last 180 seconds
major	open	2019-11-19T12:56:57.060Z	93	Production vSRX	system.cpu	vSRX	Rule=system.cpu/check-system-cpu, Trigger=re-cpu-utilization, event=no-data-received	No data present for last 300 seconds
major	open	2019-11-19T12:56:58.070Z	93	Production vSRX	system.memory	vSRX	Rule=system.memory/check-system-memory, Trigger=re-memory-buffer-utilization, event=no-data-received	No data present for last 300 seconds
major	open	2019-11-19T12:56:58.067Z	93	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-interface-flaps, Trigger=link-flaps, event=no-data-received	No data present for last 180 seconds
major	open	2019-11-19T12:56:57.060Z	93	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-out-errors, Trigger=out-errors, event=no-data-received	No data present for last 180 seconds
major	open	2019-11-19T12:56:58.067Z	93	Production vSRX	system.cpu	vSRX	Rule=system.cpu/check-system-cpu-load-average, Trigger=cpu-utilization-5min, event=no-data-received	No data present for last 120 seconds
major	open	2019-11-19T12:56:58.058Z	93	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-ifl-out-traffic, Trigger=out-traffic, event=no-data-received	No data present for last 300 seconds
major	open	2019-11-19T12:56:58.063Z	93	Production vSRX	interface.statistics	vSRX	Rule=interface.statistics/check-interface-status, Trigger=link-state, event=no-data-received	No data present for last 120 seconds

[Top of page](#)

Device Inventory

Device Id	Hostname	Serial Number	Product	Release	Platform
vSRX	SRX-PNF	1D530FD8888B		19.2R1.8	vSRX

[Top of page](#)

Plots

Create a Field Snapshot

You can capture fields (and their values) from rules applied to deployed devices. In the Paragon Insights CLI, you identify the fields to capture by specifying an *xpath* as shown below (without spaces):

```
{
  capture-fields: [
    /device-group[device-group-name='DevGrp1']/device[device-id=/mx240-1]/topic[topic-
name='system.cpu']/rule[rule-name='check-system-cpu']/re-cpu-utilization,
    /device-group[device-group-name='DevGrp2']/device[device-id=/mx240-1]/topic[topic-
name='interface.statistics']/rule[rule-name='check-in-errors']/in-errors-count
  ]
}
```

In the Paragon Insights GUI, you specify the fields during the creation of a report. Paragon Insights takes care of creating the *xpath* from your configuration in the **Settings > System > Reports > Add a Report Setting** or the **Settings > System > Reports > Edit Report** window as shown in [Figure 60 on page 277](#) below:

Figure 60: Add/Edit Report Setting

Edit a Report Setting

Name*

Test-Report-1

Report Format*

HTML

Schedule(s)*

Hourly

Schedule(s) to run report

Destination(s)*

EmailAdmins

Canvas(es)

Panel(s)

Add Fields

DevGrp1

mx240-1

system.cpu/check-system-cpu

re-cpu-utilization

Field(s)

interface.statistics/check-in-errors

in-errors-count

Field(s)

Topic/Rule

Device

Device Group

CANCEL

SAVE

SAVE & DEPLOY

Compare (Diff) Reports

You can compare the differences between two reports for a device group or network group. The diff allows you to view added/removed/modified alerts, devices, health information, and graphs.

1. On the Reports page, select two reports and click the **Diff Reports** button.
2. The diff opens as an HTML page in a new tab.

Sample Report Diff

A sample of a report diff for a device group is shown below.

DIFF

Header Diff

Field	Old	New
Name	S1-v1-dut_cdk	S1-v2-dut_cdk
Report	r1	r2
Time	2019-11-20T12:02	2019-11-20T11:28

Alert Data Diff

Added	Removed	Modified
0	5	6

Field	Old	New
SeverityCounts	Critical: 0 Major: 18 Minor: 1 Normal: 4	Critical: 0 Major: 18 Minor: 0 Normal: 0
StatusCounts	Ack: 0 Closed: 4 Open: 19 Shelved: 0 Expired: 0	Ack: 0 Closed: 0 Open: 18 Shelved: 0 Expired: 0
TotalAlerts	23	18
TotalMinorOpen	1	0
TotalNormalClosed	4	0

Added Alerts

None

Removed Alerts

Severity	Status	CreateTime	DuplicateCount	Environment	Service	Resource	Event	Text
minor	open	2019-11-20T11:53:20.686Z	0	Production:dut_cdk	system.cpu	r0	► Event	Routing Engine0 15min CPU utilization(71) is in medium range
normal	closed	2019-11-20T11:51:20.682Z	0	Production:dut_cdk	system.cpu	r0	► Event	Routing Engine0 5min CPU utilization(53) is normal
normal	closed	2019-11-20T11:49:20.692Z	0	Production:dut_cdk	system.cpu	r0	► Event	Routing Engine0 CPU utilization(1) is normal
normal	closed	2019-11-20T11:49:21.690Z	0	Production:dut_cdk	system.processes	r0	► Event	ppt_11_80000010 cpu utilization(0) is normal
normal	closed	2019-11-20T11:49:20.688Z	0	Production:dut_cdk	system.cpu	r0	► Event	Routing Engine0 1min CPU utilization(31) is normal

Modified Alerts

a0c2aa28-ba1d-40cf-b2d8-ca30c53713a1

Field	Old	New
duplicateCount	101	91

6f78930d-6131-46f8-b927-d69a80ee3537

Field	Old	New
duplicateCount	101	91

88f632d5-3219-4b13-846e-00c196d701aa

Field	Old	New
duplicateCount	1	0

5292b9d9-3733-48d5-a285-43b0236f6e4

Field	Old	New
duplicateCount	1	0

3146d7c5-7a4a-4816-9b02-d77a98452200

Field	Old	New
duplicateCount	1	0

2c302216-7212-41ca-835e-ed3389364b56

Field	Old	New
duplicateCount	2	0

Device Group Health Diff

Added	Removed	Modified
1	0	0

Field	Old	New
DeviceGroupHealth_Red	0	1
DeviceGroupHealth_TotalDevices	2	3

Device Group Added Devices

Name	State
r0	red

Device Group Removed Devices

None

Device Group Modified Devices

None

Device Facts Diff

Added Device Facts

None

Removed Device Facts

None

Modified Device Facts

.....



A sample of a report diff for a network group is shown below.

DIFF

Header Diff

Field	Old	New
Name	S1-r1-N1	S1-r1-dut_chk
Group	N1	dut_chk
Type	network-group	device-group

Alert Data Diff



Added	Removed	Modified
23	1	0

Statistics Changes

Field	Old	New
SeverityCounts	Critical: 0 Major: 1 Minor: 0 Normal: 0	Critical: 0 Major: 18 Minor: 1 Normal: 4
StatusCounts	Ack: 0 Closed: 0 Open: 1 Shelved: 0 Expired: 0	Ack: 0 Closed: 4 Open: 18 Shelved: 0 Expired: 0
TotalAlerts	1	23
TotalMajorOpen	1	18
TotalMinorOpen	0	1
TotalNormalClosed	0	4

Added Alerts

Severity	Status	CreateTime	DuplicateCount	Environment	Service	Resource	Event	Text
major	open	2019-11-20T10:53:20.974Z	2	Production:dut_chk	system.cpu	r0	► Event	No data is being received on Sensor
major	open	2019-11-20T06:51:54.555Z	101	Production:dut_chk	system.memory	r1	► Event	No data is being received on Sensor
major	open	2019-11-20T06:51:54.529Z	101	Production:dut_chk	system.cpu	r1	► Event	No data is being received on Sensor
minor	open	2019-11-20T11:53:20.686Z	0	Production:dut_chk	system.cpu	r0	► Event	Routing Engine0 15min CPU utilization(71) is in medium range
normal	closed	2019-11-20T11:51:20.682Z	0	Production:dut_chk	system.cpu	r0	► Event	Routing Engine0 5min CPU utilization(53) is normal
normal	closed	2019-11-20T11:49:21.690Z	0	Production:dut_chk	system.processes	r0	► Event	ppl_11_80000010 cpu utilization(0) is normal
normal	closed	2019-11-20T11:49:20.692Z	0	Production:dut_chk	system.cpu	r0	► Event	Routing Engine0 CPU utilization(1) is normal
normal	closed	2019-11-20T11:49:20.686Z	0	Production:dut_chk	system.cpu	r0	► Event	Routing Engine0 1min CPU utilization(31) is normal
major	open	2019-11-20T10:53:40.815Z	1	Production:dut_chk	system.processes	r0	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:40.781Z	1	Production:dut_chk	system.storage	r0	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:35.781Z	1	Production:dut_chk	system.processes	r0	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:25.779Z	0	Production:dut_chk	system.storage	r2	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:21.006Z	0	Production:dut_chk	system.cpu	r2	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:20.943Z	0	Production:dut_chk	system.processes	r2	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:20.909Z	0	Production:dut_chk	system.cpu	r1	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:20.875Z	0	Production:dut_chk	system.storage	r1	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:20.846Z	0	Production:dut_chk	system.processes	r1	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:20.812Z	0	Production:dut_chk	system.processes	r2	► Event	No data is being received on Sensor
major	open	2019-11-20T10:53:20.780Z	0	Production:dut_chk	system.processes	r1	► Event	No data is being received on Sensor
major	open	2019-11-20T06:35:55.872Z	83	Production:dut_chk	system.cpu	r0	► Event	No data is being received on Sensor
major	open	2019-11-20T06:35:55.912Z	83	Production:dut_chk	system.memory	r0	► Event	No data is being received on Sensor
major	open	2019-11-20T06:33:57.147Z	1	Production:dut_chk	system.cpu	r0	► Event	Routing Engine1 5min CPU utilization(\$cpu-5min) exceed high threshold(80)
major	open	2019-11-20T06:33:57.143Z	1	Production:dut_chk	system.cpu	r0	► Event	Routing Engine1 15min CPU utilization(\$cpu-15min) exceed high threshold (75)

Removed Alerts

Severity	Status	CreateTime	DuplicateCount	Environment	Service	Resource	Event	Text
major	open	2019-11-20T06:33:10.688Z	1	Production:N1	ospf	-	► Event	Peer MTUs are not equal r0-mtu: \$r0-mtu - r1-mtu: \$r1-mtu

Modified Alerts

None

Network Group Health Diff

Field	Old	New
NetworkHealth.Name	N1	
NetworkHealth.State	red	

Device Facts Diff

Added Device Facts

None

Removed Device Facts

None

Modified Device Facts

None

Graph Diff

None

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
3.1.0	Capture a snapshot of a specific set of fields at a given point in time (HealthBot Release 3.1.0 and later)
2.1.0	Generate reports on demand (HealthBot Release 2.1.0 and later)
2.1.0	Compare (diff) two reports (HealthBot Release 2.1.0 and later)

Use Exim4 for E-Mails

Exim4 is a mail transfer agent (MTA) for Unix-like systems that connect to the internet. The Exim4 agent, that is included in the Paragon Insights software, sends network health reports and alert notifications (for network or device issues) to the e-mail account of the Exim4 host user. The Exim4 host is the Paragon Insights primary node.

NOTE: In case of multinode Paragon Insights installation with more than one primary node, the Exim4 host is one of the primary nodes.

To enable Paragon Insights to use Exim4 MTA, you must do the following:

- Configure the Exim4 hostname (the primary node hostname) in the file that has the environment variables of all microservices. This configuration ensures that Paragon Insights can reach the Exim4 host when an alert or a report is generated. If the Exim4 host is not reachable, Paragon Insights does not forward the e-mail to the Exim4 agent.
- Configure your DNS server to resolve the Exim4 host's FQDN to the Paragon Insights virtual IP (VIP) address. The format of the FQDN is hostname.domain.top-level-domain. This configuration ensures that the Exim4 agent discovers the DNS mail server for the domain based on the Exim4 host's FQDN. The Exim4 agent then forwards the e-mail to this DNS mail server.

RELATED DOCUMENTATION

[Configure the Exim4 Agent to Send E-mail](#) | 283

Configure the Exim4 Agent to Send E-mail

Configure your DNS server to resolve the FQDN of the Exim4 host to the Paragon Insight's virtual IP address (VIP).

To configure the Exim4 Agent to send email, you must first configure the Paragon Insights primary node as the Exim4 host. To do so, you must enter the node's FQDN in the `healthbot.sys` file. The `healthbot.sys` file contains a list of all environment variables for the Paragon Insights microservices. After you modify the `healthbot.sys` file with the Exim4 hostname, you must run the `healthbot restart` command to restart the alerta microservice with the changes you made.

NOTE: Before you begin, configure your DNS server to resolve the FQDN of the Exim4 host to the Ingress Controller's VIP.

Use the following steps to enable the Exim4 agent to send e-mails.

1. Log into the Paragon Insights primary node using your server credentials.
2. Type the following command to become a root user.

```
root@primary-node# su -root
```
3. Change to the `/var/local/healthbot` path.

```
root@primary-node# cd /var/local/healthbot
```
4. Type the following command to open the `healthbot.sys` file.

```
root@primary-node:/var/local/healthbot# vi healthbot.sys
```
5. Scroll down to the `api-server` section in the `healthbot.sys` file.
6. Type the Paragon Insights primary node FQDN as the value for the `HOST_HOSTNAME` variable.
 For example, `HOST_HOSTNAME = example.domain.top-level-domain`.
7. Type `:wq!` to save the changes and exit the file.
8. Type the following command to update and restart the alerta microservice.

```
root@primary-node# ./healthbot restart --device-group healthbot -s alerta
```

Now, you have enabled the Exim4 agent to send e-mails to the e-mail account associated with the primary node.

To send e-mail alert notifications, you must configure your e-mail address in a notification profile and enable that notification profile on device groups. See ["Alerts and Notifications" on page 252](#) for more information.

To e-mail reports, you must configure your e-mail address in the report settings. See ["Generate Reports" on page 268](#) for more information.

RELATED DOCUMENTATION

| [Use Exim4 for E-Mails](#) | [282](#)

Manage Audit Logs

IN THIS SECTION

- [Filter Audit Logs](#) | [284](#)
- [Export Audit Logs](#) | [285](#)
- [Paragon Insights Commands and Audit Logs](#) | [286](#)

An audit log is a record of a sequence of activities that have affected a specific operation or procedure. Audit logs are useful for tracing events and for maintaining historical data.

Audit logs contain information about tasks initiated by using the Paragon Insights GUI or APIs. In addition to providing information about the resources that were accessed, audit log entries usually include details about user-initiated tasks, such as the name of the user who initiated a task, the status of the task, and date and time of execution.

Audit logs from microservices, such as `healthbot` command and `config` server are collected in the audit log client library, and are stored in the Postgres SQL database. The audit log client library is installed with microservices during Paragon Insights installation.

NOTE: Device-driven tasks (tasks not initiated by the user) are not recorded in audit logs.

Filter Audit Logs

You can filter audit logs from the **Administration > Audit Logs** page of the Paragon Insights GUI. You can apply filters to audit logs before you export audit logs in a CSV file.

1. Select **Administration > Audit Logs**.

The **Audit Logs** page appears displaying the audit logs.

2. Click **Filter > Add Filter**.

The **Add Criteria** pop-up appears.

3. Enter the following information in the **Add Criteria** pop-up.

- a. Select the column you want to apply the filter to from the **Field** drop-down list.
- b. Select the condition you want to apply the column from the **Condition** drop-down list.
- c. Enter the start or finish time in the **Value** text box, that you want to apply to the filter.

4. Click **Add** to apply the filter.

(Optional) You can export audit logs as a CSV file or PDF file after you apply a filter. For more information, see ["Export Audit Logs" on page 285](#).

Export Audit Logs

Starting in Release 4.1.0, you can export audit logs in a portable document format (PDF) file and comma-separated values (CSV) file.

To export audit logs:

1. Select **Administration > Audit Logs**.

The **Audit Logs** page appears.

2. Click **Export Logs > CSV** to download audit logs in a CSV format.

Click **Export Logs > PDF** to download audit logs in PDF format.

In the warning message that appears, do any one of the following:

- Click **Continue** to export all audit logs with all fields as CSV or PDF files.
- Click **Filter** to filter audit log fields before you export as a CSV file. You cannot filter audit logs before you export as PDF file.

To filter audit logs, see ["Filter Audit Logs" on page 284](#).

After you have applied a filter, click **Export Logs>CSV** again to start the export.

NOTE: You can export audit logs for a maximum of 30 days prior to the current date and time. For example, if the current date is May 31, 2018, you can export the audit logs starting from May 1, 2018.

Depending on the settings of the browser that you are using, the CSV/PDF file containing the audit logs for the specified time period is either downloaded directly, or you are asked to open or save the file.

Paragon Insights Commands and Audit Logs

Starting with Paragon Insights Release 4.1.0, audit logs are generated when you run the following commands from the CLI:

- add-node
- remove-node
- modify-uda-engine
- modify-udf-engine
- modify-workflow-engine
- remove-plugin
- load-plugin

The audit log generated will include information on who initiated the job, node name, and status of the job. You must enter credentials (username and password) to run the commands. If you have already set a username (HB_USERNAME) and password (HB_PASSWORD), you are not prompted to enter credentials when you run a command.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.1.0	Starting in Release 4.1.0, you can export audit logs in a portable document format (PDF) file and comma-separated values (CSV) file.

Configure a Secure Data Connection for Paragon Insights Devices

IN THIS SECTION

- [Configure Security Profiles for SSL and SSH Authentication](#) | 288

● [Configure Security Authentication for a Specific Device or Device Group](#) | 288

Paragon Insights (formerly HealthBot) supports the following authentication methods to provide a secure data connection for Paragon Insights devices:

Authentication Method	Sensor Type	Description	Required Paragon Insights Security Parameters
Mutual SSL	OpenConfig	Client authenticates itself with the server and the server authenticates itself with the client.	<ul style="list-style-type: none"> Local certificates (includes the client certificate and client key) CA certificate Server common name
Server-side SSL	OpenConfig	Server authenticates itself with the client.	<ul style="list-style-type: none"> CA certificate Server common name
Public key SSH	iAgent	Authenticates users with password-protected SSH key files.	<ul style="list-style-type: none"> SSH key file Passphrase Username
Password	All	Authenticates users with a password.	<ul style="list-style-type: none"> Username Password

You can associate SSL or SSH certificates and keys with Paragon Insights devices through user-defined security profiles:

- ["Configure Security Profiles for SSL and SSH Authentication" on page 288](#)
- ["Configure Security Authentication for a Specific Device or Device Group" on page 288](#)

Configure Security Profiles for SSL and SSH Authentication

To configure security profiles for SSL and SSH authentication:

1. Click the **Settings > Security** option in the left-nav bar.
2. Click the add profile button for one of the following profiles and enter the required information:

Security Profile	Description of Parameters	
CA	Name Enter profile name. Upload Certificate Choose the CA certificate file and then click Open . The supported file extension is CRT.	
Local Certificates	Name Enter profile name. Upload Certificate Choose the client certificate file and then click Open . The supported file extension is CRT. Upload Key Choose the client key file and then click Open . The supported file extension is KEY.	
SSH Keys	Name Enter profile name. Upload Key File Choose the private key file generated by ssh-keygen and then click Open . Passphrase Enter the authentication passphrase.	

3. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.
4. Repeat Steps 4 and 5, as needed.
5. Apply the security profiles to a specific device or device group. For more details, see ["Configure Security Authentication for a Specific Device or Device Group" on page 288](#).

Configure Security Authentication for a Specific Device or Device Group

1. Click the **Dashboard** option in the left-nav bar.
2. Click the name of the device or device group for which you want to configure security authentication. The device or device group profile pane appears, respectively.
3. Under **Authentication**, enter the required parameters for each applicable authentication method: Password, SSL, or SSH. All methods can be configured together on a single device or device group profile.

Authentication Method	Description of Parameters	
Password	Username	Enter the authentication username.
	Password	Enter the authentication password.
SSL	Server Common Name	Enter the server name protected by the SSL certificate.
	CA Profile*	Choose the applicable CA profile(s) from the drop-down list.
	Local Certificate*	Choose the applicable local certificate profile(s) from the drop-down list.
SSH	SSH Key Profile*	Choose the applicable SSH key profile(s) from the drop-down list.
	Username	Enter the authentication username.

*To edit or view details about saved security profiles, go to the Settings > Security page in the left-nav bar.

The following guidelines apply to the **Authentication** configuration:

- Paragon Insights decides which authentication method to apply to a device or device group based on which of the required security parameters are configured.
- When more than one method is valid, Paragon Insights prioritizes SSL and SSH authentication over password-based authentication.
- Paragon Insights prioritizes device-level settings over device group-level settings.

4. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.

Configure Data Summarization

IN THIS SECTION

- [Creating a Raw Data Summarization Profile | 292](#)
- [Creating a Data Rollup Summarization Profile | 294](#)
- [Applying Data Summarization Profiles to a Device Group | 297](#)

Data summarization refers to the process of creating a concise version of raw data and field data. Data can be summarized as a function of time or when a change occurs. You can improve the performance and disk space utilization of the Paragon Insights (formerly HealthBot) time series database (TSDB) by configuring data summarization methods to summarize the raw data and field data collected by Paragon Insights.

Paragon Insights collects data by using push or pull data collection methods. You can create Paragon Insights rules or use the available predefined rules to determine how and when data is collected. This collected telemetry data provides information about the state of network devices and its components. For more information on data collection methods, see [Paragon Insights Data Ingest Guide](#).

You can create a raw data summarization profile to improve the performance and disk space utilization of the TSDB. Starting with Paragon Insights Release 4.0.0, you can create a rollup summarization profile to summarize processed data that is stored in fields in the TSDB. Field data is processed data that is stored in fields in the TSDB. A field is a single piece of information that forms a record in a database. In TSDB, multiple fields of processed data make a record. In releases earlier than Paragon Insights Release 4.0.0, only raw data can be summarized.

[Table 20 on page 290](#) provides a list of the supported data summarization algorithms and a description of their output:

Table 20: Descriptions of the Data Summarization Algorithms

Algorithm	Description of output
Latest	Value of the last data point collected within the time span.
Count	Total number of data points collected within the time span.

Table 20: Descriptions of the Data Summarization Algorithms *(Continued)*

Algorithm	Description of output
Mean	Average value of the data points collected within the time span.
Min	Minimum value of the data points collected within the time span.
Max	Maximum value of the data points within the time span.
On-change	Value of the data point whenever the value is different from the previous data point (occurs independently from the user-defined time span).
Stddev	Standard deviation of the data points collected within the time span.
Sum	Sum of the data points collected within the time span.

If no summarization algorithm is associated with the data, the following algorithms are used by default:

Data type	Data summarization algorithm
Float, integer, unsigned	Mean
Boolean, string	On-change

You can use data summarization profiles to apply specific summarization algorithms to raw data and field data collected by Paragon Insights for a specific device group:

These topics provide instructions on how to create a data summarization profile.

- ["Creating a Raw Data Summarization Profile" on page 292](#)
- ["Creating a Data Rollup Summarization Profile" on page 294](#)

After you have created a data summarization profile, you can apply the profile to a device group. For more information, see ["Applying Data Summarization Profiles to a Device Group" on page 297](#).

Creating a Raw Data Summarization Profile

To create a raw data summarization profile that can be applied to a device group:

1. Click **Settings** > **Summarization Profiles** link in the left-nav bar.

2. Select **Raw Data** from the Summarization Profiles list.

The **Raw Data Summarization Profiles** page is displayed.

3. Click (+) icon to add a summarization profile.

The **Add Raw Data Summarization Profile** page is displayed.

4. In the **Name** text box, enter the name of the profile.

5. Click **Add Type Aggregate** to add an aggregate type.

The **Name** and **Function** drop-down lists are displayed.

Follow these steps to select a name data type and associate it with a data summarization algorithm.

The algorithm configured for a specific sensor path name overrides the algorithm configured for the corresponding data type.

- a. Select a name data type from the **Name** drop-down list.

The available name data types to choose from are string, integer, boolean, float, and unsigned integer.

Starting in Paragon Insights Release 4.2.0, you can also select unsigned integer as a name data type. An unsigned integer is a data type that can contain values from 0 through 4,294,967,295.

- b. After you have selected a name data type, you associate it with a data summarization function.

To associate a name data type with a data summarization function, select a function from the **Functions** drop-down list.

The available functions to choose from are latest, count, mean, min, max, on-charge, stddev, and sum.

- c. (Optional) To add another aggregate type, click **Add Type Aggregate**, and repeat step "5.a" on page 292 and step "5.b" on page 292.

6. Click **Add Path Aggregate** to add an aggregate path.

The **Name** and **Function** drop-down lists are displayed.

To assign a sensor path name and associate it with a data summarization algorithm:

NOTE:

The algorithm configured for a specific sensor path name overrides the algorithm configured for the corresponding data type.

- a. Enter a sensor path name in the **Name** text box.

You can enter a path name for a sensor that is not supported by Paragon Insights. For sensors supported by Paragon Insights, the path name must be entered in the following format:

Sensor	Path Name Format	Example
Open Config	<i>sensor-path</i>	/components/component/name
Native GPB	<i>sensor-name:sensor-path</i>	jnpr_qmon_ext:queue_monitor_element_info.percentage
iAgent	<i>yaml-table-name:sensor-path</i>	REutilizationTable:15_min_cpu_idle
SNMP	<i>snmp-table-name:sensor-path</i>	.1.3.6.1.2.1.2.2:jnxLED1Index ospfNbrTable:ospfNbrIpAddr
Syslog	<i>pattern-set: sensor-path</i>	interface_link_down:operational-status
Flow (NetFlow)	<i>template-name:sensor-path</i>	hb-ipfix-ipv4-template:sourceIPv4Address

- b. After you have entered a sensor path name, you associate it with a data summarization function.

To associate a sensor path name with a data summarization function, select a function from the Functions drop-down list.

The available functions to choose from are latest, count, mean, min, max, on-charge, stddev, and sum.

- c. (Optional) To add another aggregate path, click **Add Path Aggregate**, and repeat step "6.a" on page 293 and step "6.b" on page 293.

7. Click **Save** to only save the configuration.

Click **Save & Deploy** to save and immediately deploy the configuration.

8. You can now apply the raw data summarization profile that you created to a specific device group. For more information, see ["Applying Data Summarization Profiles to a Device Group" on page 297](#).

Creating a Data Rollup Summarization Profile

Paragon Insights Release 4.0.0 supports data rollup summarization. You can create a rollup summarization profile to summarize processed data that is stored in fields in the TSDB. Field data is processed data that provides information on network devices and its components, and is stored in fields in the TSDB. A field is a single piece of information that forms a record in a database. In TSDB, multiple fields of processed data make a record. Data rollup summarization enables efficient data storage and also ensures retaining of data for a longer duration.

You can create a data rollup summarization profile to apply to a device group from the:

- Paragon Insights graphical user interface (GUI)
- Command line interface (CLI)

Create a Data Rollup Summarization Profile by using Paragon Insights UI

To create a data rollup summarization profile:

1. Click **Settings > Summarization Profiles** link in the left-nav bar.
2. Select **Data Rollup** from the **Summarization Profiles** list.

The Data Rollup Summarization Profiles page is displayed.

3. Click (+) icon to add a summarization profile.

The **Add Data Rollup Summarization Profile** page is displayed.

4. Enter the name of the profile in the **Name** text box.

The maximum length is 64 characters.

Regex pattern: "[a-zA-Z][a-zA-Z0-9_-]*"

5. Click **Add Rule** to add an existing Paragon Insights rule for which rollup summarization must be applied.

The **Name** and **Apply on Existing Data** drop-down lists are displayed.

Follow these steps to select a rule, and to apply the rule to the profile. You can also apply the rule to existing data.

- a. Select an existing Paragon Insights from the **Name** drop-down list.
- b. To apply the rule that you selected to existing data, select **True** from the **Apply on Existing Data** drop-down list.

The default value is **False**.

- c. Click **Add Field** to define fields of the rule configured for which data rollup summarization must be applied.

To associate a field to an aggregate function:

- i. Select a field from the **Name** drop-down list for which data must be aggregated.
 - ii. Select one or more aggregate functions from the **Aggregate Function** drop-down list that you want to apply to a field.
- d. (Optional) To add another rule to the profile, click **Add Rule**, and repeat step "5.a" on page 295 through step "5.c" on page 295.

6. Click **Add Data Rollup Order** to define the frequency at which rollup summarization should occur.

The **Name** and **Retention Policy** drop-down lists, and the **Rollup Interval** text box are displayed.

To define a data rollup order:

- a. Enter a name to identify the data rollup order in the **Name** text box.

The maximum length is 64 characters.

Regex pattern: "[a-zA-Z][a-zA-Z0-9_-]*"

- b. Enter a value in the **Rollup Interval** text box to define the interval in which data is summarized.

Regex pattern: "[1-9][0-9]*[mhdw]", where m is minutes, h is hours, d is days, and w is weeks.

Minimum value is 30m. Maximum value is 52w.

- c. Select the retention policy for the rollup order from the **Retention Policy** drop-down list. A retention policy defines how long you want to retain the rolled-up data.

Selecting a retention policy is optional. If you do not select a retention policy, the device group retention policy is considered by default.

- d. (Optional) To define another data rollup order, click **Add Data Rollup Order**, and repeat step "6.a" on page 295 through step "6.c" on page 295.

7. Click **Save** to only save the configuration.

Click **Save and Deploy** to save and immediately deploy the configuration.

8. You can now apply the data rollup summarization profile that you created to a specific device group.
For more information, see ["Applying Data Summarization Profiles to a Device Group" on page 297](#).

Create a Data Rollup Summarization Profile by using CLI

[Figure 61 on page 296](#) is an example configuration of how you can configure a data rollup summarization profile from the CLI.

Figure 61: Example CLI Configuration of Creating a Data Rollup Summarization Profile

```
field-profile sample_profile1 {
  rule rule1 {
    apply-on-existing-data;
    field rule1_field1 {
      aggregate-function max;
    }
    field rule1_field2 {
      aggregate-function [ mean min ];
    }
    field rule1_field3 {
      aggregate-function [ count mean std-dev ];
    }
  }
  rule rule2 {
    field rule2_field1 {
      aggregate-function first;
    }
  }
  rule rule3 {
    apply-on-existing-data;
    field rule3_field1 {
      aggregate-function [ first std-dev ];
    }
  }
  data-rollup-order hourly_rollup {
    interval 1h;
    retention-policy 2weeks;
  }
  data-rollup-order daily_rollup {
    interval 24h;
    retention-policy 2months;
  }
  data-rollup-order weekly_rollup {
    interval 7d;
    retention-policy 6months;
  }
  data-rollup-order monthly_rollup {
    interval 4w;
    retention-policy 1year;
  }
  data-rollup-order yearly_rollup {
    interval 52w;
    retention-policy 5years;
  }
}
```


Applying Data Summarization Profiles to a Device Group

After you create a data summarization profile, you can apply the profile to a specific device group to start summarizing TSDB data:

1. Click **Configuration > Device Group** in the left-nav bar.

The **Device Group Configuration** page is displayed.

2. Select the check box next to the name of the device group to which you want to apply the data summarization profile.

3. Click the **Edit Device Group** icon to edit the device group.

The **Edit <device-group-name>** page is displayed.

4. Apply a raw data summarization profile.

To apply a raw data summarization profile to a device group:

- a. Click **Summarization**.

The **Time Span** and **Data Summarization** text boxes are displayed.

- b. Enter the **Time Span** in seconds (s), minutes (m), hours (h), days (d), weeks (w), or years (y).

- c. Choose the data summarization profiles from the drop-down list to apply the ingest data. To edit or view details about saved data summarization profiles, go to the **Data Summarization** page and click the **Settings** menu option in the left-nav bar.

If you select two or more profiles, the following guidelines apply:

- If the same data type or sensor path name is configured in two or more profiles, the associated algorithms will be combined.
- The table that stores the summarization output includes columns of summarized data for each algorithm associated with each data field collected by Paragon Insights. The naming convention for each column is as follows:

Number of algorithms associated with a data field	Column name for the summarized output
1	<i>field-name</i> Example: 5_sec_cpu_idle

(Continued)

Number of algorithms associated with a data field	Column name for the summarized output
2	<i>field-name_first-algorithm-name, field-name_second-algorithm-name</i> Example: 5_sec_cpu_idle_MIN, 5_sec_cpu_idle_MAX
3	<i>field-name_first-algorithm-name, field-name_second-algorithm-name, field-name_third-algorithm-name..</i>

Apply a data rollup summarization profile

Points to remember before you apply a data rollup summarization profile to a device group:

- Ensure that the rules present in the rollup profile are already associated with the device group.
- You can add one or more than one rollup summarization profile to a device group.
- Rules configured across all the profiles associated with the device group must be unique.
- While associating a rollup profile with a device group, the interval of the first data rollup order must be less than the device group retention policy to avoid data overflow. The device group retention policy is set to 7 days by default.
- When you want to remove a rule that is associated to a device group, you must first remove the data rollup summarization profile.

To apply a data rollup summarization profile to a device group:

a. Click Rollup Summarization.

The **Rollup Summarization Profiles** drop-down list is displayed.

b. Select the rollup summarization profiles you want to associate to this device group from the Rollup Summarization Profiles drop-down list.

c. (Optional) You can also deploy rollup configuration at the device group-level by using the CLI.

See [Figure 62 on page 299](#) for an example CLI configuration.

Figure 62: Example CLI Configuration

```
device-group DG {
  authentication {
    password {
      username root;
      password "$9$0E9lBIhx7VsYohSbs2aiHCtu0IclK8"; ## SECRET-DATA
    }
  }
  field-data {
    rollup {
      profile sample_profile1;
    }
  }
  devices [ d1 d2 ];
}
```

- 5. Click **Save** to only save the configuration.
- Click **Save and Deploy** to save and immediately deploy the configuration.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
4.0.0	Paragon Insights Release 4.0.0 supports data rollup summarization. You can create a rollup summarization profile to summarize processed data that is stored in fields in the TSDB.

Modify the UDA, UDF, and Workflow Engines

IN THIS SECTION

- [Overview | 300](#)
- [How it Works | 302](#)
- [Usage Notes | 303](#)
- [Configuration | 303](#)

Overview

When creating rules, Paragon Insights (formerly HealthBot) includes the ability to run user-defined actions (UDAs) as part of a trigger. UDAs are essentially Python scripts that can be executed by a Paragon Insights rule. For example, you might configure a rule with a trigger that reacts to some critical interface going down and responds to the event by calling a function to send an SMS alert. You can write the logic to send the SMS in a UDA python script.

Starting with Paragon Insights Release 4.1.0, you can schedule UDAs and notifications. This is useful when you deploy multiple parallel instances of Paragon Insights in different locations. You can schedule UDAs to run alternatively from UDA schedulers located in different regions. In the event of a node failure, the UDA scheduler running in a parallel instance continues to execute your UDAs and notifications. For more information, see ["Enable UDA Scheduler in Trigger Action" on page 306](#).

Paragon Insights also includes the ability to run user-defined functions (UDFs). Created as Python scripts, UDFs provide the ability to process incoming telemetry data from a device and store the processed value in the time-series database for that device/device-group. For example, the device may be sending FPC temperature in Celsius but you want to process it to be stored as Fahrenheit values in the database.

Starting with HealthBot Release 3.2.0, the processing of UDF/UDA fields has been moved to microservices called UDF farm. This approach allows for Paragon Insights to process multiple data points from multiple devices and fields at the same time (parallel processing). The result is a 4 to 5 times increase in processing performance for UDA/UDF.

While UDAs and UDFs provide excellent additional capabilities to Paragon Insights, there can be cases where the scripts may be importing Python modules that are not included in the default Paragon Insights installation. Given this, you need to be able to add modules as needed to the engine that runs these scripts. Paragon Insights Release 2.1.0 and later solves this challenge by allowing you to modify the UDA, UDF and Workflow engines, using a bash script that contains the instructions to install any dependencies.

Global Variables in Python Scripts

TAND executes Python scripts that use global variables. Global variables retain a value across multiple UDFs.

The following is an example function to calculate cumulative sum and store the value in global variable *sum*.

```
sum = 0
def cumulative_sum (a, b):
    global sum
    sum = sum + a + b
    return sum
```

Using global variables in UDFs can prevent you from availing the gains in processing performance ensured by UDF farms.

Instead of global variables, you can use the Python construct `**kwargs` to capture values that must be retained across different functions. When Paragon Insights calls a function (defined in a UDF), it sends topic name, rule name, device group, point time, and device ID that are captured using the construct `**kwargs`. In case of UDAs, Paragon Insights sends topic name and rule name while executing the Python script.

Along with infrastructure values, Paragon Insights also sends a parameter called *hb_store* in `**kwargs` that fetches the last computed value for a variable.

To illustrate how *hb_store* works in the cumulative addition example:

```
def sum(a, b, **kwargs):
    if 'sum' not in kwargs[hb_store]:
        kwargs[hb_store]['sum'] = 0 #if 'sum' is not present in kwargs, declare the
        initial 'sum' value as 0.

        kwargs[hb_store]['sum'] = kwargs[hb_store]['sum'] + a + b #Store cumulative
        addition value in 'sum'

    return kwargs[hb_store]['sum']
```

Each time a function with the above code is called, it performs addition of last stored value in 'sum' with the value of *a* and value of *b*. The new value of addition operation is displayed and stored in 'sum'.

How it Works

You can modify the UDA, UDF, or Workflow engine using the Paragon Insights CLI, as shown below.

```
user@HB-server:~$ healthbot modify-uda-engine --help
usage: healthbot modify-uda-engine [-h] (-s SCRIPT | --rollback) [--simulate]

optional arguments:
  -h, --help            show this help message and exit
  -s SCRIPT, --script SCRIPT
                        Run script in UDA engine
  --rollback, -r        Rollback UDA engine to original state
  --simulate            Run script in simulated UDA engine and show output

user@HB-server:~$ healthbot modify-udf-engine --help
usage: healthbot modify-udf-engine [-h] (-s SCRIPT | --rollback) [--simulate]
                                [--service SERVICE]

optional arguments:
  -h, --help            show this help message and exit
  -s SCRIPT, --script SCRIPT
                        Run script in UDF engine
  --rollback, -r        Rollback UDF engine to original state
  --simulate            Run script in simulated UDF engine and show output
  --service SERVICE     Modify specific service UDF

root@davinci-master:/var/local/healthbot# healthbot modify-workflow-engine --help
usage: healthbot.py modify-workflow-engine [-h] (-s SCRIPT | --rollback)
                                [--simulate]

optional arguments:
  -h, --help            show this help message and exit
  -s SCRIPT, --script SCRIPT
                        Run script in WORKFLOW engine
  --rollback, -r        Rollback WORKFLOW engine to original state
  --simulate            Run script in simulated WORKFLOW engine and show output
```

The commands have three main options:

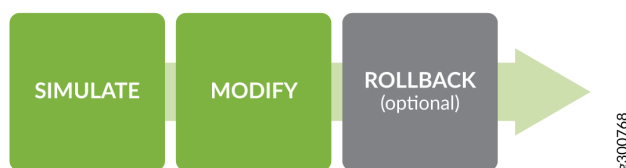
- Simulate—test a script (and view its output) in the simulated UDA/UDF/Workflow engine environment without affecting the running Paragon Insights system
- Modify—modify the actual UDA/UDF/Workflow engine using a script
- Rollback—revert to the original version of the UDA/UDF/Workflow engine

Usage Notes

- The bash script will run in a container running Ubuntu OS Release 16.04 or 18.04; write the script accordingly.
- The script must be non-interactive; any questions must be pre-answered. For example, use the ‘-y’ option when installing a package using apt-get.
- If you prefer to copy the source packages of the dependency modules onto the Paragon Insights server so the engine can manually install them instead of downloading them from the Internet, place the required source packages in the `/var/local/healthbot/input` directory. Then within your bash script, point to the `/input` directory. For example, to use a file placed in `/var/local/healthbot/input/myfile.txt`, set the bash script to access it at `/input/myfile.txt`.
- Modifying the UDA/UDF/Workflow engine more than once is *not* an incremental procedure; use a new bash script that includes both the original and new instructions, and re-run the modify procedure using the new script.
- UDA/UDF/Workflow modifications are persistent across upgrades.

Configuration

As a best practice, we recommend that you use the following workflow:



This best-practice approach ensures that you first validate your script in the simulated environment before modifying the real engine.

NOTE: The examples below use the UDA engine; these procedures apply equally to the UDF and Workflow engines.

NOTE: The procedure below assumes your Paragon Insights server is installed, including running the `sudo healthbot setup` command.

SIMULATE

Use the simulate feature to test your bash script in the simulated environment, without affecting the running Paragon Insights system.

To simulate modifying the UDA engine:

1. Enter the command `healthbot modify-uda-engine -s /<path>/<script-file> --simulate`.
2. The script runs and the output shows on screen, just as if you entered the script commands yourself.

```
user@HB-server:~$ healthbot modify-uda-engine -s /var/tmp/test-script.sh --simulate
Running /var/tmp/test-script.sh in simulated alerta engine..
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
...
Fetched 4296 kB in 15s (278 kB/s)
Reading package lists...

Building dependency tree...

Reading state information...
...
```

MODIFY

When you are satisfied with the simulation results, go ahead with the actual modification procedure.

To modify the UDA engine:

1. Load the desired bash script onto the Paragon Insights server.
2. If your Paragon Insights server is fully up and running, issue the command `healthbot stop -s alerta` to stop the running services.

3. Run the command `healthbot modify-uda-engine -s /<path>/<script-file>`.

```
user@HB-server:~$ healthbot modify-uda-engine -s /var/tmp/test-script.sh
Running /var/tmp/test-script.sh in simulated alerta engine..
Success! See /tmp/.alerta_modification.log for logs
Please restart alerta by issuing 'healthbot start --device-group healthbot -s alerta'
```

4. (Optional) As noted in the output, you can check the log file to further verify the script was loaded successfully.
5. Restart the alerta service using the command `healthbot start -s alerta`.
6. Once complete, verify that the alerta service is up and running using the command `healthbot status`.
7. To verify that the UDA engine has been updated, use the command `healthbot version -s alerta` and check that the `healthbot_alerta` container is using the `<version>-custom` tag.

```
user@HB-server:~$ healthbot version -s alerta
{'alerta': 'healthbot_alerta:2.1.0-custom'}
```

The UDA engine is now running with the installed dependencies as per the bash script.

ROLLBACK

If you have a need or desire to remove the changes to the engine, you can revert the engine to its original state.

To rollback the UDA engine:

1. Enter the command `healthbot modify-uda-engine --rollback`.

```
user@HB-server:~$ healthbot modify-uda-engine --rollback
Rolling back alerta engine to original state..
Successfully rolled back alerta engine
Please restart alerta by issuing 'healthbot start --device-group healthbot -s alerta'
```

Note that it is not necessary to restart the alerta service at this point.

2. Once complete, verify that the alerta service is up and running using the command `healthbot status`.

3. To verify that the UDA engine has reverted back, use the command `healthbot version -s alerta` and check that the `healthbot_alerta` container is using the `<version>` tag.

```
user@HB-server:~$ healthbot version -s alerta
{'alerta': 'healthbot_alerta:2.1.0'}
```

The UDA engine is now running in its original state, with no additional installed dependencies.

Enable UDA Scheduler in Trigger Action

In Paragon Insights Release 4.1.0, you can schedule UDAs and notifications to be executed within a set time interval. To schedule UDAs, you must first create a discrete scheduler and then link the scheduler in the Trigger Action page.

NOTE: You can link only one trigger action scheduler to a Paragon Insights instance.

To know more about creating a scheduler, see ["Generate Reports" on page 268](#).

The scheduler set in Trigger Action applies to all device groups and network groups. You can disable UDA scheduling in the device group or network group configuration. To know more, see ["Manage Devices, Device Groups, and Network Groups" on page 121](#).

To enable a scheduler:

1. Go to **Settings > System**.
2. Click the Trigger Action tab.

The Trigger Action page appears.

3. Select a scheduler profile that you want to associate with Trigger Action.
4. Do one of the following:

- Click **Save** to save the scheduler profile.

The profile is not applied to device or network groups. This option enables you to commit or rollback the configuration changes in the platform.

- Click **Save and Deploy** to deploy the configuration in your Paragon Insights instance.

The UDAs and notifications are generated based on the time period and time interval configured in the scheduler for the application instance.

You cannot rollback configuration changes applied through **Save and Deploy**. However, you can remove the scheduler profile and repeat the save and deploy option to cancel UDA scheduling.

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
3.2.0	Starting with HealthBot Release 3.2.0, the processing of UDF/UDA fields has been moved to microservices called UDF farm.

Commit or Roll Back Configuration Changes in Paragon Insights

In Paragon Insights, when you make changes to the configuration, you can perform the following operations:

- Save
- Save and Deploy
- Delete
- Delete and Deploy

These operations and their effect on the Paragon Insights ingest services are explained in [Table 21 on page 308](#).

For changes that were not already deployed, you can either commit or roll back the changes, which you can do using the Health Configuration Deployment Status page; the procedure is provided below [Table 21 on page 308](#).

Table 21: Operations After Changing the Configuration in Paragon Insights

Operation	Explanation	Effect on Ingest Services
Save	Saves the changes to the database, but doesn't apply the changes to the ingest services.	No effect until the changes are committed.
Delete	Deletes the changes from the database, but doesn't apply the changes to the ingest services.	
Save and Deploy	Saves the changes to the database and applies the changes to the ingest services.	<p>Paragon Insights Release 4.1.0 and earlier—The ingest services starts processing the data based on the changes configured.</p> <p>Paragon Insights Release 4.2.0 and later—When you commit a configuration, the changes are accepted after validation and acknowledged immediately.</p>
Delete and Deploy	Deletes the changes from the database and applies the changes to the ingest services.	The background job runs periodically and applies the changes to ingest services. Only after the background job applies these changes, does the ingest service process data based on the changes configured.

NOTE: Users might do the Save or Delete operations if they have multiple configuration steps and want to stack the configuration steps, so that they can later commit the changes at one go.

To commit or roll back configuration changes in Paragon Insights:

1. On the Paragon Insights banner, click the deployment status icon.

The Health Configuration Deployment Status page appears displaying:

- The status of the last deployment.
- The list of device groups and network groups for which pending changes have not been committed.
- The list of device groups and network groups for which pending deletions have not been committed.

2. You can do one of the following:

- Commit any changes that are pending deployment:

- a. Click **Commit**.

Paragon Insights triggers the commit operation and the configuration changes are applied to the ingest services. Depending on the number of configuration changes and number of device groups and network groups affected, applying the configuration changes to the ingest services might take between a few seconds or a few minutes to complete.

After the operation completes, a confirmation message is displayed. Paragon Insights starts running the associated playbooks and starts ingesting the data.

- b. Go to Step 3.

- Roll back any changes that are pending deployment:

- a. Click **Rollback** to roll back any changes that are pending deployment. (The roll back operation discards the pending configuration changes that were previously saved in the database.)

Paragon Insights triggers the roll back operation and the configuration changes are rolled back almost immediately. After the rollback is complete, a confirmation message is displayed.

- b. Go to Step 3.

NOTE: After the roll back or commit operation is completed, the status of the last deployment in the Health Configuration Deployment Status page displays the last operation that was completed.

3. Click **Close** to exit the page.

You are returned to the previous page from which you accessed the Health Configuration Deployment Status page.

Logs for Paragon Insights Services

IN THIS SECTION

- [Configure Service Log Levels for a Device Group or Network Group | 310](#)
- [Download Logs for Paragon Insights Services | 311](#)

Paragon Insights (formerly HealthBot) runs various services (such as iAgent, jtimon, and Telegraf) to monitor the health of the network and individual devices. Each of these Paragon Insights services runs independently in a containerized environment and produces its own set of log messages that are categorized by severity level. You can configure different levels of logs to collect and download.

[Table 22 on page 310](#) lists the severity levels of the Paragon Insights services logs. The severity levels are listed in order from the highest severity (greatest effect on functionality) to the lowest. If you select a lower severity level, the logs for each of the higher severity levels will also be collected. The log level for all services is set to **error** by default.

Table 22: Paragon Insights Service Log Message Severity Levels

Severity Level	Description
Error	(Highest level) Conditions that require correction.
	Conditions that warrant monitoring.
Info	Non-error conditions of interest.
Debug	(Lowest level) Debug messages.

This topic includes:

- ["Configure Service Log Levels for a Device Group or Network Group" on page 310](#)
- ["Download Logs for Paragon Insights Services" on page 311](#)

Configure Service Log Levels for a Device Group or Network Group

You can collect different severity levels of logs for the running Paragon Insights services of a device group or network group. To configure which log levels to collect:

1. Click the **Configuration > Device Group** option in the left-nav bar.
2. For a device group, click on the device group name from the list of DEVICE GROUPS.
For a network group, click on the network group name from the list of NETWORK GROUPS.
3. For a Device Group, click the **Edit Device Group (Pencil)** icon
For a Network Group, click the **Edit Network Group (Pencil)** icon

4. In the edit window that pops up, click the caret next to the **Logging Configuration** heading to display the configuration fields.
5. From the drop-down list for **Global Log Level**, select the level of the log messages that you want to collect for every running Paragon Insights service for the device or network group. See [Table 22 on page 310](#) for a definition of the log severity levels. The level is set to **error** by default.
6. In the **Log Level for specific services** section, select the log level from the drop-down list for any specific service that you want to configure differently from the **Global log level** setting. The log level that you select for a specific service takes precedence over the **Global log level** setting.
7. Click **Save** to save the configuration or click **Save and Deploy** to save and deploy the configuration.

Download Logs for Paragon Insights Services

You can choose to download the collected logs for:

- Every running Paragon Insights service for a specific device group or network group.
- A specific running Paragon Insights service for a specific device group or network group.
- Common Paragon Insights services that are running by default for the Paragon Insights application.

To download the logs for Paragon Insights services:

1. Select the **Administration > Log Collection** option in the left-nav bar.
2. Select the **Group Type**, **Group Name**, and **Service Name** of the logs that you want to download:

Parameter	Description						
Group Type	Options are: <table> <tr> <td>device</td><td>Services running for a device group</td></tr> <tr> <td>network</td><td>Services running for a network group</td></tr> <tr> <td>common-services</td><td>Services running by default for the Paragon Insights application.</td></tr> </table>	device	Services running for a device group	network	Services running for a network group	common-services	Services running by default for the Paragon Insights application.
device	Services running for a device group						
network	Services running for a network group						
common-services	Services running by default for the Paragon Insights application.						

(Continued)

Parameter	Description
Group Name	<ul style="list-style-type: none"> For device Group Type, select a device group name. For network Group Type, select a network group name. For common-services Group Type, select Paragon Insights.
Service Name	Select the specific Paragon Insights service for which you want to download the logs.

3. Click **Download** to download the logs to a file on your server. The filename is `healthbot_logs.zip` by default.

Troubleshooting

IN THIS SECTION

- [Paragon Insights Self Test | 313](#)
- [Device Reachability Test | 315](#)
- [Ingest Connectivity Test | 317](#)
- [Debug No-Data | 319](#)

Starting with Release 2.1.0, HealthBot supports four verification and troubleshooting features:

- ["Paragon Insights Self Test" on page 313](#): Verifies that Paragon Insights is working properly
- ["Device Reachability Test" on page 315](#): Verifies that network devices are up and reachable via ping & SSH
- ["Ingest Connectivity Test" on page 317](#): Validates which ingest types are supported for a given device

- ["Debug No-Data" on page 319](#): Provides debugging when a device shows "No-data" in Paragon Insights GUI

You can access these features by navigating to **Administration > Debug** from the left-nav panel.

Paragon Insights Self Test

Overview

When setting up basic functionality in Paragon Insights, it can be challenging to diagnose problems. From installation, to device configuration, to adding devices and applying playbooks, when an issue occurs there are many possible areas to investigate.

Starting with HealthBot Release 2.1.0, the self-test tool validates the core functionality of Paragon Insights. To perform the self test, the tool performs a typical set of tasks:

- Adds a simulated device to Paragon Insights
- Creates a device group, and adds the device
- Creates a rule
- Creates and deploys a playbook
- Streams data from the simulated device
- Displays ongoing status in the dashboard

The self-test instance essentially acts as a fully working setup, running entirely within the Paragon Insights system. When testing is complete, the tool provides a report.

Other Uses for the Self Test Tool

In addition to validating the Paragon Insights installation, the self-test feature also provides:

- An easy way to do a quick demo - the self test instance provides a simulated device connected to Paragon Insights, so you can demo Paragon Insights with no need to add a real device or apply playbooks.
- A good way for new users to get started - the self test auto-configures a simulated device connected to Paragon Insights, thereby eliminating the complexity of adding devices, applying playbooks, and so on.

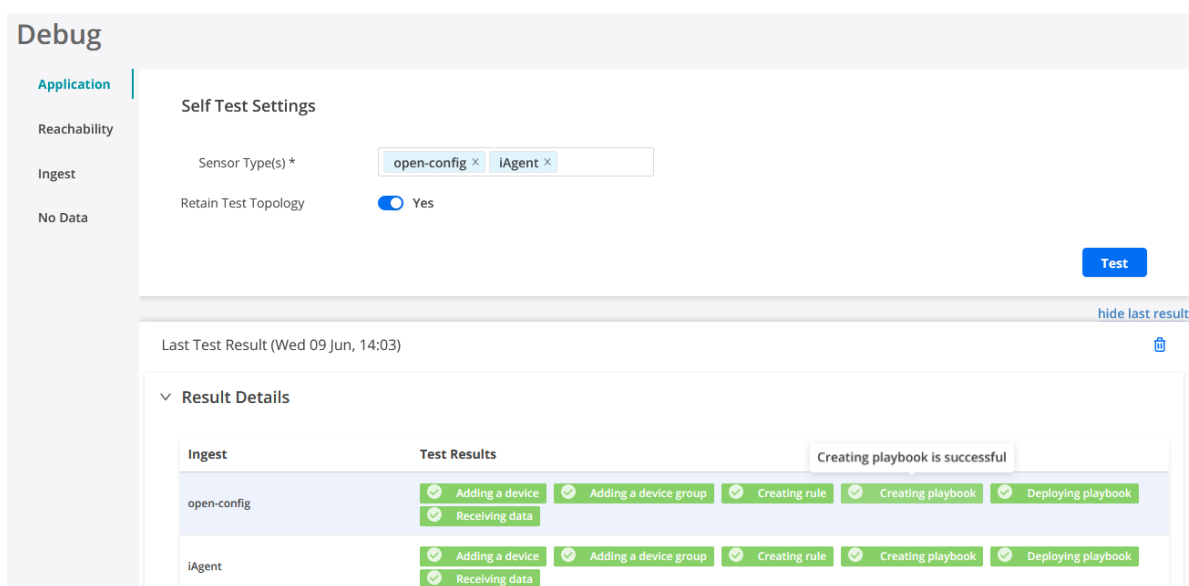
- A 'running reference' - if there is an issue with real devices, you can use a self-test instance to help determine where the issue is; if the self-test instance is OK then the problem is not with the Paragon Insights system.

Usage Notes

- Currently this feature supports simulating devices to stream data for OpenConfig telemetry and iAgent (NETCONF).
- You can retain the self-test instance to act as a 'running reference', as noted above.
- The color coding for the test results is as follows:
 - Green = pass
 - Yellow = error (unable to test)
 - Red = fail
 - Any items with yellow or red status will include a message with more detail about the issue.
- Do not use the self-test tool when there are undeployed changes, as the self-test tool issues its own deploy during execution.
- Do not use rules, playbooks, devices, device-groups or other elements created by the self-test tool with real network devices.

How to Use the Self Test Tool

1. Navigate to the **Administration > Debug** page from the left-nav panel, and select the **APPLICATION** tab.
2. Select the desired sensor type(s) from drop-down menu.
3. Click the **Test** button.
4. After a few moments, the test results appear.



The example above shows that both sensors are working as expected.

Device Reachability Test

Overview

In early versions of Paragon Insights, there was no way to easily determine whether the devices you added were up and reachable; you would need to get through the entire setup procedure - add the device, setup a device group, apply playbooks, monitor devices - at which point the health pages would indicate “no data” indicating that the setup did not work correctly. Furthermore, “no data” does not indicate whether the problem is a reachability issue or data streaming issue.

Starting with HealthBot Release 2.1.0, the device reachability tool can verify connectivity to a device. The tool performs tests using ping and SSH. Paragon Insights uses the device's IP address or host name, based on what was configured when adding the device.

Usage Notes

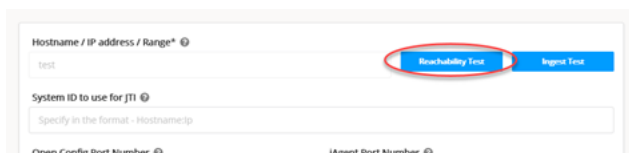
- While this feature is generally intended to help troubleshoot device onboarding, you can use it any time to check device reachability.
- The color coding for the test results is as follows:
 - Green = pass
 - Yellow = error (unable to test)

- Red = fail
- Any items with yellow or red status will include a message with more detail about the issue.

How to Use the Device Reachability Tool

1. To access the tool:

- Navigate to the **Administration > Debug** page from the left-nav panel, and select the **REACHABILITY** tab.



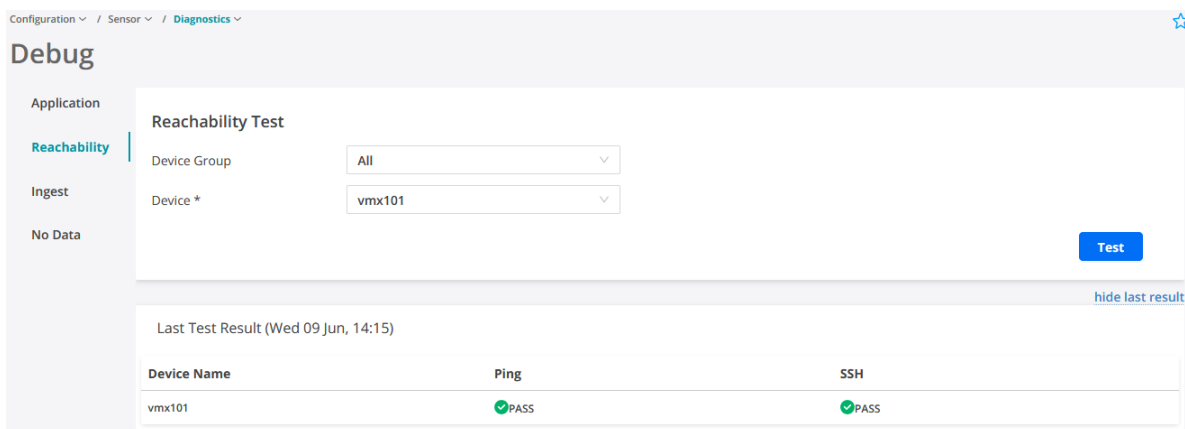
- Or, on the Dashboard page click the desired device in the device list widget, and in the pop-up window click the **REACHABILITY TEST** button.

NOTE: The **REACHABILITY TEST** button does not appear when first adding the device.

2. In the Device Reachability tool, select the desired device from drop-down menu.

3. Click the **Test** button.

4. After a few moments, the test results appear.



Device Name	Ping	SSH
vmx101	✓PASS	✓PASS

The example above shows that the ping test was successful, but the SSH test failed.

Ingest Connectivity Test

Overview

In early versions of Paragon Insights, there was no way to easily determine which ingest methods were supported for a given device; you also had no way to know whether the configured ingest method successfully established a connection with the network device.

Starting with HealthBot Release 2.1.0, the ingest connectivity tool can verify ingest methods where Paragon Insights initiates the connection, such as OpenConfig, iAgent, and SNMP. Paragon Insights does not test UDP-based ingest methods, such as syslog and Native GPB, as the UDP parameters are common to a device group and not specific to a device.

Paragon Insights validates each supported ingest method in its own way:

- OpenConfig: Establishes a gRPC connection with the device using its IP/host name, gRPC port, and credentials
- iAgent: Establishes a NETCONF session with the device using its IP/host name, NETCONF port, and credentials
- SNMP: Executes a simple SNMP GET command; expects to get a reply from the device

This tool provides multiple benefits:

- It helps to identify when there might be missing configuration on the network device side.
- It helps you choose appropriate playbooks and rules that use sensors compatible with the supported ingest methods.
- It helps to identify ingest connectivity issues early on, rather than troubleshoot the “no-data” issue described in the previous section.

Usage Notes

- While this feature is generally intended to help troubleshoot device onboarding, you can use it any time to check ingest connectivity.
- The color coding for the test results is as follows:
 - Green = pass
 - Yellow = error (unable to test)
 - Red = fail
- Any items with yellow or red status will include a message with more detail about the issue.

How to Use the Ingest Connectivity Tool

1. To access the tool:

- Navigate to the **Administration > Debug** page from the left-nav panel, and click the **INGEST** tab.
- Or, on the Dashboard page click the desired device in the device list widget, and in the pop-up window click the **INGEST TEST** button.

Edit test

Hostname / IP address / Range* ⓘ

test

Reachability Test

Ingest Test

System ID to use for JTI ⓘ

Specify in the format - Hostname:ip

Open Config Port Number ⓘ

iAgent Port Number ⓘ

NOTE: The **Ingest Test** button does not appear when first adding the device.

2. In the Ingest Connectivity tool, select the desired device from drop-down menu

3. Click the **Test** button.

4. After a few moments, the test results appear.

Configuration ▾ / Sensor ▾ / Diagnostics ▾

Debug

Application

Reachability

Ingest

No Data

Ingest Test Settings

Sensor Type(s) *

open-config × iAgent × snmp ×

Device Group

dg1 ▾

Device *

vmx101 ▾

Test

[hide last result](#)

Last Test Result (Wed 09 Jun, 14:26)

Device

open-config FAIL for vmx101. DETAILS: gRPC connection failed due to login check failure

vmx101

open-config iAgent snmp

The example above shows that iAgent is supported, OpenConfig is not supported, and SNMP encountered an error.

Debug No-Data

Overview

One of the most common problems that a Paragon Insights users face is “How to debug no-data?”. Determining the root cause is challenging as the issue can occur for a variety of reasons, including:

- Device not reachable
- Device not sending data
- Firewall blocking connections
- Ingest connectivity settings mismatch
- Rule frequency/trigger interval is configured less than the router response time
- Paragon Insights services not running

In early versions of Paragon Insights, you needed to manually look for the problem, checking add-device parameters, reviewing logs, checking the device configuration, and so on.

Starting with HealthBot Release 2.1.0, the debug no-data tool helps to determine why a device or rule is showing a status of “no-data”. The tool takes a sequential, step-by-step approach to determine at which stage incoming data is getting dropped or blocked, as follows:

- Paragon Insights Services
 - Verify that all common and device group-related services are up and running
- Device Reachability
 - Test connectivity to device using ping and SSH
- Ingest Connectivity
 - Verify that the configured ingest session is established
- Raw Data Streaming
 - Verify whether the ingest is receiving any raw data from the devices
 - Likely to be OK if device connectivity is OK
- Field Processing
 - Within rules, verify that the fields working properly, and that the field information is populated in the database

- Trigger Processing
 - Within rules, verify that the trigger settings working as intended, and status information is populated in the database
- API Verification
 - Check for API timeouts that might be affecting the GUI

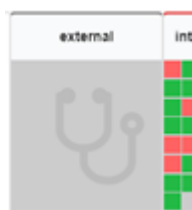
Usage Notes

- The tool runs through the entire sequence of checks, regardless of any issues along the way.
- The test results provide root cause information and advise where to focus your troubleshooting efforts.
- While this feature is generally intended to debug a device when it is marked as no-data, you can use it any time to verify that deployed rules are receiving data.
- This tool does not support rules using a syslog sensor, as the sensor data is event driven and not periodic.
- The color coding for the test results is as follows:
 - Green = pass
 - Yellow = error (unable to test)
 - Red = fail
 - Any items with yellow or red status will include a message with more detail about the issue.

How to Use the Debug No-Data Tool

1. To access the tool:

- On any device health page, click a “no data” tile.



- Or, navigate to the **Administration > Debug** page from the left-nav panel, and select the NO DATA tab.

2. In the Debug No-Data tool, select the desired device group, device, and one or more rules from the drop-down menu
3. Click the **Debug** button.
4. After a few moments, the test results appear.

The screenshot shows the 'Debug' interface for the 'No-Data' tool. On the left is a sidebar with navigation links: 'Application', 'Reachability', 'Ingest', and 'No Data' (which is highlighted). The main area is titled 'No-Data Settings' and contains the following configuration:

- Device Group *: tm-dg-tenant1-juniper
- Device *: vmx107
- Topic *: system.cpu
- Rule(s): check-system-cpu-load-average

A 'Test' button is located in the top right corner. Below the settings, the 'Last Test Result (Wed 09 Jun, 14:38)' is displayed. A summary bar shows: Device Group: tm-dg-tenant1-juniper, Device: vmx107, Topic: system.cpu, Rule(s): check-system-cpu-load-average. Below this is a table of test results:

Service	Status
> Paragon Insights Services	❌
> Device Reachability	✅
> Ingest Connectivity	✅
> Data Streaming	✅
> Field Processing	✅
> Trigger Processing	✅
> API Verification	✅

The example above shows that one common services is not working properly.

Paragon Insights Configuration – Backup and Restore

IN THIS SECTION

- [Back Up the Configuration | 322](#)
- [Restore the Configuration | 322](#)
- [Backup or Restore the Time Series Database \(TSDB\) | 323](#)

Back Up the Configuration

To back up the current Paragon Insights (formerly HealthBot) configuration:

1. Select the **Administration > Backup** option from the left-nav bar.
2. Check the check box for the configuration that you want to backup.
If you also want to back up any configuration changes that have not yet been deployed, toggle the **Backup Undeployed Configuration** switch to the right.
3. Click **Backup**.

Back Up Helper Files

Helper files are the python, yaml, or other externally created files whose tables are referenced in iAgent sensor definitions within rule definitions.

1. Select the **Administration > Backup** option from the left-nav bar.
2. Click the **BACKUP HELPER FILE** button.

Paragon Insights creates a tar archive that you can save to your computer.

Restore the Configuration

To restore the Paragon Insights configuration to a previously backed up configuration:

1. Select the **Administration > Restore** option from the left-nav bar.
2. Click the **Choose File** button.
3. Navigate to the backup file from which you want to restore the configuration, and click **Open**.
Once opened, a list of backup configuration sections appears as buttons under **Select the configuration backup file** heading. By default, all configuration elements in the selected backup file are selected to restore.
4. (Optional) From the list of buttons, you can deselect (change the “-” to “+”) individual files from restoring by clicking on the button.
5. Click **RESTORE CONFIGURATION & DEPLOY** to restore and deploy the configuration.

Restore Helper File

To restore previously backed up helper file archives:

1. Select the **Administration > Restore** option from the left-nav bar.
2. Click the **Choose File** button directly above the second line, “Select the helper backup file”.

3. Locate the backed up helper file in the file browser and click **Open**.
4. Click the **RESTORE CONFIGURATION** button to restore the helper files to Paragon Insights, or the **RESTORE CONFIGURATION & DEPLOY** button to restore the helper files to their original location within Paragon Insights.

Backup or Restore the Time Series Database (TSDB)

Starting with HealthBot Release 3.2.0, you can backup and restore the TSDB separately from other configuration elements. The backup and restore operations for the TSDB are only available through the Paragon Insights CLI. The backup and restore commands are invoked by using a predefined python script, `healthbot.py`. You must have root access to the CLI interface of the Paragon Insights server in order to issue these commands.

Use the following command to set the environment variable:

```
export HB_EXTRA_MOUNT1=/root/.kube/config
```

`HB_EXTRA_MOUNT1` is a variable and user defined.

The generic command and the optional arguments (in square brackets) for performing backup and restore is described here with examples.

```
healthbot tsdb (backup|restore) [-h] [--database DATABASE] [--all] --path PATH
```

The required arguments for the `healthbot tsdb` command are:

- `backup`—perform a backup operation
- `restore`—perform a restore operation
- `--path PATH`—create the backup file or restore the database(s) from the container at `PATH` where `PATH` is `HB_EXTRA_MOUNT1`.
- Starting with Paragon Insights Release 4.0.0, you can use the following commands.
 - `healthbot tsdb stop-services --stop service`
 - `healthbot tsdb start-services --port <portnumber>--start service`

NOTE: In Paragon Insights, the TSDB port is exposed by default. If you need to shut the TSDB port down for security reasons, you can use the `healthbot tsdb stop-services` command. External API queries to TSDB do not need the TSDB port to be exposed. However, if you use external tools such as Grafana, or you need run a query to the TSDB directly (and not through APIs), the TSDB port must be exposed.

Optional Arguments

```
-h, --help          show this help message and exit
--database DATABASE Takes backup (or restore) of the given list of databases. Either
                    database or all flag must be configured
--all               Takes backup (or restores) of all the databases. Either database or
                    all flag must be configured
```

Example – Backup the TSDB and Store it in HB_EXTRA_MOUNT3

```
healthbot tsdb backup --path HB_EXTRA_MOUNT3
```

Example – Restore the TSDB from HB_EXTRA_MOUNT2

```
healthbot tsdb restore --path HB_EXTRA_MOUNT2
```

Change History Table

Feature support is determined by the platform and release you are using. Use [Feature Explorer](#) to determine if a feature is supported on your platform.

Release	Description
3.2.0	Starting with HealthBot Release 3.2.0, you can backup and restore the TSDB separately from other configuration elements

3

CHAPTER

License Management

[Paragon Insights Licensing Overview | 326](#)

[View, Add, or Delete Paragon Insights Licenses | 326](#)

Paragon Insights Licensing Overview

Juniper Networks introduced the Juniper Flex Software Subscription Licensing model to provide an efficient way for you to manage licenses for hardware and software features. Paragon Insights uses this licensing model. For more information, see [Paragon Insights Licensing](#) topic in the Licensing Guide.

You need a license to activate the Paragon Insights graphical user interface (GUI). When you log in to the Paragon Insights GUI for the first time, the **License Management** page appears. Depending on the type of license you add, you will have access to advanced, or standard features of Paragon Insights.

For more information on managing licenses from the Paragon Insights GUI, see "[View, Add, or Delete Paragon Insights Licenses](#)" on page 326.

View, Add, or Delete Paragon Insights Licenses

IN THIS SECTION

- [Add a Paragon Insights License | 327](#)
- [Delete a Paragon Insights License | 328](#)
- [View Paragon Insights Licensing Features | 328](#)
- [View Status and Details of Paragon Insights License | 329](#)

NOTE: For detailed information on licensing, see the [Paragon Insights Licensing](#) topic in the Licensing Guide.

Starting with Paragon Insights Release 4.3.0, when you navigate to the **License Management** page, a warning message asking you to add a tier-based license or upgrade to the new license key format might be displayed.

- If no licenses are added, you must obtain a license key from the [Juniper Agile Licensing Portal](#) and add it to the **License Management** page.
- If licenses are added but the license key format is not compatible with the Paragon Insights software version that you are currently using, you must generate a new license key.

To generate a new license key in the compatible format:

1. Login to [Juniper Agile Licensing Portal](#).
2. Revoke the existing license.

A new license key is generated.

NOTE: After you download the newly generated license key file, you must remove eight lines starting from Issue Date through License Key from the file.

3. Activate the new license key that is compatible with the software version you're running. Select the software version during the activation process. For more information, see the FAQ section in the [Juniper Agile Licensing Portal](#).
4. Add the new license key to the **License Management** page. See "[Add a Paragon Insights License](#)" on page 327.

After you have obtained a Paragon Insights license through the [Juniper Agile Licensing Portal](#), you can manage your licenses from the Paragon Insights graphical user interface (GUI).

The **Settings>License Management** page enables you to:

Add a Paragon Insights License

To add a Paragon Insights license:

1. Navigate to **Settings > License Management** page.
The **License Management** page appears.
2. Click the plus (+) icon in the **Licenses Added** section to add a new license.
The Add License pop-up appears.
3. Click **Browse** and navigate to the location of the license file that you want to add.
Select the file and then click **Open**.
4. The license file is now added to the **Add License** page.
5. Click **Ok** to confirm.

The license you added is now listed in the **Licenses Added** section of the **License Management** page.

Delete a Paragon Insights License

To delete an existing Paragon Insights license:

1. Navigate to **Settings > License Management** page.
The **License Management** page is displayed.
2. From the **Licenses Added** section, click the option button (available in beginning the row) of the license you want to delete, and then click **Delete**.
The **Confirm Delete** page appears.
3. Click **Yes** to delete the license.

View Paragon Insights Licensing Features

To view Paragon Insights licensing features, navigate to **Settings > License Management** page.

The **Features Summary** section, as given in [Table 23 on page 328](#) provides information on the Paragon Insights feature licensing attributes.

Table 23: Feature License Attributes

Attribute	Description
Feature	View Paragon Insights license name.
Description	Brief description of the Paragon Insights feature license.
License Limit	The number of valid Paragon Insights licenses successfully added and available for use. NOTE: Starting in Paragon Insights Release 4.3.0, irrespective of the number of PIN-Advanced, and PIN-Standard platform licenses that you add, the maximum license limit is 1. However, with PIN-Devices licenses, the limit changes with the number of licenses that you add.
Usage Count	The number of available licenses that are currently in use in this instance of Paragon Insights.
Valid Until	Date and time when the license expires.

Table 23: Feature License Attributes (*Continued*)

Attribute	Description
Compliance	<p>Color definitions (dot indicator) that determine license compliance:</p> <ul style="list-style-type: none"> • Green—Feature licenses are in compliance with Juniper's End User License Agreement. • Yellow—Device feature licenses are $\geq 90\%$ of the limit. You are getting close to running out of licenses. This status applies only to device feature licenses. • Red—Feature licenses are not in compliance with Juniper's End User License Agreement. Click the red dot to view details about the compliance issue.

View Status and Details of Paragon Insights License

To view status and details of a Paragon Insights license, navigate to **Settings > License Management** page. The **Licenses Added** section provides information on the added Paragon Insights licenses. You can also click the option button (available in beginning the row) of the license you want to view, and then click **More > Details** to view more information about the license.

Table 24: View Details of Licenses Added

Attribute	Description
License ID	View the Paragon Insights license identification number generated through the Juniper Agile Licensing Portal.
SKU Name	<p>View the Paragon Insights software licensing package name.</p> <p>NOTE: In Paragon Insights Release 4.3.0, SKU Name is not available in the Added License section of the License Management page. This is because the new license key format does not support SKU Name.</p>
Customer ID	<p>View the customer identification.</p> <p>NOTE: The customer ID might not be displayed after you add a license. This is because the customer ID is not embedded in the new license key format.</p>

Table 24: View Details of Licenses Added *(Continued)*

Attribute	Description
Order Type	View the order type (commercial, demo, education, emergency, lab, and unknown).
Validity Type	View the validity type of a license (date-based or permanent).
Start Date	View the start date of the Paragon Insights license.
End Date	View the end date of the Paragon Insights license.
State	Displays the state of the license.
Feature ID	View the feature license identification number.
Feature Name	View the Paragon Insights feature license name.
Feature Description	View a brief description of the Paragon Insights feature license.
License Count	View the entitled license count for this Paragon Insights feature.

RELATED DOCUMENTATION

| [Paragon Insights Licensing Overview](#) | 326