

Contrail™

Contrail Analytics and Troubleshooting Guide

Published
2020-12-02

Release
5.1

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.

Contrail™ Contrail Analytics and Troubleshooting Guide

5.1

Copyright © 2020 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 the Documentation | xiii

Documentation and Release Notes | xiii

Documentation Conventions | xiii

Documentation Feedback | xvi

Requesting Technical Support | xvi

Self-Help Online Tools and Resources | xvii

Creating a Service Request with JTAC | xvii

1

AppFormix in Contrail Command

Configuring AppFormix | 2

AppFormix Overview | 2

Configuring AppFormix Alarms using Contrail Command | 3

AppFormix Alarms Overview | 3

AppFormix Alarms Operation | 4

Sliding Window Analysis | 5

Static Alarm | 6

Dynamic Alarm | 7

Alarm Definition | 8

Required Parameters for Dynamic Alarms | 14

States for Alarm Mode | 15

Aggregation Functions for Alarm Processing | 16

Comparison Functions for Alarm Processing | 18

Dynamic Baseline Examples | 21

Configuring an Alarm Rule | 23

Configuring Instances in AppFormix | 25

Instance Details Overview | 25

Creating Instances | 26

Viewing Cluster Node Details and Metric Values | 31

Time | 32

Legend | 32

Chart Data Values | 32

Viewing Cluster Node Details and Host Charts | 33

Metrics Collected by AppFormix | 35

Host CPU Data Metrics | 36

Host Disk Metrics | 37

Host Memory Usage | 38

Host Mount Metrics | 39

Host Network Data | 39

Instances | 40

Network Device | 42

Contrail Release 5.0 vRouter Plug-In | 45

OpenContrail vRouter on a Host | 46

OpenStack Project in Chart View | 47

RabbitMQ Service | 48

ScaleIO Service | 51

gRPC Sensors | 54

2

Configuring Contrail

Optimizing Contrail | 58

vRouter Command Line Utilities | 58

Overview | 58

vif Command | 59

flow Command | 62

vrfstats Command | 64

rt Command | 66

dropstats Command | 67

mpls Command | 71

mirror Command | 73

vxlan Command | 76

nh Command | 77

Monitoring and Troubleshooting Contrail

Configuring Traffic Mirroring to Monitor | 82

Configuring Traffic Analyzers and Packet Capture for Mirroring | 82

- Traffic Analyzer Images | 82

- Configuring Traffic Analyzers | 83

- Setting Up Traffic Mirroring Using Configure > Networking > Services | 83

Configuring Interface Monitoring and Mirroring | 89

Mirroring Enhancements | 91

- Mirroring Specified Traffic | 91

- Configuring Headers and Next Hops | 91

- How Mirroring is Implemented | 92

Analyzer Service Virtual Machine | 92

- Packet Format for Analyzer | 92

- Metadata Format | 93

- Wireshark Changes | 94

- Troubleshooting Packet Display | 94

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring) | 95

Understanding Contrail Analytics | 97

Understanding Contrail Analytics | 97

Contrail Alerts | 98

- Alert API Format | 99

- Analytics APIs for Alerts | 100

- Analytics APIs for SSE Streaming | 100

- Built-in Node Alerts | 101

Underlay Overlay Mapping in Contrail | 102

- Overview: Underlay Overlay Mapping using Contrail Analytics | 103

- Underlay Overlay Analytics Available in Contrail | 103

- Architecture and Data Collection | 104

- New Processes/Services for Underlay Overlay Mapping | 104

- External Interfaces Configuration for Underlay Overlay Mapping | 105

- Physical Topology | 106

- SNMP Configuration | 106

Link Layer Discovery Protocol (LLDP) Configuration	106
IPFIX and sFlow Configuration	107
Sending pRouter Information to the SNMP Collector in Contrail	109
pRouter UVEs	110
Contrail User Interface for Underlay Overlay Analytics	111
Enabling Physical Topology on the Web UI	111
Viewing Topology to the Virtual Machine Level	112
Viewing the Traffic of any Link	112
Trace Flows	113
Search Flows and Map Flows	114
Overlay to Underlay Flow Map Schemas	114
Module Operations for Overlay Underlay Mapping	118
SNMP Collector Operation	118
Topology Module Operation	120
IPFIX and sFlow Collector Operation	121
Troubleshooting Underlay Overlay Mapping	122
Script to add pRouter Objects	123

Configuring Contrail Analytics | 126

Analytics Scalability	127
High Availability for Analytics	128
System Log Receiver in Contrail Analytics	129
Overview	129
Redirecting System Logs to Contrail Collector	129
Exporting Logs from Contrail Analytics	130
Sending Flow Messages to the Contrail System Log	130
User Configuration for Analytics Alarms and Log Statistics	131
Configuring Alarms Based on User-Visible Entities Data	131
Examples: Detecting Anomalies	133
Configuring the User-Defined Log Statistic	135
Implementing the User-Defined Log Statistic	138

Alarms History | 141**Viewing Alarms History | 141****Node Memory and CPU Information | 143****Role- and Resource-Based Access Control for the Contrail Analytics API | 144****Configuring Analytics as a Standalone Solution | 145****Overview: Contrail Analytics as a Standalone Solution | 145****Configuration Examples for Standalone | 146****Examples: Inventory File Controller Components | 146****JSON Configuration Examples | 147****Configuring Secure Sandesh and Introspect for Contrail Analytics | 149****Configuring Secure Sandesh Connection | 149****Configuring Secure Introspect Connection | 149****Using Contrail Analytics to Monitor and Troubleshoot the Network | 151****Monitoring the System | 152****Debugging Processes Using the Contrail Introspect Feature | 155****Monitor > Infrastructure > Dashboard | 159****Monitor Dashboard | 159****Monitor Individual Details from the Dashboard | 160****Using Bubble Charts | 161****Color-Coding of Bubble Charts | 161****Monitor > Infrastructure > Control Nodes | 162****Monitor Control Nodes Summary | 162****Monitor Individual Control Node Details | 164****Monitor Individual Control Node Console | 165****Monitor Individual Control Node Peers | 168****Monitor Individual Control Node Routes | 169****Monitor > Infrastructure > Virtual Routers | 171****Monitor vRouters Summary | 171****Monitor Individual vRouters Tabs | 173****Monitor Individual vRouter Details Tab | 173****Monitor Individual vRouters Interfaces Tab | 175****Monitor Individual vRouters Networks Tab | 176****Monitor Individual vRouters ACL Tab | 177**

- Monitor Individual vRouters Flows Tab | 179
- Monitor Individual vRouters Routes Tab | 180
- Monitor Individual vRouter Console Tab | 181

Monitor > Infrastructure > Analytics Nodes | 183

- Monitor Analytics Nodes | 183
- Monitor Analytics Individual Node Details Tab | 184
- Monitor Analytics Individual Node Generators Tab | 186
- Monitor Analytics Individual Node QE Queries Tab | 186
- Monitor Analytics Individual Node Console Tab | 187

Monitor > Infrastructure > Config Nodes | 189

- Monitor Config Nodes | 189
- Monitor Individual Config Node Details | 190
- Monitor Individual Config Node Console | 191

Monitor > Networking | 193

- Monitor > Networking Menu Options | 193
- Monitor -> Networking -> Dashboard | 194
- Monitor > Networking > Projects | 196
- Monitor Projects Detail | 196
- Monitor > Networking > Networks | 199

Query > Flows | 203

- Query > Flows > Flow Series | 204
- Example: Query Flow Series | 206
- Query > Flow Records | 208
- Query > Flows > Query Queue | 210

Query > Logs | 212

- Query > Logs Menu Options | 212
- Query > Logs > System Logs | 212
- Sample Query for System Logs | 214
- Query > Logs > Object Logs | 216

Common Support Answers | 218

Debugging Ping Failures for Policy-Connected Networks | 218

Debugging BGP Peering and Route Exchange in Contrail | 225

Example Cluster | 225

Verifying the BGP Routers | 226

Verifying the Route Exchange | 228

Debugging Route Exchange with Policies | 231

Debugging Peering with an MX Series Router | 232

Debugging a BGP Peer Down Error with Incorrect Family | 234

Configuring MX Peering (iBGP) | 237

Checking Route Exchange with an MX Series Peer | 239

Checking the Route in the MX Series Router | 240

Troubleshooting the Floating IP Address Pool in Contrail | 242

Example Cluster | 243

Example | 244

Example: MX80 Configuration for the Gateway | 245

Ping the Floating IP from the Public Network | 248

Troubleshooting Details | 248

Get the UUID of the Virtual Network | 249

View the Floating IP Object in the API Server | 249

View floating-ips in floating-ip-pools in the API Server | 254

Check Floating IP Objects in the Virtual Machine Interface | 257

View the BGP Peer Status on the Control Node | 261

Querying Routes in the Public Virtual Network | 262

Verification from the MX80 Gateway | 264

Viewing the Compute Node Vnsw Agent | 266

Advanced Troubleshooting | 270

Removing Stale Virtual Machines and Virtual Machine Interfaces | 273

Problem Example | 273

Show Virtual Machines | 274

Show Virtual Machines Using Python API | 276

Delete Methods | 277

Troubleshooting Link-Local Services in Contrail | 278

Overview of Link-Local Services | 278

Troubleshooting Procedure for Link-Local Services | 279

Metadata Service | 281

Troubleshooting Procedure for Link-Local Metadata Service | 281

Contrail Commands and APIs

Contrail Commands | 284

Getting Contrail Node Status | 284

Overview | 284

UVE for NodeStatus | 284

Node Status Features | 285

Using Introspect to Get Process Status | 293

contrail-status script | 294

contrail-logs (Accessing Log File Messages) | 297

Command-Line Options for Contrail-Logs | 297

Option Descriptions | 298

Example Uses | 299

contrail-status (Viewing Node Status) | 301

contrail-version (Viewing Version Information) | 303

Backing Up Contrail Databases Using JSON Format | 305

Preliminary Cautions | 305

Simple Backup Using JSON Format | 306

Restore Simple Database Backup | 306

Example Backup and Restore With JSON | 307

Example: Perform Simple Backup | 308

Example: Perform Restore | 309

Contrail Application Programming Interfaces (APIs) | 314

Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs) | 314

User-Visible Entities | 315

Common UVEs in Contrail | 316

Virtual Network UVE | 316

Virtual Machine UVE | 317

- vRouter UVE | 317
- UEs for Contrail Nodes | 318
- Wild Card Query of UVEs | 318
- Filtering UVE Information | 318

Log and Flow Information APIs | 329

- HTTP GET APIs | 329
- HTTP POST API | 329
- POST Data Format Example | 330
- Query Types | 332
- Examining Query Status | 332
- Examining Query Chunks | 332
- Example Queries for Log and Flow Data | 333

Working with Neutron | 337

- Data Structure | 337
- Network Sharing in Neutron | 338
- Commands for Neutron Network Sharing | 338
- Support for Neutron APIs | 339
- Contrail Neutron Plugin | 339
- DHCP Options | 340
- Incompatibilities | 340

Support for Amazon VPC APIs on Contrail OpenStack | 341

- Overview of Amazon Virtual Private Cloud | 341
- Mapping Amazon VPC Features to OpenStack Contrail Features | 342
- VPC and Subnets Example | 342
- Euca2ools CLI for VPC and Subnets | 343
- Security in VPC: Network ACLs Example | 344
- Euca2ools CLI for Network ACLs | 345
- Security in VPC: Security Groups Example | 346
- Euca2ools CLI for Security Groups | 347
- Elastic IPs in VPC | 347
- Euca2ools CLI for Elastic IPs | 348
- Euca2ools CLI for Route Tables | 348
- Supported Next Hops | 349
- Internet Gateway Next Hop Euca2ools CLI | 349

NAT Instance Next Hop Euca2ools CLI | **349**

Example: Creating a NAT Instance with Euca2ools CLI | **350**

About the Documentation

IN THIS SECTION

- Documentation and Release Notes | **xiii**
- Documentation Conventions | **xiii**
- Documentation Feedback | **xvi**
- Requesting Technical Support | **xvi**

Use this guide to understand AppFormix and Contrail analytics. AppFormix provides monitoring availability of the Contrail control plane services. Contrail analytics nodes are responsible for the collection of system state information, usage statistics, and debug information from all of the software modules across all of the nodes of the system.

Documentation and Release Notes

To obtain the most current version of all Juniper Networks[®] technical documentation, see the product documentation page on the Juniper Networks website at <https://www.juniper.net/documentation/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <https://www.juniper.net/books>.

Documentation Conventions

[Table 1 on page xiv](#) defines notice icons used in this guide.

Table 1: Notice Icons







Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

Table 2 on page xiv defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: user@host> configure
Fixed-width text like this	Represents output that appears on the terminal screen.	user@host> show chassis alarms No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# set system domain-name <i>domain-name</i>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level. The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	stub <default-metric <i>metric</i> >;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast (<i>string1</i> <i>string2</i> <i>string3</i>)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [<i>community-ids</i>]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i> ; retain; } } }
; (semicolon)	Identifies a leaf statement at a configuration hierarchy level.	

GUI Conventions

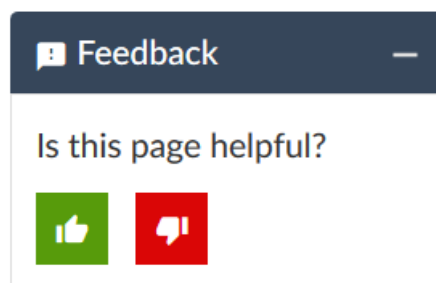
Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none"> In the Logical Interfaces box, select All Interfaces. To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

We encourage you to provide feedback so that we can improve our documentation. You can use either of the following methods:

- Online feedback system—Click TechLibrary Feedback, on the lower right of any page on the [Juniper Networks TechLibrary](#) site, and do one of the following:



- Click the thumbs-up icon if the information on the page was helpful to you.
- Click the thumbs-down icon if the information on the page was not helpful to you or if you have suggestions for improvement, and use the pop-up form to provide feedback.
- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active Juniper Care or Partner Support Services support contract, or are

covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <https://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <https://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <https://www.juniper.net/customers/support/>
- Search for known bugs: <https://prsearch.juniper.net/>
- Find product documentation: <https://www.juniper.net/documentation/>
- Find solutions and answer questions using our Knowledge Base: <https://kb.juniper.net/>
- Download the latest versions of software and review release notes: <https://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <https://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <https://www.juniper.net/company/communities/>
- Create a service request online: <https://myjuniper.juniper.net>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://entitlementsearch.juniper.net/entitlementsearch/>

Creating a Service Request with JTAC

You can create a service request with JTAC on the Web or by telephone.

- Visit <https://myjuniper.juniper.net>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <https://support.juniper.net/support/requesting-support/>.

1

PART

AppFormix in Contrail Command

Configuring AppFormix | 2

Configuring AppFormix

IN THIS CHAPTER

- AppFormix Overview | 2
- Configuring AppFormix Alarms using Contrail Command | 3
- Configuring Instances in AppFormix | 25
- Viewing Cluster Node Details and Metric Values | 31
- Metrics Collected by AppFormix | 35

AppFormix Overview

AppFormix is a cloud service optimization tool that provides advanced monitoring, scheduling, and performance management for software-defined infrastructure, where containers and virtual machines (VMs) can have life cycles much shorter than in traditional development environments.

AppFormix leverages big-data analytics and machine learning in a distributed architecture that puts the power of self-driving infrastructure at the core of most any cloud. It redefines the state-of-the-art in telemetry and management across software-defined infrastructure and application software layers. On top of all of this, real-time and historic monitoring, performance visibility and dynamic optimization features improve cloud orchestration, security, accounting and planning to users.

Starting with Contrail release 5.1, the following AppFormix features are supported in Contrail Command:

- Installing AppFormix using Contrail Command
- Configuring AppFormix Alarms using Contrail Command
- Configuring Instances in AppFormix
- Viewing Cluster Node Details and Metric Values

NOTE: For information about installing AppFormix, see the *Contrail Installation and Configuration Guide*. For more information about AppFormix, see the [AppFormix User Guide](#).

RELATED DOCUMENTATION

[Installing AppFormix using Contrail Command](#)

[Configuring AppFormix Alarms using Contrail Command | 3](#)

[Configuring Instances in AppFormix | 25](#)

[Viewing Cluster Node Details and Metric Values | 31](#)

Configuring AppFormix Alarms using Contrail Command

IN THIS SECTION

- [AppFormix Alarms Overview | 3](#)
- [AppFormix Alarms Operation | 4](#)
- [Alarm Definition | 8](#)
- [Configuring an Alarm Rule | 23](#)

With AppFormix Alarms, you can configure an alarm to be generated when a condition is met in the infrastructure. AppFormix performs distributed analysis of metrics at the point of collection for efficient and responsive detection of events that match an alarm. AppFormix has two types of alarms:

Static—User-provided static threshold is used for comparison.

Dynamic—Dynamically-learned adaptive threshold is used for comparison.

NOTE: In order to configure alarms, your AppFormix license subscription must be active.

AppFormix Alarms Overview

For both static and dynamic alarms, AppFormix Agent continuously collects measurements of metrics (see [“Metrics Collected by AppFormix” on page 35](#)) for different entities, such as hosts, instances, and network devices. Beyond simple collection, the agent also analyzes the stream of metrics at the time of collection to identify alarm rules that match. For a particular alarm, the agent aggregates the samples according to a user-specified function (average, standard deviation, min, max, sum) and produces a single measurement for each user-specified measurement interval. For a given measurement interval, the agent compares each

measurement to a threshold. For an alarm with a static threshold, a measurement is compared to a fixed value using a user-specified comparison function (above, below, equal). For dynamic thresholds, a measurement is compared with a value learned by AppFormix over time.

You can further configure alarm parameters that require multiple intervals to match. This allows you to configure alarms to match sustained conditions, while also detecting performance over small time periods. Maximum values over a wide time range can be over-exaggerate conditions. Yet, averages can dilute the information. A balance is better achieved by measuring over small intervals and watching for repeated matches in multiple intervals. For example, to monitor CPU usage over a three-minute period, an alarm may be configured to compare average CPU utilization over fiveseconds intervals, yet only raise an alarm when 36 (or some subset of 36) intervals match the alarm condition. This provides better visibility into sustained performance conditions than a simple average or maximum over three minutes.

Dynamic thresholds enable outlier detection in resource consumption based on historical trends. Resource consumption may vary significantly at various hours of the day and days of the week. This makes it difficult to set a static threshold for a metric. For example, 70% CPU usage may be considered normal for Monday mornings between 10:00 AM and 12:00 PM, but the same amount of CPU usage may be considered abnormally high for Saturday nights between 9:00 PM and 10:00 PM.

With dynamic thresholds, AppFormix learns trends in metrics across all resources in scope to which an alarm applies. For example, if an alarm is configured for a host aggregate, AppFormix learns a baseline from metric values collected for hosts in that aggregate. Similarly, an alarm with a dynamic threshold configured for a project learns a baseline from metric values collected for instances in that project. Then, the agent generates an alarm when a measurement deviates from the baseline value learned for a particular time period.

When creating an alarm with a dynamic threshold, you select a metric, a period of time over which to establish a baseline, and the sensitivity to measurements that deviate from the baseline. The sensitivity can be configured as *high*, *medium*, or *low*. Higher sensitivity will report smaller deviations from the baseline and vice versa.

AppFormix Alarms Operation

IN THIS SECTION

- [Sliding Window Analysis | 5](#)
- [Static Alarm | 6](#)
- [Dynamic Alarm | 7](#)

AppFormix Agent performs distributed, real-time statistical analysis on a time-series data stream. Agent analyzes metrics over multiple measurement intervals using a configurable sliding window mechanism. An alarm is generated when the AppFormix Agent determines that metric data matches the alarm criteria over a configurable number of measurement intervals. The type of sample aggregation and the threshold for an alarm is configurable. Two types of alarms are supported: static and dynamic. The difference is how the threshold is determined and used to compare measured metric data. The following sections describe the overall sliding window analysis, and explains the details of static thresholds and dynamic baselines used by the analysis.

Sliding Window Analysis

AppFormix Agent evaluates alarms using sliding window analysis. The sliding window analysis compares a stream of metrics within a configurable measurement interval to a static threshold or dynamic baseline. The length of each measurement interval is configurable to one-second granularity. In each measurement interval, raw time-series data samples are combined using an aggregation function, such as *average*, *max*, and *min*. The aggregated value is compared against the static threshold or dynamic baseline using a configurable comparison function, such as *above* or *below*. Multiple measurement intervals comprise a sliding window. A configurable number of intervals in the sliding window must match the rule criteria for the agent to generate a notification for the alarm.

Figure 1: Alarm Generation Mechanics

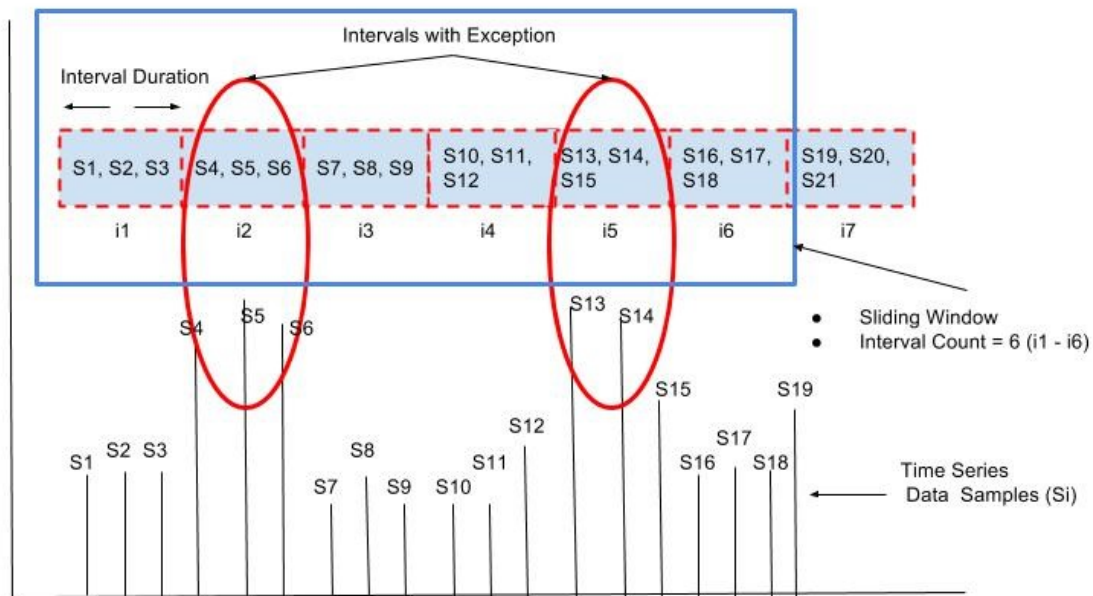


Figure 1 on page 5 shows an example in which the sliding window consists of six adjacent measurement intervals (i1 to i6), as specified by the Interval Count parameter. In measurement interval i1, the average of samples S1, S2, S3 is computed as S_{avg} . Depending on the alarm type *static* or *dynamic*, S_{avg} is then

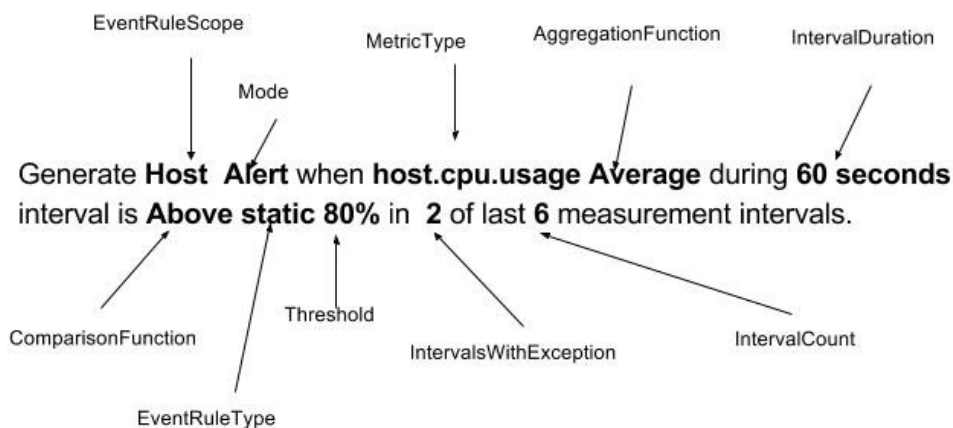
compared with the configured static threshold or dynamically learned baseline using a user-specified comparison function such as *above* or *below*. The output of the comparison determines whether a specific measurement interval is marked as an *interval with exception*. This evaluation is repeated for each measurement interval within the sliding window (for example, i1 to i6).

In the example in [Figure 1 on page 5](#), the agent determines that two intervals, i2 and i5, are *intervals with exception* by comparing the aggregate value for the measurement interval with a static threshold or dynamic baseline, depending on alarm type. Assuming interval i1 is the first interval for which the alarm is configured, the alarm becomes active at end of interval i6, when AppFormix Agent determines that at least two out of the most recent six measurement intervals are marked as exceptions. When an alarm is configured using the Dashboard, Interval Count, and Intervals with Exception are set to 1 by default. As a result, the agent can generate an alarm after processing data for one measurement interval.

Static Alarm

A static alarm threshold is provided at the time of alarm definition. [Figure 2 on page 6](#) depicts an example of a static alarm definition, followed by the equivalent JSON used for API configuration of an alarm. The condition defined in the example is to evaluate an average of **host.cpu.usage** samples over a 60 second measurement interval. The measured value is compared against a static threshold of 80% to determine if a given measurement interval matches the alarm rule. [Figure 2 on page 6](#) identifies the components in a static alarm definition.

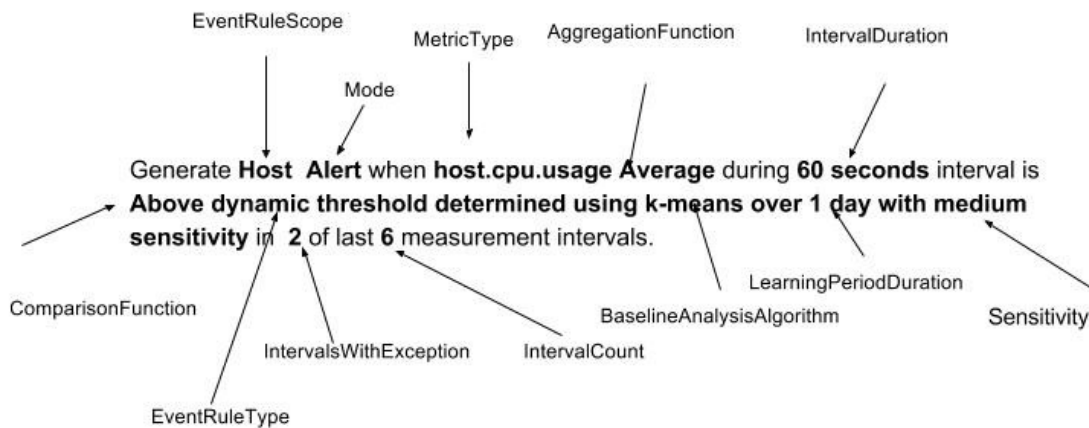
Figure 2: Static Alarm Definition



Dynamic Alarm

A dynamic alarm threshold is learned by AppFormix using historical data for the set of entities for which an alarm is configured. [Figure 3 on page 7](#) shows an example of a dynamic alarm definition and identifies the components in a dynamic alarm definition.

Figure 3: Dynamic Alarm Definition



When using a dynamic threshold, you do not configure a static threshold value. Instead, you specify three parameters that control how the learning is performed. The learning algorithm produces a baseline across the entities. The baseline is comprised of a mean value and a standard deviation. The baseline is updated continuously as additional metric data is collected.

Following is a list of the three learning parameters and information about how they work:

BaselineAnalysisAlgorithm—Selects the machine learning algorithm used for determining the dynamic threshold. The following algorithms are available:

k-means—AppFormix employs a k-means algorithm to produce an expected operating range for a set of entities at a granularity of each hour of each day (up to one week). The learned baselines are computed using data from a configurable learning period duration. The baselines are updated continuously over time, based on the most recent data. The k-means Baseline Analysis Algorithm is useful for observing performance that is unexpected for a given time of day.

For example, a k-means algorithm may learn a dynamic baseline for 1:00 PM - 2:00 PM that may be 80% +/- 10%, whereas, the baseline between 3:00 AM - 4:00 AM may be 20% +/- 5%. An

alarm is raised if the measured metric is 75% of the value between 3:00 AM - 4:00 AM, but the same measurement is acceptable during 1:00 PM - 2:00 PM time period.

ewma—The Exponentially Weighted Moving Average (EWMA) algorithm produces a single baseline that is updated hourly. The configurable Learning Period duration allows you to control the relative weight assigned to recent data versus older data. This algorithm is useful to create an alarm that can detect sudden changes in a metric.

For example, an EWMA algorithm can learn a dynamic baseline of 60% +/- 10% from data over the last 24 hours. This baseline is used for the next 1-hour interval to determine if real-time data deviates from the normal operating region. After every 1-hour interval, the EWMA baseline is updated and a new updated baseline is used for alarm generation in the future.

LearningPeriodDuration—A dynamic baseline is determined using the historical data. This parameter determines the length of time period from which most recent historical data is used to compute a dynamic baseline. For example, 1 hour, 1 day, or 1 week. At the time of rule configuration, AppFormix might not yet have enough historical data for a given entity. In this case, learning is performed as data becomes available. Alarm evaluation begins after one Learning Period of data is available and baselines are generated.

Sensitivity—The sensitivity of a dynamic alarm controls the allowable magnitude of deviation from the learned mean. The sensitivity parameter controls a multiplier of the learned standard deviation. You can select *low*, *medium*, or *high* as sensitivity. AppFormix Agent compares real-time measurements to the range defined by:

$$\text{mean} - \text{sensitivity} * \text{std_dev} < x < \text{mean} + \text{sensitivity} * \text{std_dev}$$

Alarm Definition

IN THIS SECTION

- [Required Parameters for Dynamic Alarms | 14](#)
- [States for Alarm Mode | 15](#)
- [Aggregation Functions for Alarm Processing | 16](#)
- [Comparison Functions for Alarm Processing | 18](#)
- [Dynamic Baseline Examples | 21](#)

Figure 2 on page 6 shows an example of a static alarm definition. Every alarm definition has the following components shown in Table 3 on page 9.

Table 3: Alarm Definition Components

Item	Options	Description
Module	Alarms, Service Alarms	When Alarms is selected, you can configure alarms for entities such as hosts, instances, and network devices. When Service Alarms is selected, then you are able to configure alarms for services such as RabbitMQ, MySQL, ScaleIO, and OpenStack services.
Alarm Rule Type	Static, Dynamic	<p>This determines the type of threshold that alarm uses to determine if alarm should be generated or not. Following are the two types that are supported.</p> <ul style="list-style-type: none"> • Static—When an alarm is defined as static, the rule definition should include a predefined static threshold. For example, cpu.usage static threshold can be 80%. • Dynamic—When an alarm is defined as dynamic, the baseline is learned using historical data. Additional parameters are required such as baseline analysis algorithm, learning period duration, and sensitivity.
Name	Alarm name	A name identifies the alarm. Name is displayed in the Dashboard and is the user-facing identifier for external notification systems.
Scope	Host, Instance, Network Device, Virtual Network	Type of entity such as host, instance, or network device to which the alarm applies. For example, if scope is selected as Instance , then you can further select to configure rule to all instances present in the infrastructure, or instances that are present in a specific project or an aggregate.
Service	RabbitMQ, MySQL, Ceph, OpenStack, Cassandra, Contrail, ScaleIO	When selected, you can configure alarms for RabbitMQ, MySQL, Ceph, OpenStack, Cassandra, Contrail, and ScaleIO services.

Table 3: Alarm Definition Components (*continued*)

Item	Options	Description
Metric Scope	Cluster, Node, Queue	Select the metric scope of what you want to monitor, such as cluster, node, or queue and then the metric to monitor.
Object	Options dependent on Metric Scope selection.	Object that will be monitored.
Generate	Event, Alarm	When conditions for the alarm are met, generate an event or alarm.
For Metric	cpu.usage, memory.usage	Metrics that will be monitored. For example, host.cpu.usage or instance.cpu.usage.
When	Value	—
Interval (seconds)	Value in seconds	The duration of one measurement interval in seconds. Depending on the sampling frequency of a metric under observation, one or more raw samples might be received within an interval duration. All raw samples received within Interval duration are processed using aggregation functions such as average, sum, max, min, and std-dev.
Is	Value	Example: When <i>Value</i> Is Above Threshold -8. Italics in example represent variables.

Table 3: Alarm Definition Components (*continued*)

Item	Options	Description
Threshold	Threshold value	<p>A numeric value to which measurements are compared. AppFormix supports two types of thresholds: static or dynamic.</p> <ul style="list-style-type: none"> • Static Threshold—A fixed value that is specified when an alarm is configured. For example host.cpu.usage above 90%, where 90% is the static threshold. • Dynamic Threshold—The threshold is learned dynamically by the system. Unsupervised learning is used to learn about historical trends to determine the dynamic threshold. For example, if an event rule is defined for Host aggregate, then the dynamic baseline is determined for the aggregate by applying the baseline analysis algorithm to data received from all member hosts of the aggregate. Figure 6 on page 22 shows the dynamic baseline determined using the most recent 24-hour time frame of historical data and k-means clustering algorithm. This baseline is used for the next 24 hours for alarm generation while considering the hour of the day and its corresponding baseline mean and standard deviation. For example, on Tuesday 8:00 AM - 9:00 AM, a baseline computed for Monday 8:00 AM - 9:00 AM is used as a reference threshold for alarm generation. <p>The required parameters for dynamic threshold are:</p> <ul style="list-style-type: none"> • Baseline Analysis Algorithm • Learning Period Duration • Sensitivity <p>Table 4 on page 14 describes the required parameters for a dynamic alarm and the supported options.</p>

Table 3: Alarm Definition Components (*continued*)

Item	Options	Description
Baseline Analysis Algorithm	k-means, ewma	Table 4 on page 14 describes these options. See Figure 6 on page 22 and Figure 7 on page 22 for baseline analysis examples
Learning Period Duration	1 week, 1 month	Table 4 on page 14 describes these options.
Sensitivity	Low, medium, high	Table 4 on page 14 describes these options.
Severity	None, information, warning, error, critical	Indicates seriousness of the alarm. <i>Critical</i> indicates a major alarm. <i>Information</i> indicates a minor alarm.
Advanced	When selected, includes Intervals with Exception, Interval Count, and Status.	—
Aggregate/Project	All hosts, all instances. AggregateId, ProjectId	Select the set of entities an alarm will monitor. If Scope is Instance , then you can configure an alarm for the set of instances present in a specific project, aggregate, or all instances in the infrastructure. If Scope is Host , then you can configure an alarm for a set of hosts present in a specific aggregate or all hosts in the infrastructure.
Alarm Mode	Alert, Event	Mode can be configured as an alert or event.
Aggregation Function	Average, Max, Min, Sum, Std-dev	Determines how data samples received in one measurement interval are processed to generate an aggregated value for comparison. Agent collects multiple samples of a metric during a measurement interval. Agent combines the samples according to the aggregation function, in order to determine a single value for comparison with the threshold (static or dynamic) in a measurement interval. Table 7 on page 17 lists and describes the aggregation functions for alarm processing.

Table 3: Alarm Definition Components (*continued*)

Item	Options	Description
Comparison Function	Above, Below, Equal, Increasing-at-a-minimum-rate-of, Decreasing-at-a-minimum-rate-of	Determines how to compare output of the Aggregation Function with the static or dynamic threshold. Table 8 on page 18 shows different comparison functions supported for AppFormix alarms. Figure 4 on page 20 and Figure 5 on page 21 show examples of the Comparison Function, showing both increases and decreases at a minimum rate.
Static Threshold	When alarm rule type is “static”	—
Alarm Severity	None, information, warning, error, critical	Indicates seriousness of the alarm. <i>Critical</i> indicates a major alarm. <i>Information</i> indicates a minor alarm.
Notification	None, PagerDuty, Custom Service, Service Now, Slack	Methods of notification alerting you to conditions of operation.
Intervals with Exception	For example, “2”	This is the minimum number of measurement intervals within the sliding window for which a condition for an alarm must be met to raise the alarm. In Figure 3 on page 7 , there are two Intervals with Exception: i2 and i5. When configuring an alarm in the Dashboard, Intervals with Exception is set to 1 by default. The Interval with Exception can be specified in the Dashboard by selecting Monitoring > Alarms > Add Rule . Intervals with Exception can not be greater than the Interval Count.

Table 3: Alarm Definition Components (*continued*)

Item	Options	Description
Interval Count	For example, "3"	Maximum number of adjacent measurement intervals for which a statistical analysis is performed before deciding if an alarm is generated or not. In Figure 3 on page 7 , there are 6 measurement Intervals (i1 to i6) in the sliding window. Each measurement interval has duration specified by the Interval Duration parameter. When configuring an alarm in Dashboard, Interval Count is set to 1 by default. The Interval Count can be specified in the Dashboard by selecting Monitoring > Alarms > Add New Rule .
Status	Enable, Disable	Used to set and also verify status of alarm rule. Set status as enabled or disabled.

Required Parameters for Dynamic Alarms

[Table 4 on page 14](#) describes the required parameters for a dynamic alarm and the supported options.

Table 4: Required Parameters for Dynamic Alarm

Required Parameters for Dynamic Threshold	Description	Supported Options
Baseline Analysis Algorithm	Baseline Analysis Algorithm is used to perform unsupervised learning on historical data. The baseline analysis is performed continuously as new data is received.	<ul style="list-style-type: none"> • K-Means clustering • Exponential Weighted Mean Average (EWMA)

Table 4: Required Parameters for Dynamic Alarm (*continued*)

Required Parameters for Dynamic Threshold	Description	Supported Options
Learning Period Duration	<p>The Learning Period Duration specifies the amount of historical data used by the Baseline Analysis Algorithm to determine a baseline. The dynamic baseline is continuously updated using data from the most recent Learning Duration.</p> <p>When a dynamic alarm is configured, baseline analysis is performed using data from the most recent Learning Duration, if available. If there is not sufficient data available, AppFormix Agent evaluates metrics as soon as enough data is present to learn the first set of baselines.</p> <p>Example: When Learning Duration is 1 day, the agent compares metrics to per-hour baselines for the last 24 hours.</p> <p>Example: When Learning Duration is 1 week, the agent compares metrics to per-hour baselines for the last 7 x 24 hours.</p>	<ul style="list-style-type: none"> 1 week—Baseline is determined for each hour of last 1 week of data. Next 1 week of baselines are determined based on data of the last week. 1 month—Baseline is determined based on last 4 weeks of data. Baselines are learned for each hour of each day of week (7 x 24 baselines). Next 1 week of baselines are determined based on data of the last 4 weeks. For example, a baseline on Monday at 2:00 PM - 3:00 PM is learned using metric data from the last 4 Mondays at 2:00 PM - 3:00 PM.
Sensitivity	<p>The dynamic baseline provides a normal operating region of a given metric for a given scope. As seen in Figure 6 on page 22, the dynamic baseline is a tuple which has mean and std-dev applicable for a specific hour of the day.</p> <p>The sensitivity factor determines what is the allowable band of operation. Measurements outside of the band of operation cause an interval with exception. For example, if the baseline mean is 20 and std-dev is 2, then normal operating region is between 18 and 22. When sensitivity is <i>low</i> then normal operating region is treated as 10 (mean - 5*std-dev) and 30 (mean + 5*std-dev). In this case, if the measured average of a metric is between 10 and 30, then no alarm is raised. In contrast, if the average is 5 or 35, then an alarm is raised.</p>	<ul style="list-style-type: none"> Low—Any data point beyond 5 * std-dev from the baseline mean is outlier. Medium—Any data point beyond 3 * std-dev from baseline mean is outlier. High—Any data point beyond 2 * std-dev from baseline mean is outlier.

States for Alarm Mode

[Table 5 on page 16](#) shows all possible states for an alarm with the mode configured as alert.

Table 5: States for Alarm Mode Defined as Alert

State	Description
Learning	This is the initial state of each alarm. In this state, the alarm is processing real-time data and alarm stays in this state until sufficient data has been processed to make the decision about if an alarm should be generated or not. The duration of the learning period depends on the sliding window parameters.
Active	The condition specified by an alarm is met. Alarm will stay in this state as long as alarm conditions are satisfied.
Inactive	Condition specified by an alarm is not met. For example, after the learning state, the alarm transitions from active to inactive state because CPU usage was below the set threshold.
Disabled	Agent is not actively analyzing data for this alarm. The alarm is either deleted or temporarily disabled by the user.

Table 6 on page 16 shows all possible states for an alarm with the mode configured as event.

Table 6: States for Alarm Mode Defined as Event

State	Description
Enabled	This is the initial state of the alarm with the mode set to Event when a rule is configured. It stays in this state until conditions are met to generate an alarm.
Triggered	When conditions for alarm generation are satisfied, then an alarm is generated with a state of <i>triggered</i> . Alarm generation is logged at the end of each measurement interval as long conditions for alarms continue to be met.
Disabled	Agent is not actively analyzing data for this alarm. The alarm is either deleted or has been temporarily disabled by the user.

Aggregation Functions for Alarm Processing

Table 7 on page 17 lists and describes the aggregation functions for alarm processing.

Table 7: Aggregation Functions for Alarm Processing

Aggregation Function	Description
Average	<p>Statistical average of all data samples received within one measurement interval.</p> <p>Example: Generate Host Alert when Cpu-Usage Average during a 60 seconds interval is Above 80% of 2 of the last 3 measurement intervals.</p> <p>In this example, the measurement interval is 60 seconds. An alarm is generated if the average of the CPU usage samples exceeds 80% in any 2 measurement intervals out of 3 adjacent measurement intervals.</p>
Sum	<p>Sum of all data samples received within one measurement interval.</p> <p>Example: Generate Host Alert when Cpu-Usage Sum during a 60 seconds interval is Above 250% of 2 of the last 3 measurement intervals.</p> <p>In this example, An alarm is generated if the CPU usage sum is above 250% in any 2 measurement intervals out of 3 adjacent measurement intervals, where each measurement interval is 60 seconds in duration.</p>
Max	<p>Maximum sample value observed within one measurement interval.</p> <p>Example: Generate Host Alert when Cpu-Usage Max during a 60 seconds interval is Above 95% of 2 of the last 3 measurement intervals.</p> <p>In this example, the alarm is generated if the maximum CPU usage is above 95% in any 2 measurement intervals out of 3 adjacent measurement intervals, where each measurement interval is 60 seconds in duration.</p>
Min	<p>Minimum sample value observed within one measurement interval.</p> <p>Example: Generate Host Alert when Cpu-Usage Min during a 60 seconds interval is Below 5% of 2 of the last 3 measurement intervals.</p> <p>In this example, the alarm is generated if the minimum CPU usage is below 5% in any 2 measurement intervals out of 3 adjacent measurement intervals, where each measurement interval is 60 seconds in duration.</p>

Table 7: Aggregation Functions for Alarm Processing (*continued*)

Aggregation Function	Description
Std-Dev	<p>Standard Deviation of the time-series data is determined based on the samples received until current measurement interval.</p> <p>Example: Generate Host Alert when Cpu-Usage std-dev during a 60 seconds interval is Above 2 sigma of 2 of the last 3 measurement intervals.</p> <p>In this example, the alarm is generated when the raw time series samples are above mean + 2*sigma in at least 2 measurement intervals out of the last 3 measurement intervals, where each measurement interval is a duration of 60 seconds.</p>

Comparison Functions for Alarm Processing

Figure 4 on page 20 and Figure 5 on page 21 show examples of the Comparison Function, showing both increases and decreases at a minimum rate.

Table 8 on page 18 shows different comparison functions supported for AppFormix alarms.

Table 8: Comparison Functions for Alarm Processing

Comparison Operator	Description
Above	<p>Determine if result of the aggregation function within a given measurement interval is <i>above</i> the threshold.</p> <p>NOTE: For dynamic threshold <i>above</i>, AppFormix compares whether the result of the aggregation function is outside of the normal operating region (mean +/- sigma*sensitivity).</p>
Below	<p>Determine if result of the aggregation function determined for a given measurement interval is <i>below</i> the threshold.</p> <p>NOTE: For dynamic threshold, <i>below</i> compares whether the result of aggregation function is within the normal operating region (mean +/- sigma*sensitivity).</p>
Equal	Determine if result of the aggregation function is <i>equal</i> to the threshold.

Table 8: Comparison Functions for Alarm Processing (*continued*)

Comparison Operator	Description
Increasing-at-a-minimum-rate-of	<p>This comparison function is useful when you are interested in tracking a sudden increase in the value of a given metric instead of its absolute value. For example, if ingress or egress network bandwidth starts increasing within short intervals then you might want to raise an alarm. Figure 4 on page 20 shows sudden increase in metric average between measurement interval i1 and i2. Similarly, sudden increase is observed in metric average between measurement intervals i4 to i5.</p> <p>Example: Generate Host Alert when the <code>host.network.ingress.bit_rate</code> average during a 60 seconds interval is increasing-at-a-minimum-rate-of 25% of 2 of the last 3 measurement intervals.</p> <p>In the example, if the mean ingress bit rate increases by at least 25% in 2 measurement intervals out of 3, then an alarm is raised.</p>
Decreasing-at-a-minimum-rate-of	<p>This comparison function is useful when you are interested in tracking sudden decrease in the value of a given metric instead of its absolute value. For example, egress network bandwidth starts decreasing within short intervals then you might want to raise an alarm to investigate the root cause. Figure 5 on page 21 shows sudden decrease in metric average between measurement interval i1 and i2. Similarly, sudden decrease is observed in metric average between measurement intervals i3 and i4.</p> <p>Example: Generate Host Alert when the <code>host.network.egress.bit_rate</code> average during a 60 seconds interval is decreasing-at-a-minimum-rate-of 25% of 2 of the last 3 measurement intervals.</p> <p>In the example, if the mean egress bit rate decreases by at least 25% in 2 measurement intervals out of 3, then an alarm is raised.</p>

Figure 4: Comparison Function Showing Increasing-at-a-minimum-rate-of

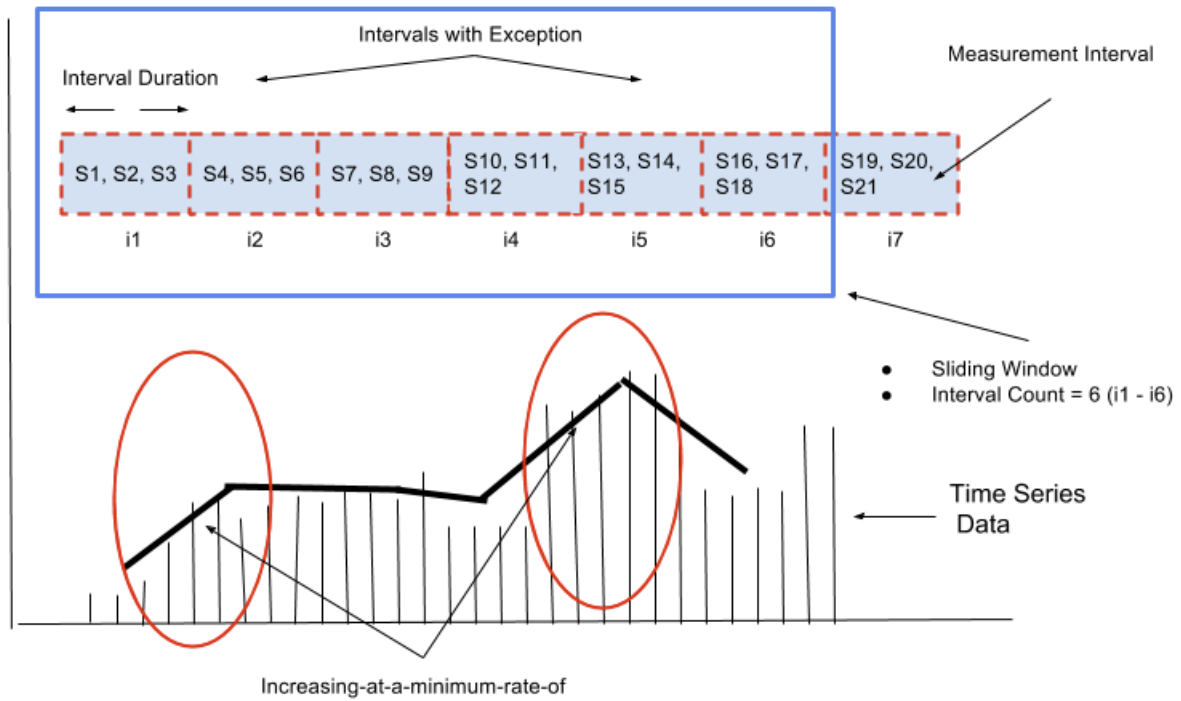
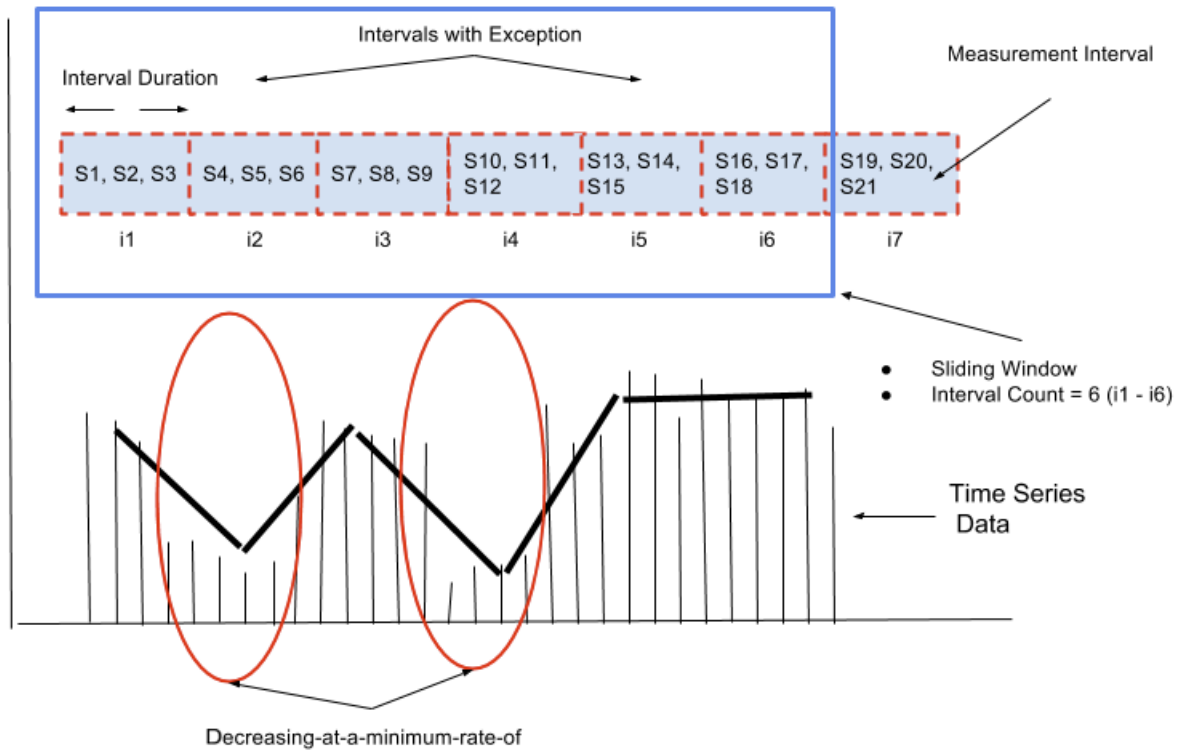


Figure 5: Comparison Function Showing Decreasing-at-a-minimum-rate-of



Dynamic Baseline Examples

Figure 6 on page 22 shows the dynamic baseline computed by 24 hours of data and the k-means clustering algorithm. For a given hour of the day, the blue dot is the **mean**; the green bar is the **mean + std-dev**; the purple bar is **mean - std-dev**.

Figure 6: Dynamic Baseline Determined by Last 24 Hours of Data and K-Means Clustering Algorithm

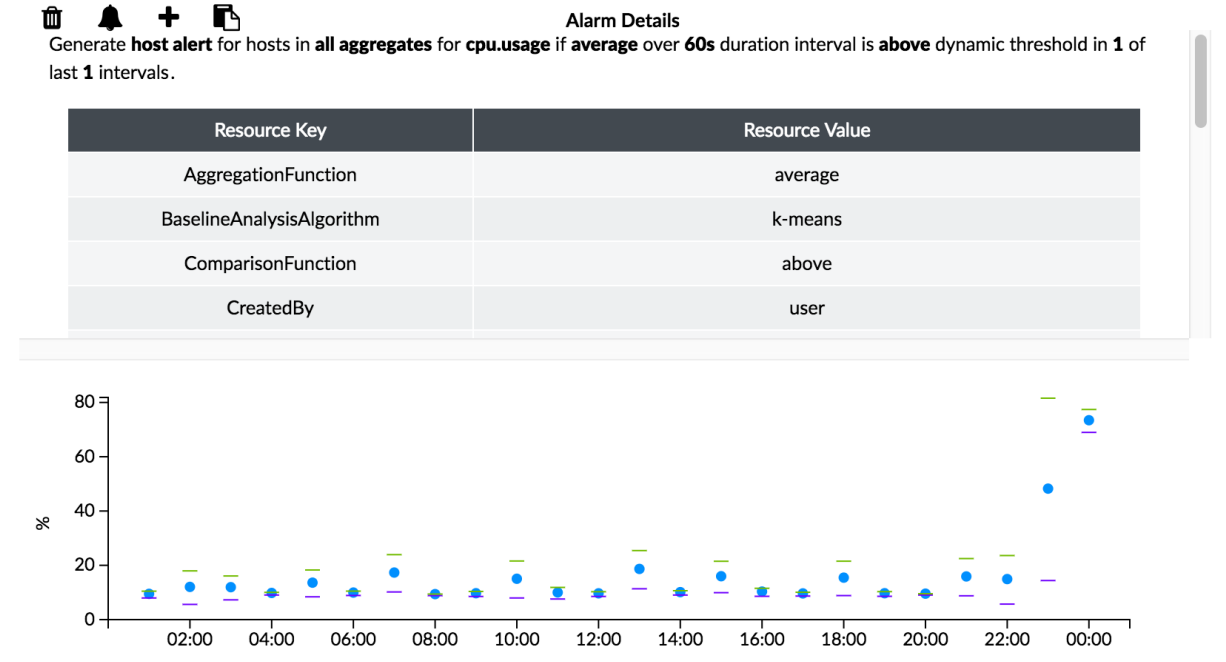
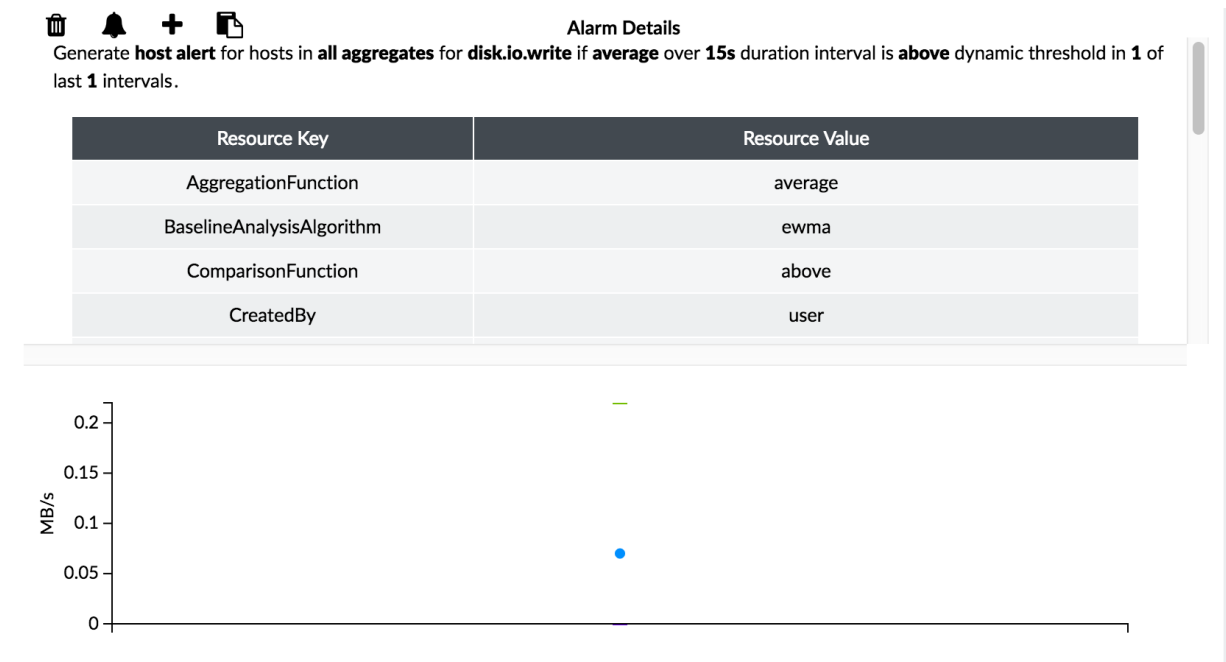


Figure 7 on page 22 shows the dynamic baseline computed by 24 hours of historical data using the EWMA algorithm. This baseline is used for the next 1 hour for alarm generation until it is updated again using the most recent 24 hours of data.

Figure 7: Dynamic Baseline Determined by Last 24 Hours of Historical Data Using EWMA

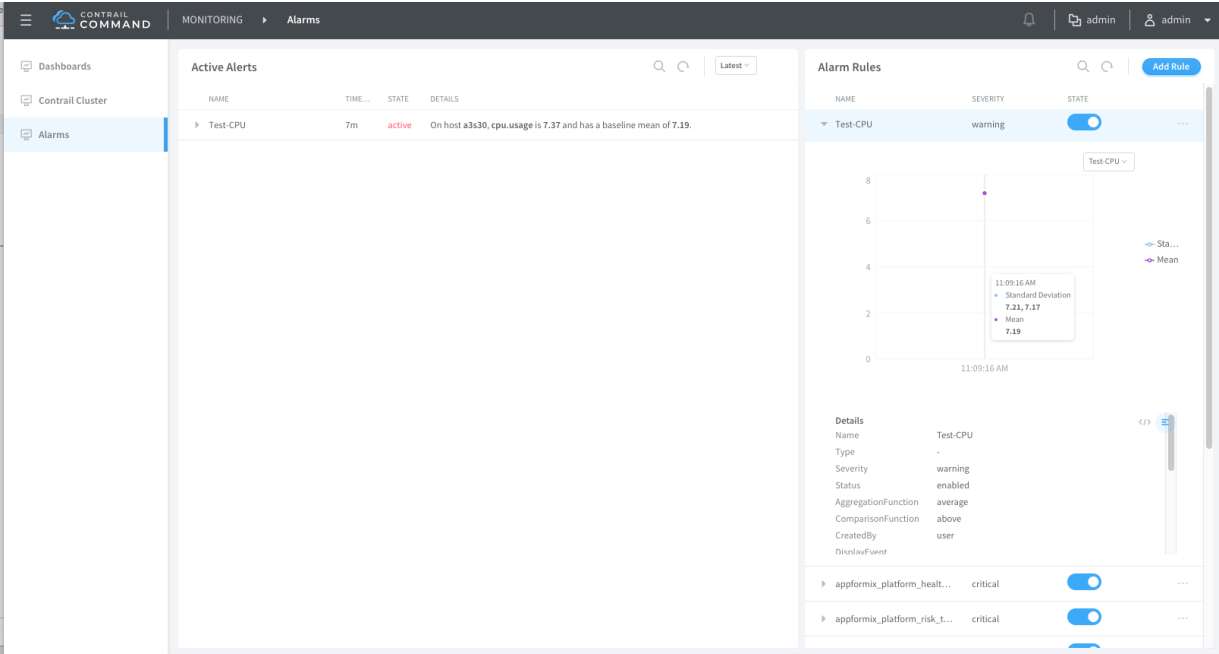


Configuring an Alarm Rule

To configure an alarm:

1. Select **Monitoring > Alarms**.
2. In the Alarm Rules panel, click **Add Rule** to create a new rule to trigger an alarm when a user-defined condition is met on one of the selected entities in the network.

Figure 8: Alarm Active Alerts and Alarm Rules Panel in Contrail Command



3. For Module, select one of the following options. Based on your selection, the fields differ.
Alarms—When Alarms is selected, you can configure alarms for entities such as hosts, instances, and network devices.
Service Alarms—When Service Alarms is selected, then you are able to configure alarms for services in your environment, such as RabbitMQ, MySQL, ScaleIO, and OpenStack services.

Figure 9: Create and Configure an Alarm in Contrail Command

The screenshot shows the 'Create Alarm' page in the Contrail Command web interface. The breadcrumb navigation at the top indicates the path: MONITORING > Alarms > Create Alarm. The left sidebar contains links to Dashboards, Contrail Cluster, and Alarms. The main form area is titled 'Module' with radio buttons for 'Alarms' (selected) and 'Service Alarms'. Below this is the 'Alarm Rule Type' section with radio buttons for 'Static' (selected) and 'Dynamic'. The form fields include:

- Name***: A text input field.
- Scope**: A dropdown menu.
- Generate**: A dropdown menu.
- For Metric**: A dropdown menu.
- When**: A dropdown menu.
- Interval (seconds)**: A text input field with the value '60'.
- Is**: A dropdown menu.
- Threshold (I)**: A text input field.
- Severity**: A dropdown menu.
- Advanced**: An unchecked checkbox.

 At the bottom of the form are two buttons: 'Create' (in blue) and 'Cancel' (in light blue).

4. Select Alarm Rule Type.

- **Static**—When an alarm is defined as static, the rule definition should include a predefined static threshold determined by the user.
- **Dynamic**—When alarm is defined as dynamic, the threshold is dynamically determined by the baseline algorithm, which can be either k-means or ewma.

5. Select the metric for the rule and specify interval when the rule should *trigger* an alarm. For other parameters, see [Table 3 on page 9](#) and descriptions in section "Alarm Definition."

6. Click **Create** to save the alarm.

RELATED DOCUMENTATION

[Configuring Instances in AppFormix | 25](#)

[Viewing Cluster Node Details and Metric Values | 31](#)

[Metrics Collected by AppFormix | 35](#)

Configuring Instances in AppFormix

This section describes the Instances detail screen and how to configure instances for virtual or physical servers using Contrail Command.

NOTE: In order to view and configure instances, your AppFormix license subscription must be active.

Instance Details Overview

Table 9 on page 25 provides descriptions for the instances column headers.

Table 9: Instance Details Headers and Columns

Header	Description
Status	The lights indicate the provisioning status of an instance and has multiple states: Red indicates an alert state, green indicates a normal state, yellow indicates warning state and grey for other states. Spinning circle means “in progress” and solid dot means “static.” If status information is missing (no-data), this field is empty.
Name	Shows the name of each instance.
State	Shows the current state of the instance. Power On: Active means the instance is running.
Server Type	Indicates which server type is in use, such as Baremetal Server. No LCM (lifecycle management) means that Contrail Command is not managing the server.
Networks	Displays the virtual network (VLAN) associated with the instance.
IP Addresses	Shows the IP address of the server.
Console	Indicates if a console port is available for the server.

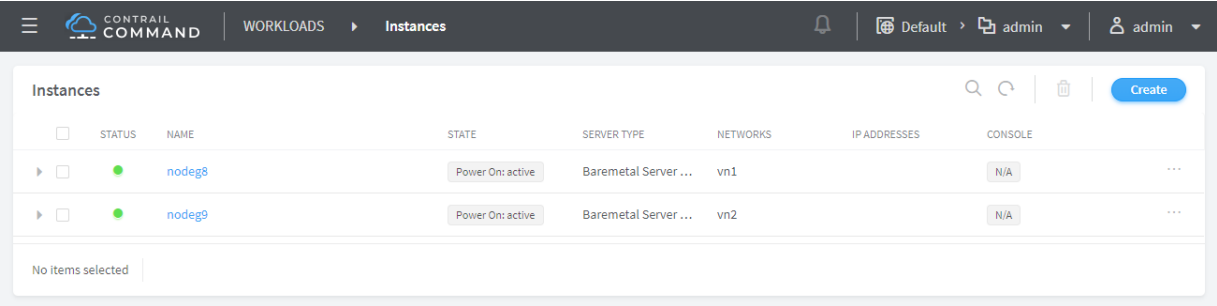
Creating Instances

A virtual network in the Contrail environment allows hosts in the same network to communicate with each other. This is similar to assigning a VLAN to each host so that hosts on the same VLAN can reach each other. An instance then matches the virtual network to devices and their interfaces, as shown in [Figure 10 on page 26](#).

To configure an instance to map a virtual network to devices and interfaces:

1. Select **Infrastructure > Workloads > Instances**. All virtual machine instances and baremetal server instances created appear on the Instances screen.

Figure 10: Workloads > Instances



2. Click **Create**, as shown in [Figure 10 on page 26](#), to add a new instance.

NOTE: (Optional) Click the ellipsis (...) to edit or remove an instance.

3. Select Server Type, which is either physical or virtual.

- a. When either Virtual Machine or New Baremetal Server are selected, complete the following fields, described in [Table 10 on page 27](#), to define an instance for the selected server:

Figure 11: Create an Instance for a Virtual Machine or New Baremetal Server

The screenshot shows the 'Create Instance' form in the Contrail Command interface. The form is divided into several sections:

- Server Type:** Three radio buttons are present: 'Virtual Machine' (selected), 'New Baremetal Server', and 'Existing Baremetal Server'.
- Instance Name:** A text input field with an asterisk indicating it is required.
- Select Boot Source:** A dropdown menu with 'Image' selected.
- Select Image:** A dropdown menu with 'Image' selected.
- Select Flavor:** A dropdown menu with 'Flavor' selected.
- Available Networks:** A section with a search bar, an 'Add all' button, and a list of networks. The networks listed are 'vn1', 'vn2', and two internal network IDs.
- Allocated Networks:** A section with a search bar, a 'Remove all' button, and a message 'Allocate at least one network'.
- Select SSH Key:** A dropdown menu.
- Availability Zone:** A dropdown menu.
- Count (1-10):** A dropdown menu with '1' selected.

At the bottom of the form are two buttons: 'Create' and 'Cancel'.

Table 10: Create Instance Fields—Virtual Machine or New Baremetal Server

Field	Description
Instance Name	Enter a name for this instance you are creating.
Select Boot Source	Select an image or clone from the list as your boot source.
Select Image	Select the software image from the list.
Select Flavor	Select the default configurations for virtual machines.
Available Networks	Network resources that are currently available.

Table 10: Create Instance Fields—Virtual Machine or New Baremetal Server *(continued)*

Field	Description
Allocated Networks	Network resources that can be allocated according to the demands of workloads.
Select SSH Key	Select an SSH key credential.
Availability Zone	Select an availability zone. An availability zone groups network nodes that run services like DHCP, L3, FW, and others. This allows you to associate an availability zone with their resources so that the resources get high availability.

Table 10: Create Instance Fields—Virtual Machine or New Baremetal Server *(continued)*

Field	Description
Count	Select a number from 1 - 10, which represents the number of instances to launch.

- b. When Existing Baremetal Server is selected, complete the following fields, described in Table 11 on page 30, to define an instance for the selected server:

Figure 12: Create an Instance for an Existing Baremetal Server

Contrail Command

WORKLOADS

Instances

Create Instance

Default

admin

admin

Instances

Flavors

Images

SSH Keys

Server Type

☐ Virtual Machine

☐ New Baremetal Server

☒ Existing Baremetal Server

Create Existing Baremetal Server

Instance Name*

Baremetal Node*

Associate interfaces

Interface

IP Address

VLAN ID*

Virtual Network*

Select Security Groups

+ Add

+ Add

Create

Cancel

Table 11: Create Instance Fields—Existing Baremetal Server

Field	Description
Create Existing Baremetal Server	
Instance Name	Enter a name for instance you are creating.
Baremetal Node	Select the name of the server.
Associate interfaces	
Interface	Name of the physical interface and MAC address for the server.
IP Address	IP address of the server's physical interface.

Table 11: Create Instance Fields—Existing Baremetal Server *(continued)*

Field	Description
VLAN ID	Identifier for the VLAN.
Virtual Network	Name of the virtual network to be mapped to this instance.
Select Security Groups	Defines which devices are in a security group.

4. Click **Create** to finish creating the instance.
5. To add other instances, click **Create**, as shown in figure [Figure 10 on page 26](#).

RELATED DOCUMENTATION

[Configuring AppFormix Alarms using Contrail Command | 3](#)

[Viewing Cluster Node Details and Metric Values | 31](#)

[Metrics Collected by AppFormix | 35](#)

Viewing Cluster Node Details and Metric Values

IN THIS SECTION

- [Time | 32](#)
- [Legend | 32](#)
- [Chart Data Values | 32](#)
- [Viewing Cluster Node Details and Host Charts | 33](#)

With Node Details and host charts, you can view real-time and historical values of all metrics that AppFormix monitors. Charts provide you with a way to view metrics for multiple entities across layers and organized by physical host, project, or aggregate. The charts update with the latest data streamed from the AppFormix Platform without needing to refresh. You can select which entities to display on the charts, and select the time period that is displayed. When you hover over the charts, a pop-up box shows the actual values for

the selected entities at a specific point in time. [Figure 14 on page 34](#) shows real-time metric values streamed from AppFormix.

NOTE: In order to view host charts, your AppFormix license subscription must be active.

Time

The Time in the Settings dialog box (see [Figure 14 on page 34](#)) provides navigation to a specific point in time that you want to view. Use the time and date drop-down list to select a range. Using the range selected, you can use the time slider to fine tune the time range by scaling up or down. This time range is used to query data that will be drawn in the visualizations.

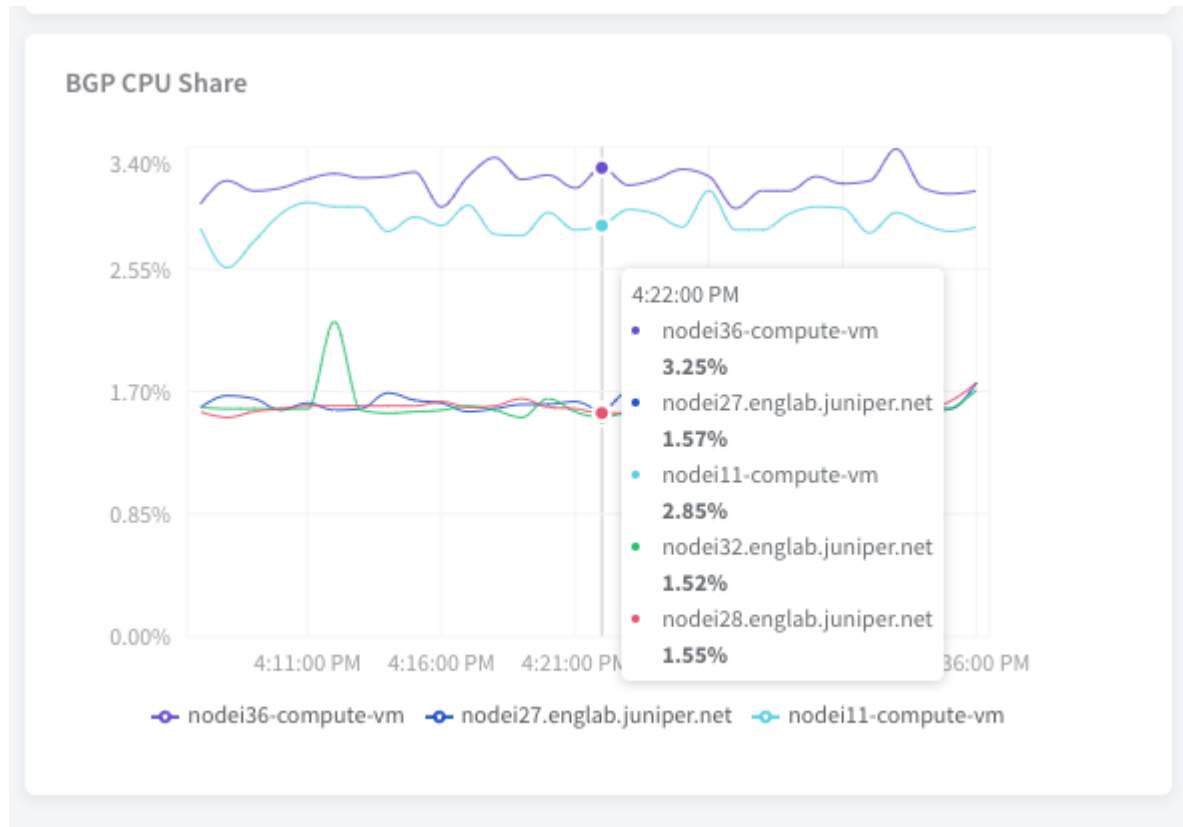
Legend

The Legend shows which entities are currently being displayed in the charts. See [Figure 14 on page 34](#). You can select a subset of entities to display to improve the clarity of the charts and focus on specific entities. The first five series are selected by default. The entity list is categorized and searchable in the Settings dialog box.

Chart Data Values

The host charts show the latest data for up to four different metrics, updating in real-time from a stream of data from the AppFormix Platform. When the cursor is positioned over the charts, a pop-up box shows the data values at that particular time. Charts can be zoomed in or out by opening the Settings dialog box and adjusting the time range. Four charts are displayed on the Dashboard at all times.

Figure 13: Chart Data Values Tool Tip for a Particular Time

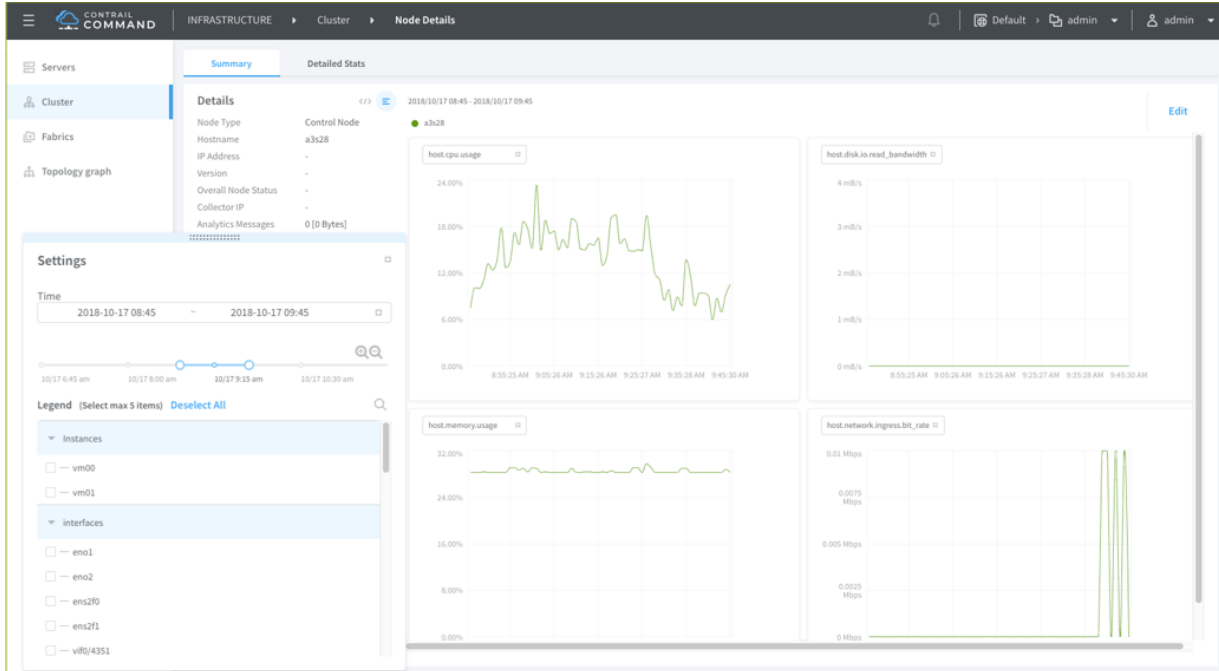


Viewing Cluster Node Details and Host Charts

To view cluster node details and charts:

1. Select **Infrastructure > Cluster > Cluster Nodes**.
2. Select the Control Nodes name to view node details.
3. In the Summary tab page, by default, basic information for the selected host is listed. Use the toggle button to switch between the textual information list and a JSON view for further technical processing.

Figure 14: Real-Time Metric Values Streamed from the AppFormix Platform

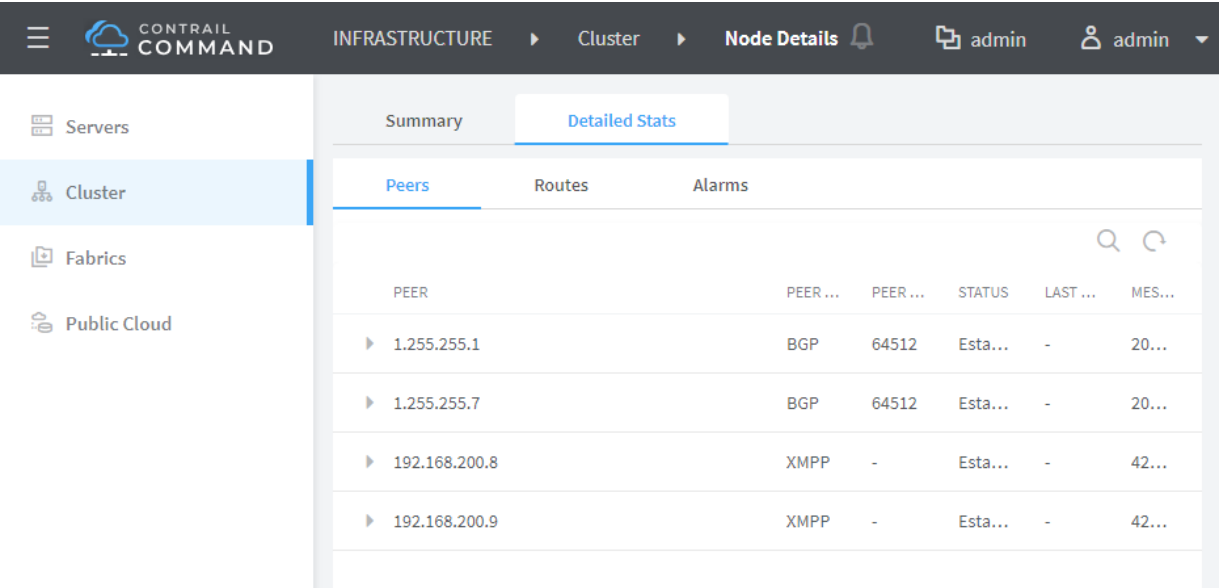


In the Settings dialog box, set the time range and legend. With an active AppFormix subscription, you have access to the Appformix data source and a separate 4-charts dashboard will show on the right to provide visualizations of data metrics.

- **Time**—For time range selection, use the time/date drop-down list to select a range. Using the range selected from the drop-down, use the time slider to fine tune (narrowing down or scaling up) the time range at a fixed step. This time range is used to query data that will be drawn in the visualizations.
- **Legend**—For legend or list of series, use the expanded list to add or remove entities drawn in a chart.

4. Select the Detailed Stats tab to view the following:

Figure 15: Node Detailed Stats for Peers, Routes, and Alarms



- Peers—Includes JSON values, peer type, peer ASN, status, and messages count.
- Routes—Includes routing table JSON values, prefix, protocol, next hop, label, security group, and origin virtual-network object (VN).
- Alarms—Includes severity, time, alarm type, and source.

RELATED DOCUMENTATION

- [Configuring Instances in AppFormix | 25](#)
- [Metrics Collected by AppFormix | 35](#)

Metrics Collected by AppFormix

IN THIS SECTION

- [Host CPU Data Metrics | 36](#)
- [Host Disk Metrics | 37](#)
- [Host Memory Usage | 38](#)
- [Host Mount Metrics | 39](#)
- [Host Network Data | 39](#)

- [Instances | 40](#)
- [Network Device | 42](#)
- [Contrail Release 5.0 vRouter Plug-In | 45](#)
- [OpenContrail vRouter on a Host | 46](#)
- [OpenStack Project in Chart View | 47](#)
- [RabbitMQ Service | 48](#)
- [ScaleIO Service | 51](#)
- [gRPC Sensors | 54](#)

A *metric* is a measured value for an element in the infrastructure. AppFormix Agent collects and calculates metrics for hosts and instances. AppFormix metrics are organized into hierarchical categories based on the type of metric.

Some metrics are a fraction of total capacity. In such cases, the category of the metric determines the total capacity by which the is computed. For instance, `host.cpu.usage` indicates the percentage of CPU consumed relative to the total CPU available on a host. In contrast, `instance.cpu.usage` is the percentage of CPU consumed relative to the total CPU available to an instance. As an example, consider an instance that is using 50% of one core on a host with 20 cores. The instance's `host.cpu.usage` will be 2.5%. If the instance has been allocated two cores, then its `instance.cpu.usage` will be 25%.

Alarms can be configured for any metric. Many metrics can also be displayed in charts. When an alarm triggers for a metric, the alarm is plotted on charts at the time of the event. In this way, metrics that cannot be plotted directly as a chart are still visually correlated in time with other metrics.

AppFormix Agent collects both raw metrics and calculated metrics. Raw metrics are values read directly from the underlying infrastructure. Calculated metrics are metrics that AppFormix Agent derives from raw metrics.

Host CPU Data Metrics

[Table 12 on page 36](#) lists the calculated metrics available for the host CPU data.

Table 12: Host CPU Data Metrics

Metric	Unit	Chart	Alarm
<code>host.cpu.usage</code>	%	x	x
<code>host.cpu.io_wait</code>	%	x	x

Table 12: Host CPU Data Metrics (*continued*)

Metric	Unit	Chart	Alarm
host.cpu.per_core.usage	%	—	x
host.cpu.per_core.user.usage	%	—	x
host.cpu.temperature	degree	—	x
host.cpu.normalized_load_1m	loadavg	x	x
host.cpu.normalized_load_5m	loadavg	x	x
host.cpu.normalized_load_15m	loadavg	x	x
host.cpu.cores.state_transition	0 or 1	—	x
host.disk.smart.predict_failure	0 or 1	—	x
host.heartbeat	0 or 1	—	x

host.cpu.normalized_load—Normalized load is calculated as a ratio of the number of running and ready-to-run threads to the number of CPU cores. This family of metrics indicate the level of demand for CPU. If the value exceeds 1, then more threads are ready to run than exists CPU cores to perform the execution. Normalized load is a provided as an average over 1-minute, 5-minute, and 15-minute intervals.

host.cpu.temperature—CPU temperature is derived from multiple temperature sensors in the processor(s) and chassis. This temperature provides a general indicator of temperature in degrees Celsius inside a physical host.

host.disk.smart.predict_failure—AppFormix Agent calculates *predict_failure* using multiple S.M.A.R.T. counters provided by disk hardware. The agent will set *predict_failure* to true (value=1) when it determines from a combination of S.M.A.R.T. counters that a disk is likely to fail. An alarm triggered for this metric contains the disk identifier in the metadata.

host.heartbeat—The *host.heartbeat* indicates if AppFormix Agent is functioning on a host. AppFormix Controller periodically checks the status of each host by making a status request to AppFormix Agent. The *host.heartbeat* metric is incremented for each successful response. Alarms can be configured to detect missed heartbeats over a given interval.

Host Disk Metrics

Table 13 on page 38 lists the raw metrics available for host disk.

Table 13: Host Disk Metrics

Metric	Unit	Chart	Alarm
host.disk.io.read	MBps	x	x
host.disk.io.write	MBps	x	x
host.disk.response_time	ms	x	x
host.disk.read_response_time	ms	x	x
host.disk.write_response_time	ms	x	x
host.disk.smart.hdd.command_timeout	count	—	x
host.disk.smart.hdd.current_pending_sector_count	count	—	x
host.disk.smart.hdd.offline_uncorrectable	count	—	x
host.disk.smart.hdd.reallocated_sector_count	count	—	x
host.disk.smart.hdd.reported_uncorrectable_errors	count	—	x
host.disk.smart.ssd.available_reserved_space	count	—	x
host.disk.smart.ssd.media_wearout_indicator	count	—	x
host.disk.smart.ssd.reallocated_sector_count	count	—	x
host.disk.smart.ssd.wear_leveling_count	count	—	x
host.disk.usage.bytes	GB	x	x
host.disk.usage.percent	%	x	x

Host Memory Usage

Table 14 on page 38 lists the raw metrics available for host memory usage.

Table 14: Metrics for Host Memory Usage

Metric	Unit	Chart	Alarm
host.memory.usage	%	x	x

Table 14: Metrics for Host Memory Usage (*continued*)

Metric	Unit	Chart	Alarm
host.memory.dirty.rate	dirty pages/s	x	x
host.memory.page_in_out.rate	dirty pages/s	x	x
host.memory.page_fault.rate	dirty pages/s	x	x
host.memory.swap.usage	dirty pages/s	x	x

Host Mount Metrics

[Table 15 on page 39](#) lists the raw metrics available for host mount.

Table 15: Host Mount Metrics

Metric	Unit	Chart	Alarm
host.mount.usage	%	x	x
host.mount.io.read	MBps	x	x
host.mount.io.write	MBps	x	x
host.mount.detect_change	1 or 0	—	x
host.mount.usage.bytes	GB	x	—

Host Network Data

[Table 16 on page 39](#) lists the raw metrics available for host network data.

Table 16: Host Network Data Metrics

Metric	Unit	Chart	Alarm
host.network.ingress.bit_rate	Mbps	x	x
host.network.egress.bit_rate	Mbps	x	x
host.network.ingress.packet_rate	packets/s	x	x
host.network.egress.packet_rate	packets/s	x	x

Table 16: Host Network Data Metrics (*continued*)

Metric	Unit	Chart	Alarm
host.network.ingress.errors	errors/s	x	x
host.network.egress.errors	errors/s	x	x
host.network.ingress.drops	drops/s	x	x
host.network.egress.drops	drops/s	x	x
host.network.ipv4tables.rule_count	count	x	x
host.network.ipv6tables.rule_count	count	x	x
openstack.host.disk_gb.allocated.count	count	x	x
openstack.host.disk_gb.allocated.percentage	percentage	—	x
openstack.host.memory_mb.allocated.count	count	x	x
openstack.host.memory_mb.allocated.percentage	percentage	—	x
openstack.host.vcpus_allocated.count	count	x	x
openstack.host.vcpus_allocated.percentage	percentage	—	x

Instances

[Table 17 on page 40](#) lists the raw metrics available for instances.

Table 17: Raw Metrics for Instances

Metric	Chart	Alarm
instance.cpu.usage	x	x
instance.disk.io.read_bandwidth	x	x
instance.disk.io.read_iops	x	x
instance.disk.io.read_iosize	x	x
instance.disk.io.read_response_time	x	x

Table 17: Raw Metrics for Instances (*continued*)

Metric	Chart	Alarm
instance.disk.io.write_bandwidth	x	x
instance.disk.io.write_iops	x	x
instance.disk.io.write_iosize	x	x
instance.disk.io.write_response_time	x	x
instance.disk.usage.bytes	x	x
instance.disk.usage.percentage	x	x
instance.memory.usage	x	x
instance.network.egress.bit_rate	x	x
instance.network.egress.drops	x	x
instance.network.egress.errors	x	x
instance.network.egress.packet_rate	x	x
instance.network.ingress.bit_rate	x	x
instance.network.ingress.drops	x	x
instance.network.ingress.errors	x	x
instance.network.ingress.packet_rate	x	x

[Table 18 on page 41](#) lists the calculated metric available for instances.

Table 18: Calculated Metrics for Instances

Metric	Chart	Alarm
instance.heartbeat	—	x

instance.heartbeat—The *instance.heartbeat* indicates whether an instance is running. AppFormix Agent periodically checks the state of host processes associated with each instance. The **instance.heartbeat**

metric is incremented for each successful status check. Alarms may be configured to detect missed heartbeats over a given interval.

Network Device

AppFormix can collect network device metrics using SNMP or Juniper Telemetry Interface (JTI). See *Network Devices* for configuration and monitoring information.

[Table 19 on page 42](#) lists the metrics available per interface with SNMP network device monitoring. See *Network Device Monitoring with SNMP*.

Table 19: Metrics Available per Interface with SNMP Network Device Monitoring

Metric	Unit	Chart	Alarm
snmp.interface.out_discards	discards/s	x	x
snmp.interface.in_discards	discards/s	x	x
snmp.interface.in_errors	errors/s	x	x
snmp.interface.out_unicast_packets	packets/s	x	x
snmp.interface.in_octets	octets/s	x	x
snmp.interface.in_unicast_packets	packets/s	x	x
snmp.interface.out_packet_queue_length	count	x	x
snmp.interface.speed	bits/s	x	x
snmp.interface.out_octets	octets/s	x	x
snmp.interface.in_unknown_protocol	packets/s	x	x
snmp.interface.in_non_unicast_packets	packets/s	x	x
snmp.interface.out_errors	errors/s	x	x
snmp.interface.out_non_unicast_packets	packets/s	x	x

[Table 20 on page 43](#) lists the metrics available per interface with JTI network device monitoring. See also *Network Device Monitoring with JTI and Custom Sensors for JTI and gRPC*.

Table 20: Metrics Available per Interface with JTI Network Device Monitoring

Metric	Unit	Chart	Alarm
junos.system.linecard.interface.egress_errors.if_errors	errors/s	x	x
junos.system.linecard.interface.egress_errors.if_discard	discards/s	x	x
junos.system.linecard.interface.egress_stats.if_1sec_pkts	packets/s	x	x
junos.system.linecard.interface.egress_stats.if_octets	octets/s	x	x
junos.system.linecard.interface.egress_stats.if_mc_pkts	packets/s	x	x
junos.system.linecard.interface.egress_stats.if_bc_pkts	packets/s	x	x
junos.system.linecard.interface.egress_stats.if_1sec_octets	octets/s	x	x
junos.system.linecard.interface.egress_stats.if_pkts	packets/s	x	x
junos.system.linecard.interface.egress_stats.if_uc_pkts	packets/s	x	x
junos.system.linecard.interface.egress_stats.if_pause_pkts	packets/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_fifo_errors	errors/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_frame_errors	errors/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_l3_incompletes	packets/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_runts	packets/s	x	x
junos.system.linecard.interface.ingress_errors.if_errors	errors/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_l2chan_errors	errors/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_resource_errors	errors/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_qdrops	drops/s	x	x
junos.system.linecard.interface.ingress_errors.if_in_l2_mismatch_timeouts	packets/s	x	x
junos.system.linecard.interface.ingress_stats.if_1sec_pkts	packets/s	x	x
junos.system.linecard.interface.ingress_stats.if_octets	octets/s	x	x

Table 20: Metrics Available per Interface with JTI Network Device Monitoring (*continued*)

Metric	Unit	Chart	Alarm
junos.system.linecard.interface.ingress_stats.if_mc_pkts	packets/s	x	x
junos.system.linecard.interface.ingress_stats.if_bc_pkts	packets/s	x	x
junos.system.linecard.interface.ingress_stats.if_1sec_octets	octets/s	x	x
junos.system.linecard.interface.ingress_stats.if_error	errors/s	x	x
junos.system.linecard.interface.ingress_stats.if_pkts	packets/s	x	x
junos.system.linecard.interface.ingress_stats.if_uc_pkts	packets/s	x	x
junos.system.linecard.interface.ingress_stats.if_pause_pkts	packets/s	x	x

Table 21 on page 44 lists the metrics available per interface queue with JTI network device monitoring.

Table 21: Metrics Available per Interface Queue with JTI Network Device Monitoring

Metric	Unit	Chart	Alarm
junos.system.linecard.interface.egress_queue_info.bytes	bytes/s	x	x
junos.system.linecard.interface.egress_queue_info.packets	packets/s	x	x
junos.system.linecard.interface.egress_queue_info.allocated_buffer_size	bytes	x	x
junos.system.linecard.interface.egress_queue_info.avg_buffer_occupancy	bytes	x	x
junos.system.linecard.interface.egress_queue_info.cur_buffer_occupancy	bytes	x	x
junos.system.linecard.interface.egress_queue_info.peak_buffer_occupancy	bytes	x	x
junos.system.linecard.interface.egress_queue_info.red_drop_bytes	bytes/s	x	x
junos.system.linecard.interface.egress_queue_info.red_drop_packets	packets/s	x	x
junos.system.linecard.interface.egress_queue_info.rl_drop_bytes	bytes/s	x	x
junos.system.linecard.interface.egress_queue_info.rl_drop_packets	packets/s	x	x
junos.system.linecard.interface.egress_queue_info.tail_drop_packets	packets/s	x	x

Contrail Release 5.0 vRouter Plug-In

Table 22 on page 45 lists metrics published by the Contrail Release 5.0 vRouter plug-in. See *Contrail Monitoring* to view and configure Contrail services from the GUI. See *Service Monitoring* to configure Contrail monitoring using Ansible.

Table 22: Metrics for Contrail Release 5.0 vRouter Plug-In

Metric	Unit	Chart	Alarm
plugin.contrail.vrouter.v5. aged_flows	count	x	x
plugin.contrail.vrouter.v5. total_flows	count	x	x
plugin.contrail.vrouter.v5. exception_packets	count	x	x
plugin.contrail.vrouter.v5. drop_stats_flow_queue_limit_exceeded	count	x	x
plugin.contrail.vrouter.v5. drop_stats_flow_table_full	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vlan_fwd_enq	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vlan_fwd_tx	count	x	x
plugin.contrail.vrouter.v5. flow_export_drops	count	x	x
plugin.contrail.vrouter.v5. flow_export_sampling_drops	count	x	x
plugin.contrail.vrouter.v5. flow_rate_active_flows	count	x	x
plugin.contrail.vrouter.v5. flow_rate_deleted_flows	count	x	x
plugin.contrail.vrouter.v5. flow_rate_added_flows	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_discard	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_pull	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_flow_no_memory	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_flow_invalid_protocol	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_flow_action_drop	count	x	x

Table 22: Metrics for Contrail Release 5.0 vRouter Plug-In (*continued*)

plugin.contrail.vrouter.v5. drop_stats_vhost_ds_interface_drop	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_duplicated	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_push	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_invalid_nh	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_invalid_protocol	count	x	x
plugin.contrail.vrouter.v5. drop_stats_vhost_ds_drop_pkts	count	x	x

OpenContrail vRouter on a Host

Table 23 on page 46 lists raw metrics available for an OpenContrail vRouter on a host. See *Service Monitoring* to configure Contrail monitoring using Ansible.

Table 23: Raw Metrics for OpenContrail vRouter

Metric	Chart	Alarm
plugin.contrail.vrouter.aged_flows	x	x
plugin.contrail.vrouter.total_flows	x	x
plugin.contrail.vrouter.exception_packets	x	x
plugin.contrail.vrouter.drop_stats_flow_queue_limit_exceeded	x	x
plugin.contrail.vrouter.drop_stats_flow_table_full	x	x
plugin.contrail.vrouter.drop_stats_vlan_fwd_enq	x	x
plugin.contrail.vrouter.drop_stats_vlan_fwd_tx	x	x
plugin.contrail.vrouter.flow_export_drops	x	x
plugin.contrail.vrouter.flow_export_sampling_drops	x	x
plugin.contrail.vrouter.flow_rate_active_flows	x	x
plugin.contrail.vrouter.flow_rate_added_flows	x	x

Table 23: Raw Metrics for OpenContrail vRouter (*continued*)

Metric	Chart	Alarm
plugin.contrail.vrouter.flow_rate_deleted_flows	x	x

OpenStack Project in Chart View

Table 24 on page 47 lists the raw metrics available in the OpenStack Project Chart View. See *AppFormix Role-Based Access* to grant AppFormix permissions to read-only OpenStack users.

Table 24: Raw Metrics for OpenStack Project

Metric	Chart	Alarm
openstack.project.active_instances.count	x	x
openstack.project.active_instances.percentage	—	x
openstack.project.floating_ip.allocated.count	x	x
openstack.project.floating_ip.allocated.percentage	—	x
openstack.project.ram.allocated.count	x	x
openstack.project.ram.allocated.percentage	—	x
openstack.project.security_group.allocated.count	x	x
openstack.project.security_group.allocated.percentage	—	x
openstack.project.total_disk_usage_gb_hours.count	—	x
openstack.project.total_hours.count	—	x
openstack.project.total_memory_usage_mb_hours.count	—	x
openstack.project.total_vcpu_usage_hours.count	—	x
openstack.project.vcpus.allocated.count	—	x
openstack.project.vcpus.allocated.percentage	—	x
openstack.project.virtual_network.allocated.count	x	x

Table 24: Raw Metrics for OpenStack Project (*continued*)

Metric	Chart	Alarm
openstack.project.virtual_network.allocated.percentage	—	x
openstack.project.volume.allocated.count	x	x
openstack.project.volume.allocated.percentage	—	x
openstack.project.volume_gb.allocated.count	x	x
openstack.project.volume_gb.allocated.percentage	—	x

RabbitMQ Service

Table 25 on page 48 lists the raw metrics available for RabbitMQ monitoring. See *RabbitMQ Monitoring* to view and configure RabbitMQ services from the GUI. See *Service Monitoring* to configure RabbitMQ monitoring using Ansible.

Table 25: Raw Metrics for RabbitMQ Monitoring

Metric	Unit	Chart	Alarm
rabbit.cluster.connection_totals.blocked_connections	count	x	x
rabbit.cluster.connection_totals.blocked_connections_details	messages/s	x	x
rabbit.cluster.message_stats.ack	count	x	x
rabbit.cluster.message_stats.ack_details	messages/s	x	x
rabbit.cluster.message_stats.deliver	count	x	x
rabbit.cluster.message_stats.deliver_details	messages/s	x	x
rabbit.cluster.message_stats.deliver_get	count	x	x
rabbit.cluster.message_stats.deliver_get_details	messages/s	x	x
rabbit.cluster.message_stats.get	count	x	x
rabbit.cluster.message_stats.get_details	messages/s	x	x
rabbit.cluster.message_stats.publish	count	x	x

Table 25: Raw Metrics for RabbitMQ Monitoring (continued)

Metric	Unit	Chart	Alarm
rabbit.cluster.message_stats.publish_details	messages/s	x	x
rabbit.cluster.message_stats.redeliver	count	x	x
rabbit.cluster.message_stats.redeliver_details	messages/s	x	x
rabbit.cluster.object_totals.channels	count	x	x
rabbit.cluster.object_totals.connections	count	x	x
rabbit.cluster.object_totals.consumers	count	x	x
rabbit.cluster.object_totals.exchanges	count	x	x
rabbit.cluster.object_totals.queues	count	x	x
rabbit.cluster.queue_totals.blocked_queues	count	x	x
rabbit.cluster.queue_totals.blocked_queues_details	messages/s	x	x
rabbit.cluster.queue_totals.consumer_utilisation_percent	count	x	x
rabbit.cluster.queue_totals.messages	count	x	x
rabbit.cluster.queue_totals.messages_details	messages/s	x	x
rabbit.cluster.queue_totals.messages_ready	count	x	x
rabbit.cluster.queue_totals.messages_ready_details	messages/s	x	x
rabbit.cluster.queue_totals.messages_unacknowledged	count	x	x
rabbit.cluster.queue_totals.messages_unacknowledged_details	messages/s	x	x
rabbit.queue.consumers	count	—	x
rabbit.queue.consumer_utilisation	count	—	x
rabbit.queue.messages	count	—	x
rabbit.queue.messages_ready	count	—	x

Table 25: Raw Metrics for RabbitMQ Monitoring (continued)

Metric	Unit	Chart	Alarm
rabbit.queue.messages_ready_detail	count	—	x
rabbit.queue.memory	count	—	x
rabbit.queue.messages_detail	count	—	x
rabbit.queue.messages_unacknowledged	count	—	x
rabbit.queue.messages_unacknowledged_detail	count	—	x
rabbit.queue.state	count	—	x
rabbit.node.sockets_total	count	x	x
rabbit.node.fd_total	count	x	x
rabbit.node.sockets_used_percent	count	x	x
rabbit.node.run_queue	count	x	x
rabbit.node.proc_used_percent	count	x	x
rabbit.node.proc_total	count	x	x
rabbit.node.mem_used_percent	count	x	x
rabbit.node.uptime	count	x	x
rabbit.node.disk_usage_ratio	count	x	x
rabbit.node.disk_free_alarm	count	x	x
rabbit.node.fd_used_percent	count	x	x
rabbit.node.mem_limit	count	x	x
rabbit.node.mem_alarm	count	x	x
rabbit.node.disk_free	count	x	x
rabbit.node.sockets_used	count	x	x

Table 25: Raw Metrics for RabbitMQ Monitoring (*continued*)

Metric	Unit	Chart	Alarm
rabbit.node.processors	count	x	x
rabbit.node.running	count	x	x
rabbit.node.disk_free_limit	count	x	x
rabbit.node.fd_used	count	x	x
rabbit.node.proc_used	count	x	x
rabbit.node.mem_used	count	x	x
rabbit.node.heartbeat	count	x	x
rabbit.node.latency	count	x	x

ScaleIO Service

[Table 26 on page 51](#) lists the raw metrics available for ScaleIO monitoring. See *ScaleIO Monitoring* for viewing and configuring ScaleIO services from the GUI. See *Service Monitoring* to configure ScaleIO monitoring using Ansible.

Table 26: Raw Metrics for ScaleIO Monitoring

Metric	Unit	Chart	Alarm
numOfDevices	count	x	x
numOfProtectionDomains	count	x	x
numOfSdc	count	x	x
numOfSds	count	x	x
numOfStoragePools	count	x	x
numOfVtrees	count	x	x
numOfSnapshots	count	x	x
numOfVolumes	count	x	x

Table 26: Raw Metrics for ScaleIO Monitoring (continued)

Metric	Unit	Chart	Alarm
numOfThickBaseVolumes	count	x	x
numOfThinBaseVolumes	count	x	x
numOfVolumesInDeletion	count	x	x
numOfMappedToAllVolumes	count	x	x
numOfUnmappedVolumes	count	x	x
capacityAvailableForVolumeAllocationInKb	Kbyte	x	x
capacityInUseInKb	Kbyte	x	x
capacityLimitInKb	Kbyte	x	x
unusedCapacityInKb	Kbyte	x	x
spareCapacityInKb	Kbyte	x	x
protectedCapacityInKb	Kbyte	x	x
maxCapacityInKb	Kbyte	x	x
snapCapacityInUseInKb	Kbyte	x	x
thickCapacityInUseInKb	Kbyte	x	x
thinCapacityInUseInKb	Kbyte	x	x
bckRebuildReadBandwidth	Kbyte/sec	x	x
bckRebuildWriteBandwidth	Kbyte/sec	x	x
fwdRebuildReadBandwidth	Kbyte/sec	x	x
fwdRebuildWriteBandwidth	Kbyte/sec	x	x
normRebuildReadBandwidth	Kbyte/sec	x	x
normRebuildWriteBandwidth	Kbyte/sec	x	x

Table 26: Raw Metrics for ScaleIO Monitoring (continued)

Metric	Unit	Chart	Alarm
primaryReadBandwidth	Kbyte/sec	x	x
primaryWriteBandwidth	Kbyte/sec	x	x
rebalanceReadBandwidth	Kbyte/sec	x	x
rebalanceWriteBandwidth	Kbyte/sec	x	x
secondaryReadBandwidth	Kbyte/sec	x	x
secondaryWriteBandwidth	Kbyte/sec	x	x
totalReadBandwidth	Kbyte/sec	x	x
totalWriteBandwidth	Kbyte/sec	x	x
bckRebuildReadIops	IOPS	x	x
bckRebuildWriteIops	IOPS	x	x
fwdRebuildReadIops	IOPS	x	x
fwdRebuildWriteIops	IOPS	x	x
normRebuildReadIops	IOPS	x	x
normRebuildWriteIops	IOPS	x	x
primaryReadIops	IOPS	x	x
primaryWriteIops	IOPS	x	x
rebalanceReadIops	IOPS	x	x
rebalanceWriteIops	IOPS	x	x
secondaryReadIops	IOPS	x	x
secondaryWriteIops	IOPS	x	x
totalReadIops	IOPS	x	x

Table 26: Raw Metrics for ScaleIO Monitoring (continued)

Metric	Unit	Chart	Alarm
totalWritelops	IOPS	x	x
bckRebuildReadlosize	Kbyte	x	x
bckRebuildWritelosize	Kbyte	x	x
fwdRebuildReadlosize	Kbyte	x	x
fwdRebuildWritelosize	Kbyte	x	x
normRebuildReadlosize	Kbyte	x	x
normRebuildWritelosize	Kbyte	x	x
primaryReadlosize	Kbyte	x	x
primaryWritelosize	Kbyte	x	x
rebalanceReadlosize	Kbyte	x	x
rebalanceWritelosize	Kbyte	x	x
secondaryReadlosize	Kbyte	x	x
secondaryWritelosize	Kbyte	x	x
totalReadlosize	Kbyte	x	x
totalWritelosize	Kbyte	x	x

gRPC Sensors

Table 27 on page 55 lists the available gRPC sensors. To enable these sensors, see *Custom Sensors for JTI and gRPC*.

NOTE: These sensors are applicable only for Juniper network devices.

Table 27: gRPC Sensors

Sensor	Chart	Alarm
/junos/services/label-switched-path/usage/	x	x
/components/	x	x
/junos/system/subscriber-management/infra/sdb/statistics/	x	x
/junos/task-memory-information/ task-memory-overall-report/task-memory-stats-list/ task-memory-stats/	x	x
/junos/task-memory-information/ task-memory-overall-report/task-size-block-list/ task-size-block/	x	x
/lldp/interfaces/interface/state/	x	x
/interfaces/	x	x
/bgp-rib/afi-safis/afi-safi/ipv4-unicast/loc-rib/	x	x
/bgp-rib/afi-safis/afi-safi/ipv6-unicast/loc-rib/	x	x
/bgp-rib/afi-safis/afi-safi/ipv4-unicast/neighbors/	x	x
/bgp-rib/afi-safis/afi-safi/ipv6-unicast/neighbors/	x	x
/junos/system/linecard/qmon/	x	x
/junos/system/linecard/optics/	x	x
/junos/system/linecard/packet/usage/	x	x
/junos/system/linecard/firewall/	x	x
/junos/rsvp-interface-information/	x	x
/junos/system/linecard/npu/memory	x	x
/junos/system/linecard/cpu/memory/	x	x

Table 27: gRPC Sensors (continued)

Sensor	Chart	Alarm
/lacp/	x	x
/network-instances/network-instance/protocols/ protocol/isis/levels/level/	x	x
/junos/services/segment-routing/interface/ingress/ usage/	x	x
/junos/services/segment-routing/interface/egress/usage/	x	x
/lldp/	x	x
/mpls/	x	x
/nd6-information/	x	x
/arp-information/	x	x
/junos/system/subscriber-management/infra/network/ ppp/	x	x
/network-instances/network-instance/protocols/ protocol/bgp/	x	x
/network-instances/network-instance/protocols/ protocol/isis/levels/level/	x	x
/junos/services/segment-routing/sid/usage/	x	x

RELATED DOCUMENTATION

[Configuring AppFormix Alarms using Contrail Command | 3](#)
[Configuring Instances in AppFormix | 25](#)
[Viewing Cluster Node Details and Metric Values | 31](#)

2

PART

Configuring Contrail

Optimizing Contrail | 58

Optimizing Contrail

IN THIS CHAPTER

- [vRouter Command Line Utilities | 58](#)

vRouter Command Line Utilities

IN THIS SECTION

- [Overview | 58](#)
- [vif Command | 59](#)
- [flow Command | 62](#)
- [vrfstats Command | 64](#)
- [rt Command | 66](#)
- [dropstats Command | 67](#)
- [mpls Command | 71](#)
- [mirror Command | 73](#)
- [vxlan Command | 76](#)
- [nh Command | 77](#)

Overview

This section describes the shell prompt utilities available for examining the state of the vrouter kernel module in Contrail.

The most useful commands for inspecting the Contrail vrouter module are summarized in the following table.

Command	Description
vif	Inspect vrouter interfaces associated with the vrouter module.
flow	Display active flows in a system.
vrfstats	Display next hop statistics for a particular VRF.
rt	Display routes in a VRF.
dropstats	Inspect packet drop counters in the vrouter.
mpls	Display the input label map programmed into the vrouter.
mirror	Display the mirror table entries.
vxlan	Display the vxlan table entries.
nh	Display the next hops that the vrouter knows.
--help	Display all command options available for the current command.

The following sections describe each of the vrouter utilities in detail.

vif Command

The vrouter requires vrouter interfaces (**vif**) to forward traffic. Use the **vif** command to see the interfaces that are known by the vrouter.

NOTE: Having interfaces only in the OS (Linux) is not sufficient for forwarding. The relevant interfaces must be added to vrouter. Typically, the set up of interfaces is handled by components like **nova-compute** or vrouter agent.

Example: vif --list

```
# vif --list
vif0/0  OS: pkt0
        Type:Agent HWaddr:00:00:5e:00:01:00 IPaddr:0
```

```

Vrf:65535 Flags:L3 MTU:1514 Ref:2
RX packets:6591 bytes:648577 errors:0
TX packets:12150 bytes:1974451 errors:0
vif0/1 OS: vhost0
Type:Host HWaddr:00:25:90:c3:08:68 IPaddr:0
Vrf:0 Flags:L3 MTU:1514 Ref:3
RX packets:3446598 bytes:4478599344 errors:0
TX packets:851770 bytes:1337017154 errors:0
vif0/2 OS: plp0p0 (Speed 1000, Duplex 1)
Type:Physical HWaddr:00:25:90:c3:08:68 IPaddr:0
Vrf:0 Flags:L3 MTU:1514 Ref:22
RX packets:1643238 bytes:1391655366 errors:2812
TX packets:3523278 bytes:6806058059 errors:0
vif0/18 OS: tap3214fc7e-88
Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0
Vrf:13 Flags:PL3L2 MTU:9160 Ref:6
RX packets:60 bytes:4873 errors:0
TX packets:21 bytes:2158 errors:0

```

Table 28: vif Fields

vif Output Field	Description
vif0/X	The vrouter assigned name, where 0 is the router id and X is the index allocated to the interface within the vrouter.
OS: pkt0	The pkt0 (in this case) is the name of the actual OS (Linux) visible interface name. For physical interfaces, the speed and the duplex settings are also displayed.
Type:xxxxx	<p>Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0</p> <p>The type of interface and its IP address, as defined by vrouter. The values can be different from what is seen in the OS. Types defined by vrouter include:</p> <ul style="list-style-type: none"> • Virtual – Interface of a virtual machine (VM). • Physical – Physical interface (NIC) in the system. • Host – An interface toward the host. • Agent – An interface used to trap packets to the vrouter agent when decisions need to be made for the forwarding path.

Table 28: vif Fields (*continued*)

vif Output Field	Description
Vrf:xxxxx	<p>Vrf:65535 Flags:L3 MTU:1514 Ref:2</p> <p>The identifier of the vrf to which the interface is assigned, the flags set on the interface, the MTU as understood by vrouter, and a reference count of how many individual entities actually hold reference to the interface (mainly of debugging value).</p> <p>Flag options identify that the following are enabled for the interface:</p> <ul style="list-style-type: none"> • P - Policy • L3 - Layer 3 forwarding • L2 - Layer 2 bridging • X - Cross connect mode, only set on physical and host interfaces, indicating that packets are moved between physical and host directly, with minimal intervention by vrouter. Typically set when the agent is not alive or not in good shape. • Mt - Mirroring transmit direction • Mr - Mirroring receive direction • Tc - Checksum offload on the transmit side. Valid only on the physical interface.
Rx	<p>RX packets:60 bytes:4873 errors:0</p> <p>Packets received by vrouter from this interface.</p>
Tx	<p>TX packets:21 bytes:2158 errors:0</p> <p>Packets transmitted out by vrouter on this interface.</p>

vif Options

Use **vif --help** to display all options available for the vif command. Following is a brief description of each option.

NOTE: It is not recommended to use the following options unless you are very experienced with the system utilities.

```
# vif --help
Usage: vif [--create <intf_name> --mac <mac>]
          [--add <intf_name> --mac <mac> --vrf <vrf>]
```

```

--type [vhost|agent|physical|virtual][--policy, --mode
<mode:x>]]
[--delete <intf_id>]
[--get <intf_id>][--kernel]
[--set <intf_id> --vlan <vlan_id> --vrf <vrf_id>]
[--list]
[--help]

```

Option	Description
--create	Creates a 'Host' interface with name <intf_name> and mac <mac> on the host kernel. The 'vhost0' interface that you see on Linux is a typical example of invocation of this command.
--add	Adds the existing interfaces in the host OS to vrouter, with type and flag options.
--delete	Deletes the interface from vrouter. The <intf_id> is the vrouter interface id as given by vif0/X , where X is the iID
--get	Displays a specific interface. The <intf_id> is the vrouter interface id, unless the command is appended by the '--kernel' option, in which case the ID can be the kernel ID.
--set	Set working parameters of an interface. The only ones supported are the vlan id and the vrf . The vlan id as understood by vrouter differs from what one typically expects, and is relevant as of now only for interfaces of service instances.
--list	Display all of the interfaces of which the vrouter is aware.
--help	Display all options available for the current command.

flow Command

Use the **flow** command to display all active flows in a system.

Example: flow -l

Use `-l` to list everything in the flow table. The `-l` is the only relevant debugging option.

```
# flow -l
Flow table
  Index      Source:Port      Destination:Port  Proto(V)
-----
263484      1.1.1.1.252:1203      1.1.1.1.253:0      1 (3)
              (Action:F, S(nh):91, Statistics:22/1848)
379480      1.1.1.1.253:1203      1.1.1.1.252:0      1 (3)
              (Action:F, S(nh):75, Statistics:22/1848)
```

Each record in the flow table listing displays the index of the record, the source ip: source port, the destination ip: destination port, the inet protocol, and the source vrf to which the flow belongs.

Each new flow has to be approved by the vrouter agent. The agent does this by setting actions for each flow. There are three main actions associated with a flow table entry: Forward ('F'), Drop ('D'), and Nat ('N').

For NAT, there are additional flags indicating the type of NAT to which the flow is subject, including: SNAT (S), DNAT (D), source port translation (Ps), and destination port translation (Pd).

S(nh) indicates the source nexthop index used for the RPF check to validate that the traffic is from a known source. If the packet must go to an ECMP destination, E:X is also displayed, where 'X' indicates the destination to be used through the index within the ECMP next hop.

The Statistics field indicates the Packets/Bytes that hit this flow entry.

There is a Mirror Index field if the traffic is mirrored, listing the indices into the mirror table (which can be dumped by using **mirror --dump**).

If there is an explicit association between the forward and the reverse flows, as is the case with NAT, you will see a double arrow in each of the records with either side of the arrow displaying the flow index for that direction.

Example: flow -r

Use **-r** to view all of the flow setup rates.

```
# flow -r
New =    2, Flow setup rate =    3 flows/sec, Flow rate =    3 flows/sec,
for last  548 ms
New =    2, Flow setup rate =    3 flows/sec, Flow rate =    3 flows/sec,
for last  543 ms
New =   -2, Flow setup rate =   -3 flows/sec, Flow rate =   -3 flows/sec,
for last  541 ms
New =    2, Flow setup rate =    3 flows/sec, Flow rate =    3 flows/sec,
for last  544 ms
New =   -2, Flow setup rate =   -3 flows/sec, Flow rate =   -3 flows/sec,
for last  542 ms
```

Example: flow --help

Use **--help** to display all options available for the flow command.

```
# flow --help
Usage:flow [-f flow_index][-d flow_index][-i flow_index]
           [--mirror=mirror table index]
           [-l]
    -f <flow_index> Set forward action for flow at flow_index <flow_index>
    -d <flow_index> Set drop action for flow at flow_index <flow_index>
    -i <flow_index> Invalidate flow at flow_index <flow_index>
    --mirror          mirror index to mirror to
    -l                List  all flows
    -r                Start dumping flow setup rate
    --help            Print this help
```

vrfstats Command

Use **vrfstats** to display statistics per next hop for a **vrf**. It is typically used to determine if packets are hitting the expected next hop.

Example: vrfstats --dump

The **--dump** option displays the statistics for all vrfs that have seen traffic. In the following example, there was traffic only in **Vrf 0** (the public vrf). **Receives** shows the number of packets that came in the fabric destined to this location. **Encaps** shows the number of packets destined to the fabric.

If there is VM traffic going out on the fabric, the respective tunnel counters will increment.

```
# vrfstats --dump
Vrf: 0
Discards 414, Resolves 3, Receives 165334
Ecmp Composites 0, L3 Mcast Composites 0, L2 Mcast Composites 0, Fabric
Composites 0, Multi Proto Composites 0
Udp Tunnels 0, Udp Mpls Tunnels 0, Gre Mpls Tunnels 0
L2 Encaps 0, Encaps 130955
```

Example: vrfstats --get 0

Use **--get 0** to retrieve statistics for a particular **vrf**.

```
# vrfstats --get 0
Vrf: 0
Discards 418, Resolves 3, Receives 166929
Ecmp Composites 0, L3 Mcast Composites 0, L2 Mcast Composites 0, Fabric
Composites 0, Multi Proto Composites 0
Udp Tunnels 0, Udp Mpls Tunnels 0, Gre Mpls Tunnels 0
L2 Encaps 0, Encaps 132179
```

Example: vrfstats --help

```
Usage: vrfstats --get <vrf>
                                --dump
                                --help

--get <vrf>      Displays packet statistics for the vrf <vrf>

--dump           Displays packet statistics for all vrfs

--help           Displays this help message
```

rt Command

Use the `rt` command to display all routes in a vrf.

Example: `rt --dump`

The following example displays **inet** family routes for **vrf 0**.

```
# rt --dump 0

Kernel IP routing table 0/0/unicast
```

Destination	PPL	Flags	Label	Nexthop
0.0.0.0/8	0		-	5
1.0.0.0/8	0		-	5
2.0.0.0/8	0		-	5
3.0.0.0/8	0		-	5
4.0.0.0/8	0		-	5
5.0.0.0/8	0		-	5

In this example output, the first line displays the routing table that is being dumped. In **0/0/unicast**, the first 0 is for the router id, the next 0 is for the vrf id, and unicast identifies the unicast table. The vrouters maintain separate tables for unicast and multicast routes. By default, if the **--table** option is not specified, only the unicast table is dumped.

Each record in the table output specifies the destination prefix length, the parent route prefix length from which this route has been expanded, the flags for the route, the MPLS label if the destination is a VM in another location, and the next hop id. To understand the second field "PPL", it is good to keep in mind that the unicast routing table is internally implemented as an 'mtree'.

The **Flags** field can have two values. **L** indicates that the label field is valid, and **H** indicates that **vroutd** should proxy arp for this IP.

The **Nexthop** field indicates the next hop ID to which the route points.

Example: `rt --dump --table mcst`

To dump the multicast table, use the **—table** option with **mcst** as the argument.

```
# rt --dump 0 --table mcst

Kernel IP routing table 0/0/multicast

(Src,Group)                                Nexthop

0.0.0.0,255.255.255.255
```

dropstats Command

Use the dropstats command to see packet drop counters in vrouter.

Example: dropstats

```
# dropstats

GARP                                0

ARP notme                          12904

Invalid ARPs                       0

Invalid IF                         0

Trap No IF                         0

IF TX Discard                      0

IF Drop                            49

IF RX Discard                      0

Flow Unusable                      0

Flow No Memory                     0
```

Flow Table Full	0
Flow NAT no rflow	0
Flow Action Drop	0
Flow Action Invalid	0
Flow Invalid Protocol	0
Flow Queue Limit Exceeded	0
Discards	34
TTL Exceeded	0
Mcast Clone Fail	0
Cloned Original	0
Invalid NH	2
Invalid Label	0
Invalid Protocol	0
Rewrite Fail	0
Invalid Mcast Source	0
Push Fails	0
Pull Fails	0
Duplicated	0
Head Alloc Fails	0

Head Space Reserve Fails	0
PCOW fails	0
Invalid Packet	0
Misc	0
Nowhere to go	0
Checksum errors	0
No Fmd	0
Ivalid VNID	0
Fragment errors	0
Invalid Source	0

dropstats ARP Block

GARP packets from VMs are dropped by vrouter, an expected behavior. In the example output, the first counter GARP indicates how many packets were dropped.

ARP requests that are not handled by vrouter are dropped, for example, requests for a system that is not a host. These drops are counted by **ARP notme** counters.

The **Invalid ARPs** counter is incremented when the Ethernet protocol is ARP, but the ARP operation was neither a request nor a response.

dropstats Interface Block

Invalid IF counters are incremented normally during transient conditions, and should not be a concern.

Trap No IF counters are incremented when vrouter is not able to find the interface to trap the packets to vrouter agent, and should not happen in a working system.

IF TX Discard and **IF RX Discard** counters are incremented when vrouter is not in a state to transmit and receive packets, and typically happens when vrouter goes through a reset state or when the module is unloaded.

IF Drop counters indicate packets that are dropped in the interface layer. The increase can typically happen when interface settings are wrong.

dropstats Flow Block

When packets go through flow processing, the first packet in a flow is cached and the vrouter agent is notified so it can take actions on the packet according to the policies configured. If more packets arrive after the first packet but before the agent makes a decision on the first packet, then those new packets are dropped. The dropped packets are tracked by the Flow unusable counter.

The **Flow No Memory** counter increments when the flow block doesn't have enough memory to perform internal operations.

The **Flow Table Full** counter increments when the vrouter cannot install a new flow due to lack of available slots. A particular flow can only go in certain slots, and if all those slots are occupied, packets are dropped. It is possible that the flow table is not full, but the counter might increment.

The **Flow NAT no rflow** counter tracks packets that are dropped when there is no reverse flow associated with a forward flow that had action set as NAT. For NAT, the vrouter needs both forward and reverse flows to be set properly. If they are not set, packets are dropped.

The **Flow Action Drop** counter tracks packets that are dropped due to policies that prohibit a flow.

The **Flow Action Invalid** counter usually does not increment in the normal course of time, and can be ignored.

The **Flow Invalid Protocol** usually does not increment in the normal course of time, and can be ignored.

The **Flow Queue Limit Exceeded** usually does not increment in the normal course of time, and can be ignored.

dropstats Miscellaneous Operational Block

The **Discard** counter tracks packets that hit a discard next hop. For various reasons interpreted by the agent and during some transient conditions, a route can point to a discard next hop. When packets hit that route, they are dropped.

The **TTL Exceeded** counter increments when the MPLS time-to-live goes to zero.

The **Mcast Clone Fail** happens when the vrouter is not able to replicate a packet for flooding.

The **Cloned Original** is an internal tracking counter. It is harmless and can be ignored.

The **Invalid NH** counter tracks the number of packets that hit a next hop that was not in a state to be used (usually in transient conditions) or a next hop that was not expected, or no next hops when there was a next hop expected. Such increments happen rarely, and should not continuously increment.

The **Invalid Label** counter tracks packets with an MPLS label unusable by vrouter because the value is not in the expected range.

The **Invalid Protocol** typically increments when the IP header is corrupt.

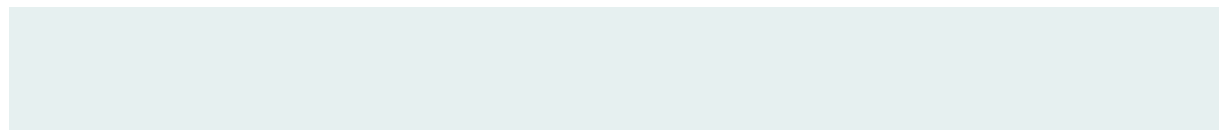
The **Rewrite Fail** counter tracks the number of times vrouter was not able to write next hop rewrite data to the packet.

The **Invalid Mcast Source** tracks the multicast packets that came from an unknown or unexpected source and thus were dropped.

The **Duplicated** counter tracks the number of duplicate packets that are created after dropping the original packets. An original packet is duplicated when generic send offload (GSO) is enabled in the vRouter or the original packet is unable to include the header information of the vRouter agent.

The **Invalid Source** counter tracks the number of packets that came from an invalid or unexpected source and thus were dropped.

The remaining counters are of value only to developers.



mpls Command

The **mpls** utility command displays the input label map that has been programmed in the vrouter.

Example: mpls --dump

The **—dump** command dumps the complete label map. The output is divided into two columns. The first field is the label and the second is the next hop corresponding to the label. When an MPLS packet with the specified label arrives in the vrouter, it uses the next hop corresponding to the label to forward the packet.

```
# mpls -dump

MPLS Input Label Map
```

Label	NextHop
16	9
17	11

You can inspect the operation on **nh 9** as follows:

```
# nh --get 9

Id:009  Type:Encap      Fmly: AF_INET  Flags:Valid, Policy,  Rid:0  Ref_cnt:4

      EncapFmly:0806 Oif:3 Len:14 Data:02 d0 60 aa 50 57 00 25 90 c3 08 69 08 00
```

The nh output shows that the next hop directs the packet to go out on the interface with index 3 (**Oif:3**) with the given rewrite data.

To check the index of 3, use the following:

```
# vif -get 3

vif0/3  OS: tapd060aa50-57

      Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:0

      Vrf:1  Flags:PL3L2 MTU:9160 Ref:6

      RX packets:1056  bytes:103471 errors:0
```

```
TX packets:1041  bytes:102372  errors:0
```

The **-get 3** output shows that the index of 3 corresponds to a tap interface that goes to a VM.

You can also dump individual entries in the map using the **-get** option, as follows:

```
# mpls -get 16

MPLS Input Label Map

Label      NextHop
-----
16         9
```

Example: mpls -help

```
# mpls -help

Usage: mpls --dump

        mpls --get <label>

        mpls --help

--dump  Dumps the mpls incoming label map

--get    Dumps the entry corresponding to label <label>
         in the label map

--help   Prints this help message
```

mirror Command

Use the **mirror** command to dump the mirror table entries.

Example: Inspect Mirroring

The following example inspects a mirror configuration where traffic is mirrored from network **vn1 (1.1.1.0/24)** to network **vn2 (2.2.2.0/24)**. A ping is run from 1.1.1.253 to 2.2.2.253, where both IPs are valid VM IPs, then the flow table is listed:

```
# flow -l

Flow table

Index          Source:Port      Destination:Port  Proto(V)
-----
135024         2.2.2.253:1208   1.1.1.253:0      1 (1)
               (Action:F, S(nh):17,  Statistics:208/17472 Mirror Index :
0)

387324         1.1.1.253:1208   2.2.2.253:0      1 (1)
               (Action:F, S(nh):8,  Statistics:208/17472 Mirror Index :
0)
```

In the example output, **Mirror Index:0** is listed, it is the index to the mirror table. The mirror table can be dumped with the **—dump** option, as follows:

```
# mirror --dump

Mirror Table

Index   NextHop   Flags   References
-----
0       18        3
```

The mirror table entries point to next hops. In the example, the index 0 points to next hop 18. The **References** indicate the number of flow entries that point to this entry.

A next hop get operation on ID 18 is performed as follows:

```
# nh --get 18

Id:018  Type:Tunnel      Fmly: AF_INET  Flags:Valid, Udp,   Rid:0  Ref_cnt:2

      Oif:0 Len:14 Flags Valid, Udp,  Data:00 00 00 00 00 00 00 25 90 c3 08 69 08
00

      Vrf:-1  Sip:192.168.1.10  Dip:250.250.2.253

      Sport:58818 Dport:8099
```

The **nh --get** output shows that mirrored packets go to a system with IP 250.250.2.253. The packets are tunneled as a UDP datagram and sent to the destination. **Vrf:-1** indicates that a lookup has to be done in the source **Vrf** for the destination.

You can also get an individual mirror table entry using the **--get** option, as follows:

```
# mirror --get 10

Mirror Table

Index      NextHop      Flags      References
-----
10         1             1
```

Example: mirror --help

```
# mirror --help

Usage:  mirror --dump

        mirror --get <index>

        mirror --help

--dump  Dumps the mirror table

--get   Dumps the mirror entry corresponding to index <index>
```

```
--help      Prints this help message
```

vxlan Command

The vxlan command can be used to dump the vxlan table. The vxlan table maps a network ID to a next hop, similar to an MPLS table.

If a packet comes with a vxlan header and if the VNID is one of those in the table, the vrouter will use the next hop identified to forward the packet.

Example: vxlan --dump

```
# vxlan --dump

VXLAN Table

VNID      NextHop
-----
4          16
5          16
```

Example: vxlan --get

You can use the **--get** option to dump a specific entry, as follows:

```
# vxlan --get 4

VXLAN Table

VNID      NextHop
-----
```

```
4          16
```

Example: vxlan --help

```
# vxlan --help

Usage:  vxlan --dump

        vxlan --get <vnid>

        vxlan --help

--dump  Dumps the vxlan table

--get   Dumps the entry corresponding to <vnid>

--help  Prints this help message
```

nh Command

The **nh** command enables you to inspect the next hops that are known by the vrouter. Next hops tell the vrouter the next location to send a packet in the path to its final destination. The processing of the packet differs based on the type of the next hop. The next hop types are described in the following table.

Next Hop Type	Description
Receive	Indicates that the packet is destined for itself and the vrouter should perform Layer 4 protocol processing. As an example, all packets destined to the host IP will hit the receive next hop in the default VRF. Similarly, all traffic destined to the VMs hosted by the server and tunneled inside a GRE will hit the receive next hop in the default VRF first, because the outer packet that carries the traffic to the VM is that of the server.

Next Hop Type	Description
Encap (Interface)	Used only to determine the outgoing interface and the Layer 2 information. As an example, when two VMs on the same server communicate with each other, the routes for each of them point to an encap next hop, because the only information needed is the Layer 2 information to send the packet to the tap interface of the destination VM. A packet destined to a VM hosted on one server from a VM on a different server will also hit an encap next hop, after tunnel processing.
Tunnel	Encapsulates VM traffic in a tunnel and sends it to the server that hosts the destination VM. There are different types of tunnel next hops, based on the type of tunnels used. Vrouter supports two main tunnel types for Layer 3 traffic: MPLSoGRE and MPLSoUDP. For Layer 2 traffic, a VXLAN tunnel is used. A typical tunnel next hop indicates the kind of tunnel, the rewrite information, the outgoing interface, and the source and destination server IPs.
Discard	A catch-all next hop. If there is no route for a destination, the packet hits the discard next hop, which drops the packet.
Resolve	Used by the agent to lazy install Layer 2 rewrite information.
Composite	Groups a set of next hops, called component next hops or sub next hops. Typically used when multi-destination distribution is needed, for example for multicast, ECMP, and so on.
Vxlan	A VXLAN tunnel is used for Layer 2 traffic. A typical tunnel next hop indicates the kind of tunnel, the rewrite information, the outgoing interface, and the source and destination server IPs.

Example: nh --list

```

Id:000  Type:Drop      Fmly: AF_INET  Flags:Valid,   Rid:0  Ref_cnt:1781

Id:001  Type:Resolve   Fmly: AF_INET  Flags:Valid,   Rid:0  Ref_cnt:244

Id:004  Type:Receive   Fmly: AF_INET  Flags:Valid, Policy,   Rid:0

Ref_cnt:2 Oif:1

```

```

Id:007  Type:Encap      Fmly: AF_INET  Flags:Valid, Multicast,  Rid:0
Ref_cnt:3

      EncapFmly:0806 Oif:3 Len:14 Data:ff ff ff ff ff ff 00 25 90 c4 82 2c
      08 00

Id:010  Type:Encap      Fmly:AF_BRIDGE  Flags:Valid, L2,  Rid:0  Ref_cnt:3

      EncapFmly:0000 Oif:3 Len:0 Data:

Id:012  Type:Vxlan Vrf  Fmly: AF_INET  Flags:Valid,  Rid:0  Ref_cnt:2

      Vrf:1

Id:013  Type:Composite  Fmly: AF_INET  Flags:Valid, Fabric,  Rid:0  Ref_cnt:3

      Sub NH(label): 19(1027)

Id:014  Type:Composite  Fmly: AF_INET  Flags:Valid, Multicast, L3,  Rid:0
Ref_cnt:3

      Sub NH(label): 13(0) 7(0)

Id:015  Type:Composite  Fmly:AF_BRIDGE  Flags:Valid, Multicast, L2,  Rid:0
Ref_cnt:3

      Sub NH(label): 13(0) 10(0)

Id:016  Type:Tunnel     Fmly: AF_INET  Flags:Valid, MPLSoGRE,  Rid:0  Ref_cnt:1

      Oif:2 Len:14 Flags Valid, MPLSoGRE,  Data:00 25 90 aa 09 a6 00 25 90
      c4 82 2c 08 00

      Vrf:0  Sip:10.204.216.72  Dip:10.204.216.21

Id:019  Type:Tunnel     Fmly: AF_INET  Flags:Valid, MPLSoUDP,  Rid:0  Ref_cnt:7

      Oif:2 Len:14 Flags Valid, MPLSoUDP,  Data:00 25 90 aa 09 a6 00 25 90
      c4 82 2c 08 00

```



```

Vrf:0  Sip:10.204.216.72  Dip:10.204.216.21

Id:020  Type:Composite  Fmly:AF_UNSPEC  Flags:Valid, Multi Proto,  Rid:0
Ref_cnt:2

Sub NH(label): 14(0) 15(0)

```

Example: nh --get

Use the **--get** option to display information for a single next hop.

```

# nh -get 9

Id:009  Type:Encap      Fmly:AF_BRIDGE  Flags:Valid, L2,  Rid:0  Ref_cnt:4

EncapFmly:0000 Oif:3 Len:0 Data:

```

Example: nh --help

```

# nh -help

Usage: nh --list

      nh --get <nh_id>

      nh --help

--list  Lists All Nexthops

--get   <nh_id> Displays nexthop corresponding to <nh_id>

--help  Displays this help message

```

3

PART

Monitoring and Troubleshooting Contrail

Configuring Traffic Mirroring to Monitor | **82**

Understanding Contrail Analytics | **97**

Configuring Contrail Analytics | **126**

Using Contrail Analytics to Monitor and Troubleshoot the Network | **151**

Common Support Answers | **218**

Configuring Traffic Mirroring to Monitor

IN THIS CHAPTER

- [Configuring Traffic Analyzers and Packet Capture for Mirroring | 82](#)
- [Configuring Interface Monitoring and Mirroring | 89](#)
- [Mirroring Enhancements | 91](#)
- [Analyzer Service Virtual Machine | 92](#)
- [Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 95](#)

Configuring Traffic Analyzers and Packet Capture for Mirroring

IN THIS SECTION

- [Traffic Analyzer Images | 82](#)
- [Configuring Traffic Analyzers | 83](#)
- [Setting Up Traffic Mirroring Using Configure > Networking > Services | 83](#)

Contrail provides traffic mirroring so you can mirror specified traffic to a traffic analyzer where you can perform deep traffic inspection. Traffic mirroring enables you to designate certain traffic flows to be mirrored to a traffic analyzer, where you can view traffic flows in great detail.

This section describes how to set up packet capture to mirror traffic packets to an analyzer.

Traffic Analyzer Images

Before using the Contrail interface to configure traffic analyzers and packet capture for mirroring, make sure that the following analyzer images are available in the VM image list for your system. The traffic analyzer images are enhanced for viewing details of captured packets in Wireshark. When creating a policy for the traffic analyzer, the traffic analyzer instance should always have the **Mirror to** field selected in the policy, do not select the **Apply Service** field for a traffic analyzer.

- **analyzer-vm-console-qcow2**—Standard traffic analyzer; should be named **analyzer** in the image list. This type of traffic analyzer is always configured with a single interface, and the interface should be a **Left** interface.
- **analyzer-vm-console-two-if qcow2**—This type of traffic analyzer has two interfaces, **Left** and **Management**. This traffic analyzer can have any name except the name **analyzer**, which is reserved for the single interface analyzer.

NOTE: The **analyzer-vm** images are valid for all versions of Contrail. Download the images from the Contrail 1.0 software download page:
<https://www.juniper.net/support/downloads/?p=contrail#sw> .

Configuring Traffic Analyzers

Contrail Controller enables you to mirror captured packet traffic to a traffic analyzer. Follow these steps to mirror captured packet traffic:

1. Configure analyzer(s) on the host.
2. Set up rules for packet capture.

You can set up traffic mirroring using **Configure > Networking > Services**. For more information, see “[Setting Up Traffic Mirroring Using Configure > Networking > Services](#)” on page 83.

Setting Up Traffic Mirroring Using Configure > Networking > Services

Follow these steps to set up traffic mirroring using **Configure > Networking > Services**.

1. Access **Configure > Services > Service Templates**.

The **Service Templates** screen appears; see [Figure 16 on page 84](#).

Figure 16: Service Templates

Template	Mode	Type / Version	Interface (s)	Image & Flavor
<input type="checkbox"/> netns-snat-template	In-network-nat	Source-nat / v1	Right, Left	- / -
<input type="checkbox"/> haproxy-loadbalancer-template	In-network-nat	Loadbalancer / v1	Right, Left	- / -
<input type="checkbox"/> docker-template	Transparent	Firewall / v1	Management, Left, Right	ubuntu / -
<input type="checkbox"/> nat-template	In-network-nat	Firewall / v1	Management, Left, Right	analyzer / m1.medium

Total: 4 records 50 Records

2. To create a new service template, click the + icon.

The **Create** window appears. Select the Service Template tab; see [Figure 17 on page 84](#).

Figure 17: Create Service Template

Service Template

Permissions

Name

analyzer-service-template

Version

v2

Virtualization Type

Virtual Machine

Service Mode

Transparent

Service Type

Analyzer

Interface (s)

management

left

Cancel

Save

3. Complete the fields by using the guidelines in [Table 29 on page 85](#).

Table 29: Create Service Template Fields

Field	Description
Name	Enter a descriptive text name for this service template.
Version	Select v2 from the drop-down list to indicate that this service template is based on templates version 2, valid for Contrail 3.0 and later.
Virtualization Type	Select Virtual Machine from the drop-down list to indicate the virtualization type for mirroring for this template.
Service Mode	Select Transparent from the drop-down list to indicate that this service template is for transparent mirroring.
Service Type	Select Analyzer from the drop-down list to indicate that this service template is for a traffic analyzer.
Interface(s)	<p>From the drop-down list, click the check boxes to indicate which interface types are used for this analyzer service template:</p> <ul style="list-style-type: none"> • Left • Right • Management
Save	When finished, click OK to commit the changes
Cancel	Click Cancel to clear the fields and start over.

4. Create a service instance by clicking the **Service Instances** link and clicking the + icon.

The **Create** window appears; make sure the Service Instance tab is selected. See [Figure 18 on page 86](#).

Figure 18: Create Service Instances

Create

Service Instance Permissions

Name

Service Template

Interface Type

Virtual Network

▶ Port Tuples

▶ Service Health Check

▶ Allowed Address Pair

▶ Static Route

Cancel Save

s041858

5. Complete the fields by using the guidelines in [Table 30 on page 86](#).

Table 30: Create Service Instances Fields

Field	Description
Name	Enter a text name for this service instance.
Service Template	Select from a drop-down list of available service templates the template to use for this service instance, analyzer-service-template in this example.
Interface Type	Each interface configured in the service template for this instance appears in a list.
Virtual Network	Select from a drop-down list of available virtual networks the network for each interface that is configured for the instance.
Save	Click Save to commit your changes.
Cancel	Click Cancel to clear your changes and start over.

- 6. To create a network policy rule for this service instance, click **Configure > Networking > Policies**. The **Policies** window appears. Click the + icon to get to the **Create** window; see [Figure 19 on page 87](#).

Figure 19: Create Policy

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror	QoS	+
--------	----------	--------	-------	-----------	-------------	-------	-----	----------	--------	-----	---

- 7.
- 8. Enter a name for the policy, then click the + icon in the lower portion of the screen to configure rules for the policy, see [Figure 20 on page 87](#).

Figure 20: Create Policy Rules

Action	Protocol	Source	Ports	Direction	Destination	Ports	Log	Services	Mirror	QoS	+	-
PASS	ANY	ANY (All Networks i...)	ANY	<>	ANY (All Networks i...)	ANY	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

- 9. To add policy rules, complete the fields, using the guidelines in [Table 31 on page 88](#).

NOTE: When there is a network policy attached to the virtual network, any conflicting rules configured for the analyzer will not take effect.

Table 31: Add Rule Fields

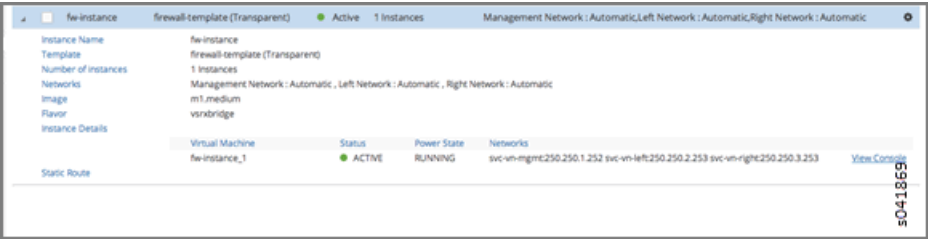
Field	Description
Action	Select PASS or DENY as the rule action.
Protocol	Select the protocol for the policy rule, or select ANY.
Source	Select from multiple drop-down lists the source for this rule, including options under CIDR, Network, Policy, or Security Group.
Ports	Select from a drop-down list the source ports for the rule.
Direction	Select the direction of flow for the packets to be captured: <ul style="list-style-type: none"> • <> (bidirectional) • > (unidirectional)
Destination	Select from multiple drop-down lists the destination for this rule, including options under CIDR, Network, Policy, or Security Group.
Ports	Select from a list the destination ports for the packets to be captured.
check boxes	Check any box that applies to this rule: Log, Services, Mirror, QoS.
Save	Click Save to commit your changes.
Cancel	Click Cancel to clear your changes and start over.

10. When finished, click **Save**.

11. To verify packet capture, at **Configure > Services > Service Instances**, select the analyzer service instance and click **View Console**.

The packet capture displays; see [Figure 21 on page 89](#). The analyzer service VM launches the Contrail-enhanced Wireshark as it starts and captures the mirrored packets destined to this service.

Figure 21: Service Instances View Console



RELATED DOCUMENTATION

- [Configuring Interface Monitoring and Mirroring | 89](#)
- [Mirroring Enhancements | 91](#)
- [Analyzer Service Virtual Machine | 92](#)
- [Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 95](#)

Configuring Interface Monitoring and Mirroring

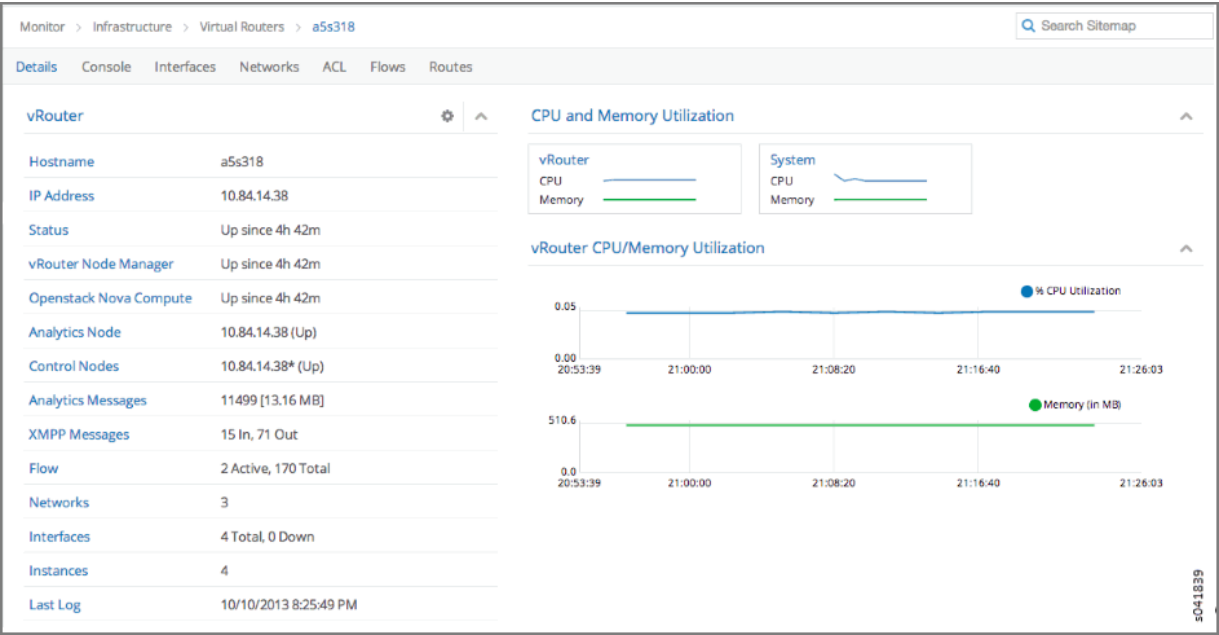
Contrail supports user monitoring of traffic on any guest virtual machine interface when using the Juniper Contrail user interface.

When interface monitoring (packet capture) is selected, a default analyzer is created and all traffic from the selected interface is mirrored and sent to the default analyzer. If a mirroring instance is already launched, the traffic will be redirected to the selected instance. The interface traffic is only mirrored during the time that the monitor packet capture interface is in use. When the capture screen is closed, interface mirroring stops.

To configure interface mirroring:

1. Select **Monitor > Infrastructure > Virtual Routers**, then select the vRouter that has the interface to mirror.
2. In the list of attributes for the vRouter, select Interfaces; see [Figure 22 on page 90](#).

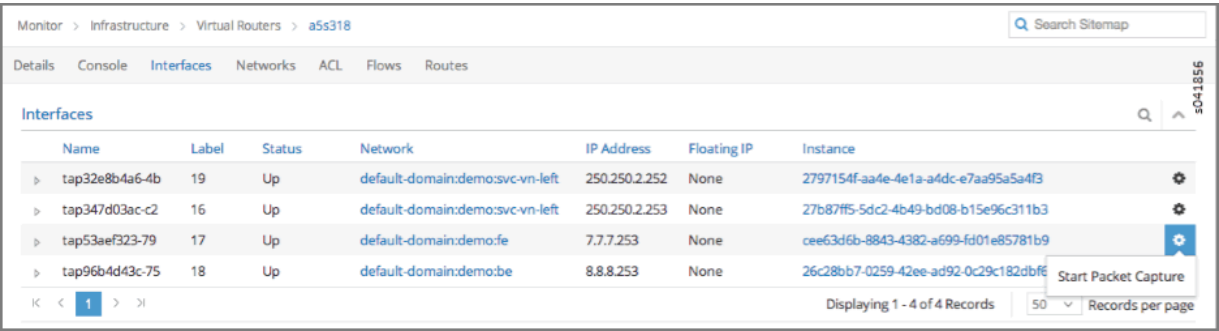
Figure 22: Individual vRouter



A list of interfaces for that vRouter appears.

3. For the interface to mirror, click the Action icon in the last column and select the option Packet Capture; see [Figure 23 on page 90](#).

Figure 23: Interfaces



The mirror packet capture starts and displays at this screen.

The mirror packet capture stops when you exit this screen.

RELATED DOCUMENTATION

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 82](#)

[Mirroring Enhancements | 91](#)

Mirroring Enhancements

Mirroring Specified Traffic

Specific traffic can be mirrored to a traffic analyzer in Contrail by:

- Configuring rules to identify the flows to be mirrored, and
- Specifying the analyzer to which the traffic is mirrored

Additionally, mirroring can be configured on virtual machine (VM) interfaces to send all the traffic to and from the interface to the specified analyzer.

Configuring Headers and Next Hops

When a packet is mirrored, a Juniper header is added to provide additional information in the analyzer, then the packet is encapsulated and sent to the destination.

Starting with Contrail 3.x releases, mirroring is enhanced with the following options:

- Option to control addition of the Juniper header in the mirrored packet.
 - When disabled, the Juniper header is not added to the mirrored packet.
- Option to control whether the next hop used is dynamic or static.
 - If dynamic is selected, the next hop based on the destination is used. Packets are forwarded to the destination based on the encapsulation priority.
 - If static is chosen, the next hop is created for the specified destination with VxLAN encapsulation using the configured VNI, destination VTEP, and MAC to transmit the mirrored packets.

The following combinations are supported:

- Dynamic next hop with Juniper header added

The default combination and the only supported case up to Release 3.0.2

- Dynamic next hop, without Juniper header
- Static next hop, without Juniper header, with the original Layer 2 packet

How Mirroring is Implemented

The Contrail vrouter agent adds a mirror entry in the vrouter and points to the next hop to be used. The data for the Juniper header is taken from the flow entry. For interface mirroring, the Juniper header has a TLV in the metadata to use the interface name instead of providing a destination VN.

For more information about implementation details, see <https://github.com/Juniper/contrail-controller/wiki/Mirroring>.

RELATED DOCUMENTATION

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 82](#)

[Configuring Interface Monitoring and Mirroring | 89](#)

[Analyzer Service Virtual Machine | 92](#)

[Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 95](#)

Analyzer Service Virtual Machine

IN THIS SECTION

- [Packet Format for Analyzer | 92](#)
- [Metadata Format | 93](#)
- [Wireshark Changes | 94](#)
- [Troubleshooting Packet Display | 94](#)

The analyzer service virtual machine (**analyzer-vm-console.qcow2**) launches a Contrail-enhanced version of the network protocol analyzer Wireshark as the analyzer starts capturing mirror packets destined to the analyzer service.

Packet Format for Analyzer

The analyzer uses the PCAP format, which has these parts:

- Global header
- PCAP packet header

Length is 2 bytes. Multiple bits might be turned on, if there are more actions. Ingress or egress bit will be present in the Action field.

Source VN or Destination VN

Length is variable and up to 256 characters

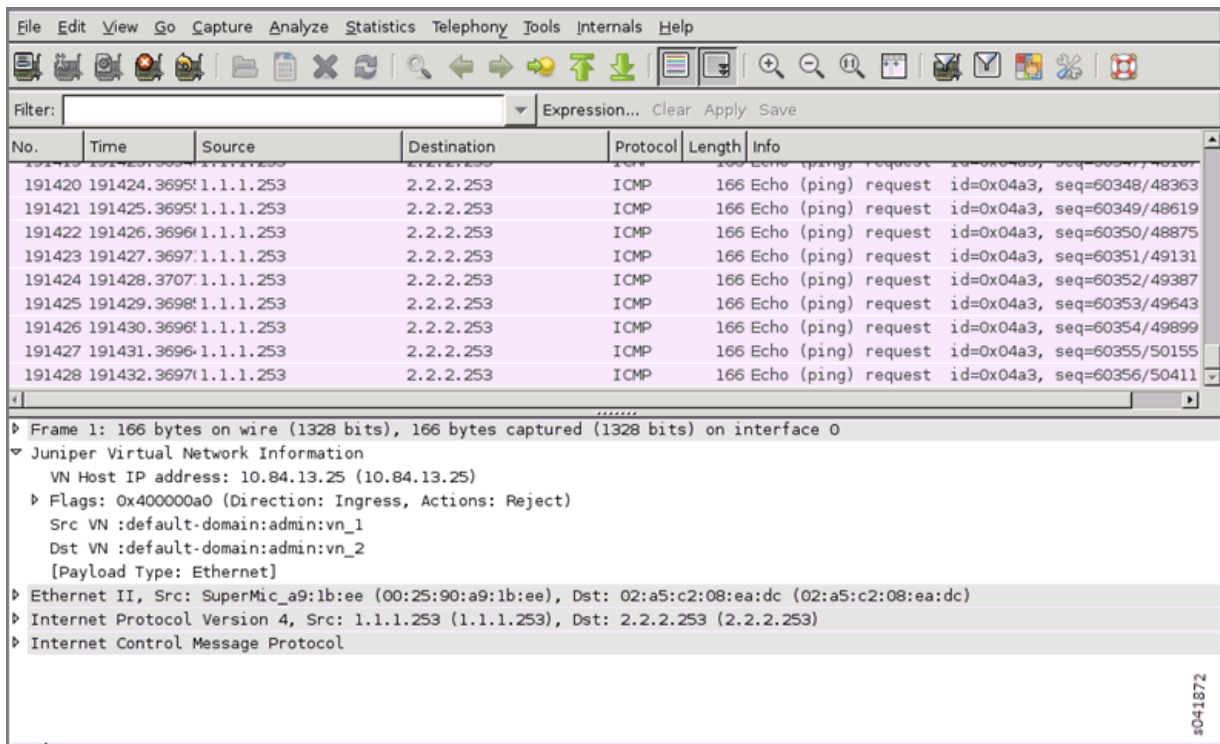
TLV end

A special type **255 (0xFF)** is used to identify the end of TLV entries. The TLV end must be last, at the end of the metadata.

Wireshark Changes

A plugin is added to the Wireshark code. The plugin parses the metadata and displays the packet fields; see example in [Figure 24 on page 94](#).

Figure 24: Wireshark Packet Display



Troubleshooting Packet Display

Follow these steps if the packets are not displaying:

1. Use **tcpdump** on the tap interfaces to see if packets are going towards the analyzer VM.

2. Check introspect to see whether the flow action has mirror activity in it or not.

RELATED DOCUMENTATION

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 82](#)

[Configuring Interface Monitoring and Mirroring | 89](#)

[Mirroring Enhancements | 91](#)

[Mapping VLAN Tags from a Physical NIC to a VMI \(NIC-Assisted Mirroring\) | 95](#)

Mapping VLAN Tags from a Physical NIC to a VMI (NIC-Assisted Mirroring)

When mirroring is enabled, the vRouter throughput reduces because of the additional packet handling overhead caused by cloning the packet to be mirrored, encapsulating it in the required header, and forwarding it to the mirror destination. Impact to throughput increases in proportion to the amount of traffic that needs to be mirrored.

A solution to avoid impact on throughput due to mirroring is to use the mirroring capabilities of an installed Network Interface Card (NIC).

Contrail Release 4.0 has the ability to mirror specific traffic to a traffic analyzer or to a physical probe using the Network interface card (NIC) instead of the vRouter to mirror packets. When NIC-assisted mirroring is enabled, ingress packets to be mirrored sent from a VM are routed to the NIC with a configured VLAN tag. The NIC is configured for VLAN port-mirroring and mirrors any packet with the VLAN tag.

In this approach, the vRouter doesn't mirror the packets. When NIC-assisted mirroring is enabled, the ingress packets coming from the VM that are to be mirrored are sent to the NIC with a configured VLAN tag.

The NIC is programmed to do VLAN port mirroring, so that any packet with the configured VLAN is mirrored additionally by the NIC. This change in vRouter is only for traffic coming from the VMs. Traffic coming from the fabric is directly mirrored from the NIC itself and there is no additional mirroring need in vRouter. The programming of the NIC itself for appropriate mirroring is outside the scope of the current activity. An example is the Niantic 82599 10G NIC, which supports VLAN port mirroring options.

The following are cautions to observe when using NIC-assisted mirroring:

- VM traffic sent to another VM running on the same compute node will not be mirrored when NIC-assisted mirroring is selected.
- Traffic coming in from the fabric interface will not be mirrored.

- When a VLAN interface is used as the fabric interface, traffic will be tagged first with the NIC-assisted mirroring VLAN, followed by the VLAN tag on the fabric interface. The NIC-assisted mirroring VLAN will be the inner tag and the fabric interface VLAN will be the outer tag.

The NIC must be programmed for VLAN port mirroring. While configuring mirroring in Contrail, the user can indicate NIC-assisted mirroring with the VLAN tag. The Contrail UI supports NIC-assisted mirroring configuration in the Ports page and in the Policies page with an additional flag for NIC-assisted mirroring and the VLAN tag to be used.

RELATED DOCUMENTATION

[Configuring Traffic Analyzers and Packet Capture for Mirroring | 82](#)

[Configuring Interface Monitoring and Mirroring | 89](#)

[Mirroring Enhancements | 91](#)

[Analyzer Service Virtual Machine | 92](#)

Understanding Contrail Analytics

IN THIS CHAPTER

- [Understanding Contrail Analytics | 97](#)
- [Contrail Alerts | 98](#)
- [Underlay Overlay Mapping in Contrail | 102](#)

Understanding Contrail Analytics

Contrail is a distributed system of compute nodes, control nodes, configuration nodes, database nodes, web UI nodes, and analytics nodes.

The analytics nodes are responsible for the collection of system state information, usage statistics, and debug information from all of the software modules across all of the nodes of the system. The analytics nodes store the data gathered across the system in a database that is based on the Apache Cassandra open source distributed database management system. The database is queried by means of an SQL-like language and representational state transfer (REST) APIs.

System state information collected by the analytics nodes is aggregated across all of the nodes.

Debug information collected by the analytics nodes includes the following types:

- System log (syslog) messages—informational and debug messages generated by system software components.
- Object log messages—records of changes made to system objects such as virtual machines, virtual networks, service instances, virtual routers, BGP peers, routing instances, and the like.
- Trace messages—records of activities collected locally by software components and sent to analytics nodes only on demand.

Statistics information related to flows, CPU and memory usage, and the like is also collected by the analytics nodes and can be queried to provide historical analytics and time-series information. The queries are performed using REST APIs.

Analytics data is written to a database in Contrail. The data expires after the default time-to-live (TTL) period of 48 hours. This default TTL time can be changed as needed by changing the value of the **database_ttl** value in the cluster configuration.

RELATED DOCUMENTATION

[Contrail Alerts | 98](#)

[Analytics Scalability | 127](#)

[High Availability for Analytics | 128](#)

Ceilometer Support in Contrail

[Underlay Overlay Mapping in Contrail | 102](#)

[Monitoring the System | 152](#)

[Debugging Processes Using the Contrail Introspect Feature | 155](#)

[Monitor > Infrastructure > Dashboard | 159](#)

[Monitor > Infrastructure > Control Nodes | 162](#)

[Monitor > Infrastructure > Virtual Routers | 171](#)

[Monitor > Infrastructure > Analytics Nodes | 183](#)

[Monitor > Infrastructure > Config Nodes | 189](#)

[Monitor > Networking | 193](#)

Understanding Flow Sampling

Fat Flows

[Query > Flows | 203](#)

[Query > Logs | 212](#)

[System Log Receiver in Contrail Analytics | 129](#)

Example: Debugging Connectivity Using Monitoring for Troubleshooting

Contrail Alerts

Starting with Contrail 3.0 and greater, Contrail alerts are provided on a per-user visible entity (UVE) basis.

Contrail analytics raise or clear alerts using Python-coded rules that examine the contents of the UVE and the configuration of the object. Some rules are built in. Others can be added using Python *stevedore* plugins.

This topic describes Contrail alerts capabilities.

Alert API Format

The Contrail alert analytics API provides the following:

- Read access to the alerts as part of the UVE GET APIs.
- Alert acknowledgement using POST requests.
- UVE and alert streaming using server-sent events (SSEs).

For example:

GET `http://<analytics-ip>:8081/analytics/uves/control-node/a6s40?flat`

```
{
  NodeStatus:  {...},
  ControlCpuState:  {...},
  UVEAlarms:  {
    alarms:  [
      {
        description:  [
          {
            value:  "0 != 2",
            rule:  "BgpRouterState.num_up_bgp_peer != BgpRouterState.num_bgp_peer"
          }
        ],
        ack:  false,
        timestamp:  1442995349253178,
        token:  "eyJ0aW1lc3RhbnXAiOiAxNDQyOTk1MzQ5MTc4LCAiaHR0cF9wb3J0Ijog
NTk5NSwgImhvc3RfaXAiOiAiMTAuODQuMTMuNDAifQ==",
        type:  "BgpConnectivity",
        severity:  4
      }
    ],
  },
  BgpRouterState:  {...}
}
```

In the example:

- Alerts are raised on a per-UVE basis and can be retrieved by a GET on a UVE.
- An **ack** indicates if the alert has been acknowledged or not.
- A **token** is used by clients when requesting acknowledgements

Analytics APIs for Alerts

The following examples show the API to use to display alerts and alarms and to acknowledge alarms.

- To retrieve a list of alerts raised against the control node named **aXXsYY**.

```
GET
http://<analytics-ip>:<rest-api-port>/analytics/uves/control-node/aXXsYY&cfilt=UVEAlarms
```

This is available for all UVE table types.

- To retrieve a list of all alarms in the system.

```
GET http://<analytics-ip>:<rest-api-port>/analytics/alarms
```

- To acknowledge an alarm.

```
POST http://<analytics-ip>:<rest-api-port>/analytics/alarms/acknowledge
Body: {"table": <object-type>, "name": <key>, "type": <alarm type>, "token": <token>}
```

Acknowledged and unacknowledged alarms can be queried specifically using the following URL query parameters along with the GET operations listed previously.

```
ackFilt=True
ackFilt=False
```

Analytics APIs for SSE Streaming

The following examples show the API to use to retrieve all or portions of SE streams.

- To retrieve an SSE-based stream of UVE updates for the control node alarms.

```
GET
http://<analytics-ip>:<rest-api-port>/analytics/uve-stream?tablefilt=control-node
```

This is available for all UVE table types. If the **tablefilt** URL query parameter is not provided, all UVEs are retrieved.

- To retrieve only the alerts portion of the SSE-based stream of UVE updates instead of the entire content.

```
GET
http://<analytics-ip>:<rest-api-port>/analytics/alarm-stream?tablefilt=control-node
```

This is available for all UVE table types. If the **tablefilt** URL query parameter is not provided, all UVEs are retrieved.

Built-in Node Alerts

The following built-in node alerts can be retrieved using the APIs listed in *Analytics APIs for Alerts*.

```
control node: {
  PartialSysinfoControl: "Basic System Information is absent for this node in
  BgpRouterState.build_info",
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in
  NodeStatus.process_info",
  XmppConnectivity: "Not enough XMPP peers are up in BgpRouterState.num_up_bgp_peer",
  BgpConnectivity: "Not enough BGP peers are up in BgpRouterState.num_up_bgp_peer",
  AddressMismatch: "Mismatch between configured IP Address and operational IP Address",
  ProcessConnectivity: "Process(es) are reporting non functional components in
  NodeStatus.process_status"
},

vrouter: {
  PartialSysinfoCompute: "Basic System Information is absent for this node in
  VrouterAgent.build_info",
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in
  NodeStatus.process_info",
  ProcessConnectivity: "Process(es) are reporting non functional components in
  NodeStatus.process_status",
  VrouterInterface: "VrouterAgent has interfaces in error state in
  VrouterAgent.error_intf_list",
  VrouterConfigAbsent: "Vrouter is not present in Configuration",
},

config node: {
  PartialSysinfoConfig: "Basic System Information is absent for this node in
  ModuleCpuState.build_info",
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in
  NodeStatus.process_info",
  ProcessConnectivity: "Process(es) are reporting non functional components in
  NodeStatus.process_status"
},

analytics node: {
  ProcessStatus: "NodeMgr reports abnormal status for process(es) in
```

```

NodeStatus.process_info"
PartialSysinfoAnalytics: "Basic System Information is absent for this node in
CollectorState.build_info",
ProcessConnectivity: "Process(es) are reporting non functional components in
NodeStatus.process_status"
},

database node: {
ProcessStatus: "NodeMgr reports abnormal status for process(es) in
NodeStatus.process_info",
ProcessConnectivity: "Process(es) are reporting non functional components in
NodeStatus.process_status"
},

```

Underlay Overlay Mapping in Contrail

IN THIS SECTION

- [Overview: Underlay Overlay Mapping using Contrail Analytics | 103](#)
- [Underlay Overlay Analytics Available in Contrail | 103](#)
- [Architecture and Data Collection | 104](#)
- [New Processes/Services for Underlay Overlay Mapping | 104](#)
- [External Interfaces Configuration for Underlay Overlay Mapping | 105](#)
- [Physical Topology | 106](#)
- [SNMP Configuration | 106](#)
- [Link Layer Discovery Protocol \(LLDP\) Configuration | 106](#)
- [IPFIX and sFlow Configuration | 107](#)
- [Sending pRouter Information to the SNMP Collector in Contrail | 109](#)
- [pRouter UVEs | 110](#)
- [Contrail User Interface for Underlay Overlay Analytics | 111](#)
- [Enabling Physical Topology on the Web UI | 111](#)
- [Viewing Topology to the Virtual Machine Level | 112](#)
- [Viewing the Traffic of any Link | 112](#)
- [Trace Flows | 113](#)
- [Search Flows and Map Flows | 114](#)
- [Overlay to Underlay Flow Map Schemas | 114](#)

- [Module Operations for Overlay Underlay Mapping | 118](#)
- [SNMP Collector Operation | 118](#)
- [Topology Module Operation | 120](#)
- [IPFIX and sFlow Collector Operation | 121](#)
- [Troubleshooting Underlay Overlay Mapping | 122](#)
- [Script to add pRouter Objects | 123](#)

Overview: Underlay Overlay Mapping using Contrail Analytics

Today's cloud data centers consist of large collections of interconnected servers that provide computing and storage capacity to run a variety of applications. The servers are connected with redundant TOR switches, which in turn, are connected to spine routers. The cloud deployment is typically shared by multiple tenants, each of whom usually needs multiple isolated networks. Multiple isolated networks can be provided by overlay networks that are created by forming tunnels (for example, gre, ip-in-ip, mac-in-mac) over the underlay or physical connectivity.

As data flows in the overlay network, Contrail can provide statistics and visualization of the traffic in the underlay network.

Underlay Overlay Analytics Available in Contrail

Starting with Contrail Release 2.20, you can view a variety of analytics related to underlay and overlay traffic in the Contrail Web user interface. The following are some of the analytics that Contrail provides for statistics and visualization of overlay underlay traffic.

- View the topology of the underlay network.

A user interface view of the physical underlay network with a drill down mechanism to show connected servers (contrail computes) and virtual machines on the servers.

- View the details of any element in the topology.

You can view details of a pRouter, vRouter, or virtual machine link between two elements. You can also view traffic statistics in a graphical view corresponding to the selected element.

- View the underlay path of an overlay flow.

Given an overlay flow, you can get the underlay path used for that flow and map the path in the topology view.

Architecture and Data Collection

Accumulation of the data to map an overlay flow to its underlay path is performed in several steps across Contrail modules.

The following outlines the essential steps:

1. The SNMP collector module polls physical routers.

The SNMP collector module receives the authorizations and configurations of the physical routers from the Contrail config module, and polls all of the physical routers, using SNMP protocol. The collector uploads the data to the Contrail analytics collectors. The SNMP information is stored in the pRouter UVEs (physical router user visible entities).

2. IPFIX and sFlow protocols are used to collect the flow statistics.

The physical router is configured to send flow statistics to the collector, using one of the collection protocols: Internet Protocol Flow Information Export (IPFIX) or sFlow (an industry standard for sampled flow of packet export at Layer 2).

3. The topology module reads the SNMP information.

The Contrail topology module reads SNMP information from the pRouter UVEs from the analytics API, computes the neighbor list, and writes the neighbor information into the pRouter UVEs. This neighbor list is used by the Contrail WebUI to display the physical topology.

4. The Contrail user interface reads and displays the topology and statistics.

The Contrail user interface module reads the topology information from the Contrail analytics and displays the physical topology. It also uses information stored in the analytics to display graphs for link statistics, and to show the map of the overlay flows on the underlay network.

New Processes/Services for Underlay Overlay Mapping

The **contrail-snmp-collector** and the **contrail-topology** are new daemons that are both added to the **contrail-analytics** node. The **contrail-analytics** package contains these new features and their associated files. The **contrail-status** displays the new services.

Example: contrail-status

The following is an example of using **contrail-status** to show the status of the new process and service for underlay overlay mapping.

```
user@host:~# contrail-status

== Contrail Control ==

supervisor-control:      active

contrail-control         active

...

== Contrail Analytics ==

supervisor-analytics:    active

...

contrail-query-engine     active

contrail-snmp-collector   active

contrail-topology         active
```

Example: Service Command

The **service** command can be used to start, stop, and restart the new services. See the following example.

```
user@host:~# service contrail-snmp-collector status

contrail-snmp-collector    RUNNING   pid 12179, uptime 1 day, 14:59:11
```

External Interfaces Configuration for Underlay Overlay Mapping

This section outlines the external interface configurations necessary for successful underlay overlay mapping for Contrail analytics.

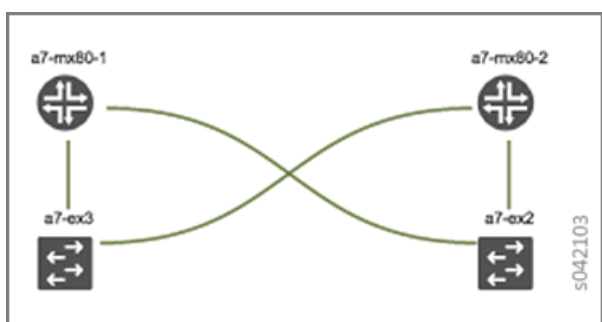
Physical Topology

The typical physical topology includes:

- Servers connected to the ToR switches.
- ToR switches connected to spine switches.
- Spine switches connected to core switches.

The following is an example of how the topology is depicted in the Contrail WebUI analytics.

Figure 25: Analytics Topology



SNMP Configuration

Configure SNMP on the physical devices so that the **contrail-snmp-collector** can read SNMP data.

The following shows an example SNMP configuration from a Juniper Networks device.

```
set snmp community public authorization read-only
```

Link Layer Discovery Protocol (LLDP) Configuration

Configure LLDP on the physical device so that the **contrail-snmp-collector** can read the neighbor information of the routers.

The following is an example of LLDP configuration on a Juniper Networks device.

```
set protocols lldp interface all
```

```
set protocols lldp-med interface all
```

IPFIX and sFlow Configuration

Flow samples are sent to the **contrail-collector** by the physical devices. Because the **contrail-collector** supports the sFlow and IPFIX protocols for receiving flow samples, the physical devices, such as MX Series devices or ToR switches, must be configured to send samples using one of those protocols.

Example: sFlow Configuration

The following shows a sample sFlow configuration. In the sample, the IP variable *<source ip>* refers to the loopback or IP that can be reachable of the device that acts as an sflow source, and the other IP variable *<collector_IP_data>* is the address of the collector device.

```
root@host> show configuration protocols sflow | display set

set protocols sflow polling-interval 0

set protocols sflow sample-rate ingress 10

set protocols sflow source-ip <source ip>4

set protocols sflow collector <collector_IP_data> udp-port 6343

set protocols sflow interfaces ge-0/0/0.0

set protocols sflow interfaces ge-0/0/1.0

set protocols sflow interfaces ge-0/0/2.0

set protocols sflow interfaces ge-0/0/3.0

set protocols sflow interfaces ge-0/0/4.0
```

Example: IPFIX Configuration

The following is a sample IPFIX configuration from a Juniper Networks device. The IP address variable `<ip_sflow collector>` represents the sflow collector (control-collector analytics node) and `<source ip>` represents the source (outgoing) interface on the router/switch device used for sending flow data to the collector. This could also be the lo0 address, if it's reachable from the Contrail cluster.

```

root@host> show configuration chassis | display set

set chassis tfeb slot 0 sampling-instance sample-ins1

set chassis network-services

root@host> show configuration chassis tfeb | display set

set chassis tfeb slot 0 sampling-instance sample-ins1


root@host > show configuration services flow-monitoring | display set

set services flow-monitoring version-ipfix template t1 flow-active-timeout
30

set services flow-monitoring version-ipfix template t1 flow-inactive-timeout
30

set services flow-monitoring version-ipfix template t1 template-refresh-rate
packets 10

set services flow-monitoring version-ipfix template t1 ipv4-template


root@host > show configuration interfaces | display set | match sampling

set interfaces ge-1/0/0 unit 0 family inet sampling input

set interfaces ge-1/0/1 unit 0 family inet sampling input

```

```

root@host> show configuration forwarding-options sampling | display set

set forwarding-options sampling instance sample-ins1 input rate 1

set forwarding-options sampling instance sample-ins1 family inet output
flow-server <ip_sflow collector> port 4739

set forwarding-options sampling instance sample-ins1 family inet output
flow-server <ip_sflow collector> version-ipfix template t1

set forwarding-options sampling instance sample-ins1 family inet output
inline-jflow source-address <source ip>

```

Sending pRouter Information to the SNMP Collector in Contrail

Information about the physical devices must be sent to the SNMP collector before the full analytics information can be read and displayed. Typically, the pRouter information is taken from the **contrail-config** file.

SNMP collector getting pRouter information from contrail-config file

The physical routers are added to the **contrail-config** by using the Contrail user interface or by using direct API, by means of provisioning or other scripts. Once the configuration is in the **contrail-config**, the **contrail-snmp-collector** gets the physical router information from **contrail-config**. The SNMP collector uses this list and the other configuration parameters to perform SNMP queries and to populate pRouter UVEs.

Figure 26: Add Physical Router Window

The screenshot shows the Juniper Network Configuration GUI. On the left is a sidebar with navigation options: Infrastructure, Physical Devices, Physical Routers, Interfaces, Networking, Services, and DNS. The main area displays the 'Physical Routers' configuration page with a list of routers: a7-ex2, a7-ex3, a7-mx80-1, and a7-mx80-2. An 'Add Physical Router' dialog box is open in the foreground. The dialog contains the following fields and sections:

- Name:** new-prouter
- Vendor:** (empty field)
- Model:** (empty field)
- Management IP:** 1.1.1.1
- Tunnel Source IP:** (empty field)
- User Credentials:** (expandable section)
- Virtual Router:** (expandable section)
- BGP Router:** (expandable section)
- SNMP Credentials:** (expandable section)
 - Version:** 2 (selected), 3
 - Community:** public

At the bottom right of the dialog are 'Cancel' and 'Save' buttons. A small identifier 'S042440' is visible on the right edge of the dialog box.

pRouter UVEs

pRouter UVEs are accessed from the REST APIs on your system from **contrail-analytics-api**, using a URL of the form:

http://<host ip>:8081/analytics/uves/prouters

The following is sample output from a pRouter REST API:

Figure 27: Sample Output From a pRouter REST API

```
[
  {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-1?flat",
    name: "a7-mx80-1"
  },
  {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-mx80-2?flat",
    name: "a7-mx80-2"
  },
  {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex3?flat",
    name: "a7-ex3"
  },
  {
    href: "http://10.84.63.130:8081/analytics/uves/prouter/a7-ex2?flat",
    name: "a7-ex2"
  }
]
```

A small identifier 's042104' is visible on the right side of the output.

Details of a pRouter UVE can be obtained from your system, using a URL of the following form:

http://<host ip>:8081/analytics/uves/prouter/a7-ex3?flat

The following is sample output of a pRouter UVE.

Figure 28: Sample Output From a pRouter UVE

```

{
  - PRouterFlowEntry: {
    flow_export_source_ip: "10.84.63.114"
  },
  - PRouterLinkEntry: {
    - link_table: [
      - {
        remote_interface_name: "ge-1/0/1",
        local_interface_name: "ge-0/0/0.0",
        remote_interface_index: 517,
        local_interface_index: 503,
        type: 1,
        remote_system_name: "a7-mx80-1"
      },
      - {
        remote_interface_name: "ge-1/0/1",
        local_interface_name: "ge-0/0/1.0",
        remote_interface_index: 517,
        local_interface_index: 505,
        type: 1,
        remote_system_name: "a7-mx80-2"
      },
      - {
        remote_interface_name: "eth1",
        local_interface_name: "ge-0/0/2.0",
        remote_interface_index: 1,
        local_interface_index: 507,
        type: 2,
        remote_system_name: "a7s35"
      },
      - {
        remote_interface_name: "eth1",
        local_interface_name: "ge-0/0/3.0",
        remote_interface_index: 1,
        local_interface_index: 509,
        type: 2,
        remote_system_name: "a7s36"
      }
    ]
  },
  - PRouterEntry: {
    + ipMib: [...],
    + ifTable: [...],
    + ifXTable: [...],
    + arpTable: [...],
    + lldpTable: {...},
    + ifStats: {...}
  }
}

```

s042435

Contrail User Interface for Underlay Overlay Analytics

The topology view and related functionality is accessed from the Contrail Web user interface, **Monitor > Physical Topology**.

Enabling Physical Topology on the Web UI

To enable the **Physical Topology** section in the Contrail Web UI:

1. Add the following lines to the `/etc/contrail/config.global.js` file of all the **contrail-webui** nodes:

```

config.optFeatureList = {};
config.optFeatureList.mon_infra_underlay = true;

```


- Restart webui supervisor.

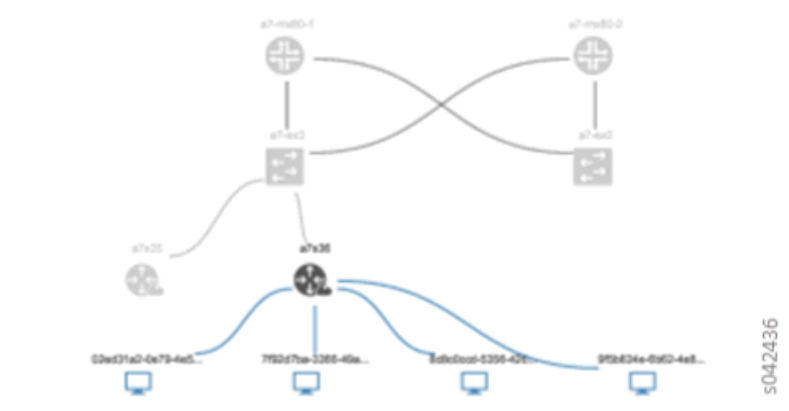
service supervisor-webui restart

The **Physical Topology** section is now available on the Contrail Web UI.

Viewing Topology to the Virtual Machine Level

In the Contrail user interface, it is possible to drill down through displayed topology to the virtual machine level. The following diagram shows the virtual machines instantiated on a7s36 vRouter and the full physical topology related to each.

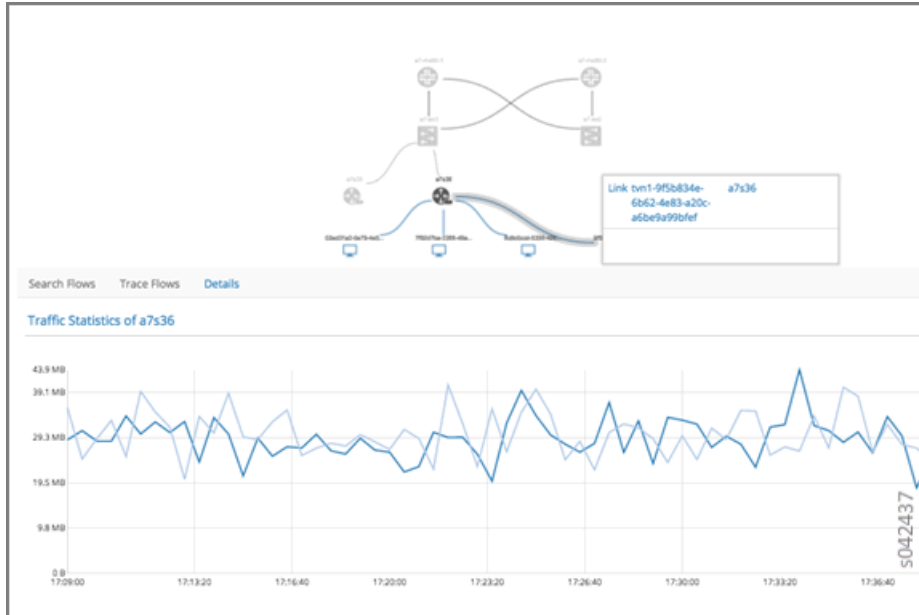
Figure 29: Physical Topology Related to a vRouter



Viewing the Traffic of any Link

At **Monitor > Physical Topology**, double click any link on the topology to display the traffic statistics graph for that link. The following is an example.

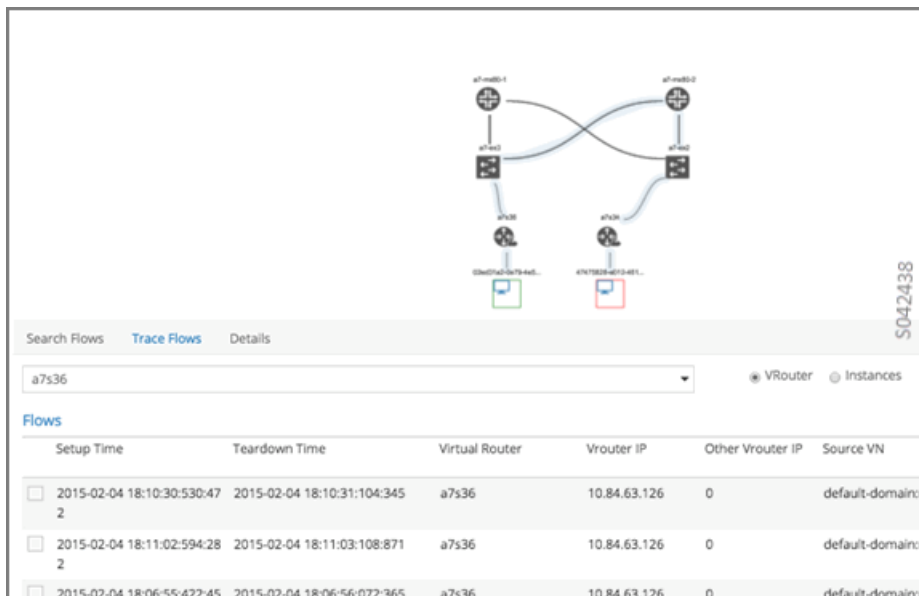
Figure 30: Traffic Statistics Graph



Trace Flows

Click the **Trace Flows** tab to see a list of active flows. To see the path of a flow, click a flow in the active flows list, then click the **Trace Flow** button. The path taken in the underlay by the selected flow displays. The following is an example.

Figure 31: List of Active Flows



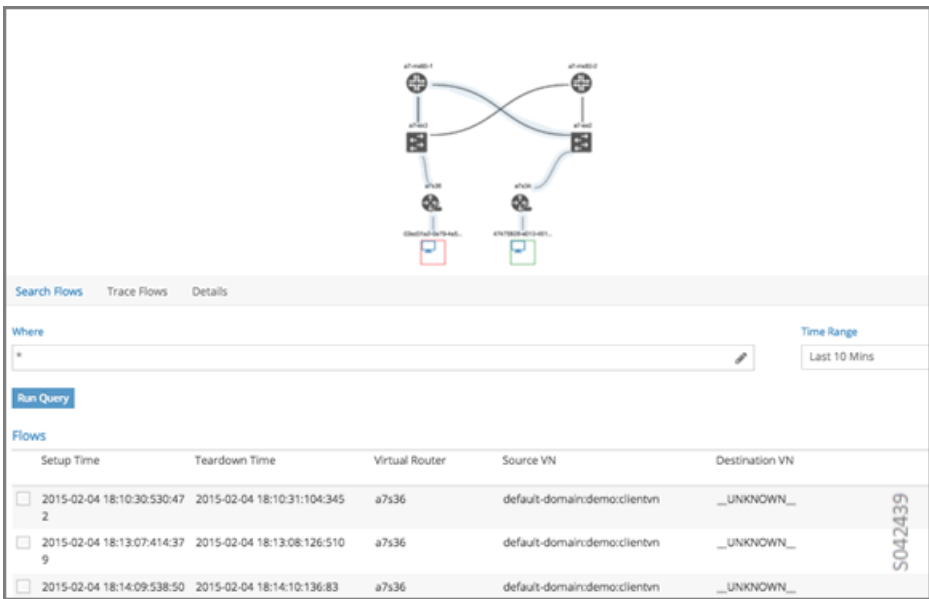
Limitations of Trace Flow Feature

Because the Trace Flow feature uses ip traceroute to determine the path between the two vRouters involved in the flow, it has the same limitations as the ip traceroute, including that Layer 2 routers in the path are not listed, and therefore do not appear in the topology.

Search Flows and Map Flows

Click the **Search Flows** tab to open a search dialog, then click the **Search** button to list the flows that match the search criteria. You can select a flow from the list and click **Map Flow** to display the underlay path taken by the selected flow in the topology. The following is an example.

Figure 32: Underlay Path



Overlay to Underlay Flow Map Schemas

The schema to query the underlay mapping information for an overlay flow is obtained from a REST API, which can be accessed on your system using a URL of the following form:

http://<host ip>:8081/analytics/table/OverlayToUnderlayFlowMap/schema

Example: Overlay to Underlay Flow Map Schema

```
{ "type": "FLOW",
```

```

"columns": [

{"datatype": "string", "index": true, "name": "o_svn", "select": false,
"suffixes": ["o_sip"]},

{"datatype": "string", "index": false, "name": "o_sip", "select": false,
"suffixes": null},

{"datatype": "string", "index": true, "name": "o_dvn", "select": false,
"suffixes": ["o_dip"]},

{"datatype": "string", "index": false, "name": "o_dip", "select": false,
"suffixes": null},

{"datatype": "int", "index": false, "name": "o_sport", "select": false,
"suffixes": null},

{"datatype": "int", "index": false, "name": "o_dport", "select": false,
"suffixes": null},

{"datatype": "int", "index": true, "name": "o_protocol", "select": false,
"suffixes": ["o_sport", "o_dport"]},

{"datatype": "string", "index": true, "name": "o_vrouter", "select": false,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_prouter", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_pifindex", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_vlan", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_sip", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_dip", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_sport", "select": null,

```

```

"suffixes": null},

{"datatype": "int", "index": false, "name": "u_dport", "select": null,
"suffixes": null},

{"datatype": "int", "index": false, "name": "u_protocol", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_flowtype", "select": null,
"suffixes": null},

{"datatype": "string", "index": false, "name": "u_otherinfo", "select": null,
"suffixes": null}]]}

```

The schema for underlay data across pRouters is defined in the Contrail installation at:

<http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema>

Example: Flow Data Schema for Underlay

```

{"type": "STAT",

"columns": [

{"datatype": "string", "index": true, "name": "Source", "suffixes": null},

{"datatype": "int", "index": false, "name": "T", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(T)", "suffixes": null},

{"datatype": "int", "index": false, "name": "T=", "suffixes": null},

{"datatype": "int", "index": false, "name": "CLASS(T=)", "suffixes": null},

{"datatype": "uuid", "index": false, "name": "UUID", "suffixes": null},

{"datatype": "int", "index": false, "name": "COUNT(flow)", "suffixes": null},

{"datatype": "string", "index": true, "name": "name", "suffixes":

```

```

["flow.pifindex"]},

{"datatype": "int", "index": false, "name": "flow.pifindex", "suffixes":
null},

{"datatype": "int", "index": false, "name": "SUM(flow.pifindex)", "suffixes":
null},

{"datatype": "int", "index": false, "name": "CLASS(flow.pifindex)", "suffixes":
null},

{"datatype": "int", "index": false, "name": "flow.sport", "suffixes": null},

{"datatype": "int", "index": false, "name": "SUM(flow.sport)", "suffixes":
null},

{"datatype": "int", "index": false, "name": "CLASS(flow.sport)", "suffixes":
null},

{"datatype": "int", "index": false, "name": "flow.dport", "suffixes": null},

{"datatype": "int", "index": false, "name": "SUM(flow.dport)", "suffixes":
null},

{"datatype": "int", "index": false, "name": "CLASS(flow.dport)", "suffixes":
null},

{"datatype": "int", "index": true, "name": "flow.protocol", "suffixes":
["flow.sport", "flow.dport"]},

{"datatype": "int", "index": false, "name": "SUM(flow.protocol)", "suffixes":
null},

{"datatype": "int", "index": false, "name": "CLASS(flow.protocol)", "suffixes":
null},

{"datatype": "string", "index": true, "name": "flow.sip", "suffixes": null},

{"datatype": "string", "index": true, "name": "flow.dip", "suffixes": null},

{"datatype": "string", "index": true, "name": "flow.vlan", "suffixes": null},

```

```
{
  "datatype": "string", "index": false, "name": "flow.flowtype", "suffixes": null},
  {
    "datatype": "string", "index": false, "name": "flow.otherinfo", "suffixes": null}]]}
```

Example: Typical Query for Flow Map

The following is a typical query. Internally, the **analytics-api** performs a query into the **FlowRecordTable**, then into the **StatTable.UFlowData.flow**, to return list of (**prouter**, **pifindex**) pairs that give the underlay path taken for the given overlay flow.

```
FROM

OverlayToUnderlayFlowMap

SELECT

prouter, pifindex

WHERE

o_svn, o_sip, o_dvn, o_dip, o_sport, o_dport, o_protocol = <overlay flow>
```

Module Operations for Overlay Underlay Mapping

SNMP Collector Operation

The Contrail SNMP collector uses a Net-SNMP library to talk to a physical router or any SNMP agent. Upon receiving SNMP packets, the data is translated to the Python dictionary, and corresponding UVE objects are created. The UVE objects are then posted to the SNMP collector.

The SNMP module sleeps for some configurable period, then forks a collector process and waits for the process to complete. The collector process goes through a list of devices to be queried. For each device, it forks a greenlet task (Python coroutine), accumulates SNMP data, writes the summary to a JSON file, and exits. The parent process then reads the JSON file, creates UVEs, sends the UVEs to the collector, then goes to sleep again.

The pRouter UVE sent by the SNMP collector carries only the raw MIB information.

Example: pRouter Entry Carried in pRouter UVE

The definition below shows the **pRouterEntry** carried in the **pRouterUVE**. Additionally, an example **LldpTable** definition is shown.

The following create a virtual table as defined by:

```
http://<host ip>:8081/analytics/table/StatTable.UFlowData.flow/schema

struct LldpTable {

    1: LldpLocalSystemData lldpLocalSystemData

    2: optional list<LldpRemoteSystemsData> lldpRemoteSystemsData

}

struct PRouterEntry {

    1: string name (key="ObjectPRouter")

    2: optional bool deleted

    3: optional LldpTable lldpTable

    4: optional list<ArpTable> arpTable

    5: optional list<IfTable> ifTable

    6: optional list<IfXTable> ifXTable

    7: optional list<IfStats> ifStats (tags="name:.ifIndex")

    8: optional list<IpMib> ipMib

}

uve sandesh PRouterUVE {

    1: PRouterEntry data
```



```
}
```

Topology Module Operation

The topology module reads UVEs posted by the SNMP collector and computes the neighbor table, populating the table with remote system name, local and remote interface names, the remote type (pRouter or vRouter) and local and remote ifindices. The topology module sleeps for a while, reads UVEs, then computes the neighbor table and posts the UVE to the collector.

The pRouter UVE sent by the topology module carries the neighbor list, so the clients can put together all of the pRouter neighbor lists to compute the full topology.

The corresponding pRouter UVE definition is the following.

```
struct LinkEntry {
    1: string remote_system_name
    2: string local_interface_name
    3: string remote_interface_name
    4: RemoteType type
    5: i32 local_interface_index
    6: i32 remote_interface_index
}

struct PRouterLinkEntry {
    1: string name (key="ObjectPRouter")
    2: optional bool deleted
    3: optional list<LinkEntry> link_table
}
```

```
uve sandesh PRouterLinkUVE {

    1: PRouterLinkEntry data

}
```

IPFIX and sFlow Collector Operation

An IPFIX and sFlow collector has been implemented in the Contrail collector. The collector receives the IPFIX and sFlow samples and stores them as statistics samples in the analytics database.

Example: IPFIX sFlow Collector Data

The following definition shows the data stored for the statistics samples and the indices that can be used to perform queries.

```
struct UFlowSample {

    1: u64 pifindex

    2: string sip

    3: string dip

    4: u16 sport

    5: u16 dport

    6: u16 protocol

    7: u16 vlan

    8: string flowtype

    9: string otherinfo

}
```

```

struct UFlowData {

    1: string name (key="ObjectPRouterIP")

    2: optional bool deleted

    3: optional list<UFlowSample> flow (tags="name:.pifindex, .sip, .dip,
.protocol:.sport, .protocol:.dport, .vlan")

}

```

Troubleshooting Underlay Overlay Mapping

This section provides a variety of links where you can research errors that may occur with underlay overlay mapping.

System Logs

Logs for **contrail-snmp-collector** and **contrail-topology** are in the following locations on an installed Contrail system:

/var/log/contrail/contrail-snmp-collector-stdout.log

/var/log/contrail/contrail-topology.log

Introspect Utility

Use URLs of the following forms on your Contrail system to access the introspect utilities for SNMP data and for topology data.

- SNMP data introspect

`http://<host ip>:5920/Snh_SandeshUVECacheReq?x=PRouterEntry`

- Topology data introspect

`http://<host ip>:5921/Snh_SandeshUVECacheReq?x=PRouterLinkEntry`

Script to add pRouter Objects

The usual mechanism for adding pRouter objects to **contrail-config** is through Contrail UI. But you also have the ability to add these objects using the Contrail **vnc-api**. To add one pRouter, save the file with the name **cfg-snmp.py**, and then execute the command as shown:

python cfg-snmp.py

Example: Content for cfg-snmp.py

```
#!/python

from vnc_api import vnc_api

from vnc_api.gen.resource_xsd import SNMPCredentials

vnc = vnc_api.VncApi('admin', 'abcde123', 'admin')

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-1')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))
```

```

vnc.physical_router_create(apr)

#$ABC123

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-mx80-2')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex3')

apr.set_physical_router_management_ip('source_ip')

apr.set_physical_router_dataplane_ip('source_ip')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

apr = vnc_api.gen.resource_client.PhysicalRouter(name='a7-ex2')

apr.set_physical_router_management_ip('ip_address')

apr.set_physical_router_dataplane_ip('ip_address')

apr.set_physical_router_snmp_credentials(SNMPCredentials(version=2,
v2_community='public'))

vnc.physical_router_create(apr)

#$ABC123'

```

RELATED DOCUMENTATION

[Understanding Contrail Analytics](#) | 97

[Contrail Alerts](#) | 98

Configuring Contrail Analytics

IN THIS CHAPTER

- [Analytics Scalability | 127](#)
- [High Availability for Analytics | 128](#)
- [System Log Receiver in Contrail Analytics | 129](#)
- [Sending Flow Messages to the Contrail System Log | 130](#)
- [User Configuration for Analytics Alarms and Log Statistics | 131](#)
- [Alarms History | 141](#)
- [Node Memory and CPU Information | 143](#)
- [Role- and Resource-Based Access Control for the Contrail Analytics API | 144](#)
- [Configuring Analytics as a Standalone Solution | 145](#)
- [Configuring Secure Sandesh and Introspect for Contrail Analytics | 149](#)

Analytics Scalability

The Contrail monitoring and analytics services (*collector* role) collect and store data generated by various system components and provide the data to the Contrail interface by means of representational state transfer (REST) application program interface (API) queries.

The Contrail components are horizontally scalable to ensure consistent performance as the system grows. Scalability is provided for the generator components (*control* and *compute* roles) and for the REST API users (*webui* role).

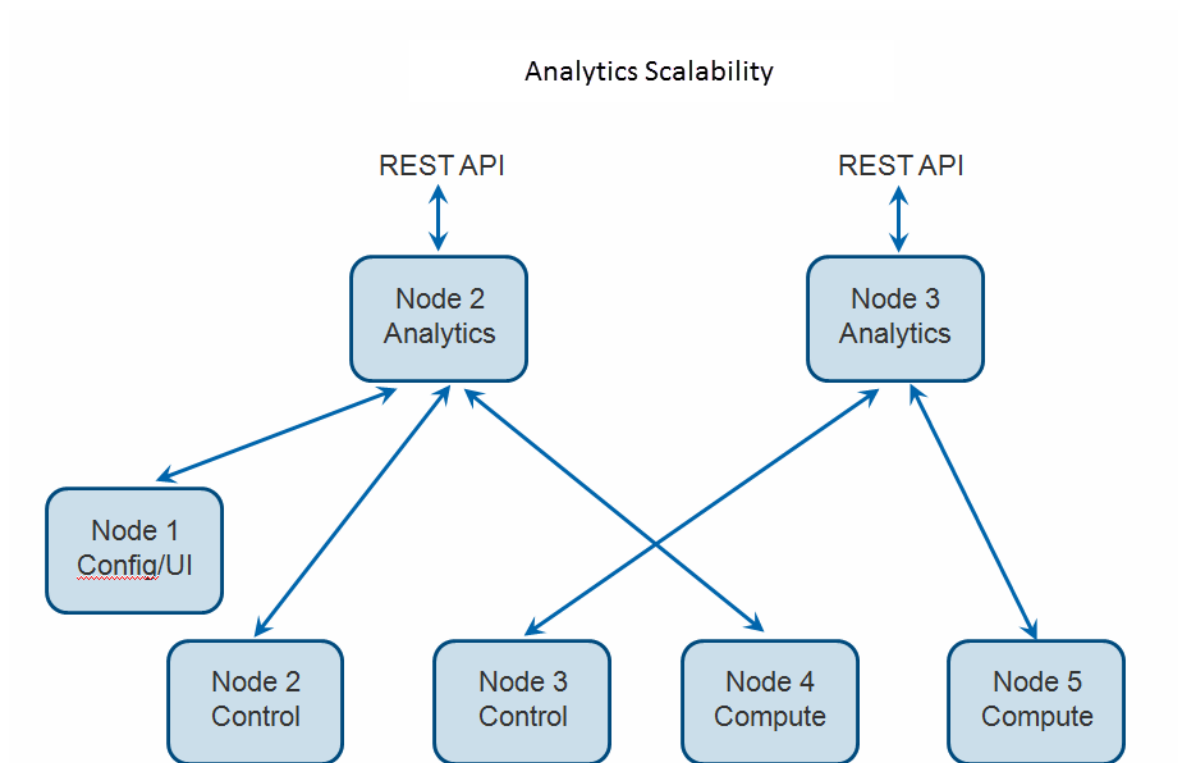
This section provides a brief description of the recommended configuration of analytics in Contrail to achieve horizontal scalability.

The following is the recommended locations for the various component roles of the Contrail system for a 5-node configuration.

- Node 1 —config role, web-ui role
- Node 2 —control role, analytics role, database role
- Node 3 —control role, analytics role, database role
- Node 4 —compute role
- Node 5 —compute role

[Figure 33 on page 128](#) illustrates scalable connections for analytics in a 5-node system, with the nodes configured for roles as recommended above. The analytics load is distributed between the two analytics nodes. This configuration can be extended to any number of analytics nodes.

Figure 33: Analytics Scalability



The analytics nodes collect and store data and provide this data through various REST API queries. Scalability is provided for the control nodes, the compute nodes, and the REST API users, with the API output displayed in the Contrail user interface. As the number of control and compute nodes increase in the system, the analytics nodes can also be increased.

High Availability for Analytics

Contrail supports multiple instances of analytics for high availability and load balancing.

Contrail analytics provides two broad areas of functionality:

- **contrail-collector** —Receives status, logs, and flow information from all Contrail processing elements (for example, generators) and records them.

Every generator is connected to one of the **contrail-collector** instances at any given time. If an instance fails (or is shut down), all the generators that are connected to it are automatically moved to another functioning instance, typically in a few seconds or less. Some messages may be lost during this movement. UVEs are resilient to message loss, so the state shown in a UVE is kept consistent to the state in the generator.

- **contrail-opserver** —Provides an external API to report UVEs and to query logs and flows.

Each analytics component exposes a northbound REST API represented by the **contrail-opserver** service (port 8081) so that the failure of one analytics component or one **contrail-opserver** service should not impact the operation of other instances.

These are the ways to manage connectivity to the **contrail-opserver** endpoints:

- Periodically poll the **contrail-opserver** service on a set of analytics nodes to determine the list of functioning endpoints, then make API requests from one or more of the functioning endpoints.
- The Contrail user interface makes use of the same northbound REST API to present dashboards, and reacts to any **contrail-opserver** high availability event automatically.

System Log Receiver in Contrail Analytics

IN THIS SECTION

- [Overview | 129](#)
- [Redirecting System Logs to Contrail Collector | 129](#)
- [Exporting Logs from Contrail Analytics | 130](#)

Overview

The **contrail-collector** process on the Contrail Analytics node can act as a system log receiver.

Redirecting System Logs to Contrail Collector

You can enable the **contrail-collector** to receive system logs by giving a valid **syslog_port** as a command line option:

--DEFAULT.syslog_port <arg>

or by adding **syslog_port** in the **DEFAULT** section of the configuration file at **/etc/contrail/contrail-collector.conf**.

For nodes to send system logs to the **contrail-collector**, the system log configuration for the node should be set up to direct the system logs to **contrail-collector**.

Example

Add the following line in `/etc/rsyslog.d/50-default.conf` on an Ubuntu system to redirect the system logs to contrail-collector.

```
*.* @<collector_ip>:<collector_syslog_port> :: @ for udp, @@ for tcp
```

The logs can be retrieved by using Contrail tool, either by using the `contrail-logs` utility on the analytics node or by using the Contrail user interface on the system log query page.

Exporting Logs from Contrail Analytics

You can also export logs stored in Contrail analytics to another system log receiver by using the **contrail-logs** utility.

The `contrail-logs` utility can take these options: `--send-syslog`, `--syslog-server`, `--syslog-port`, to query Contrail analytics, then send the results as system logs to a system log server. This is an on-demand command, one can write a cron job or a job that continuously invokes **contrail-logs** to achieve continuous sending of logs to another system log server.

Sending Flow Messages to the Contrail System Log

The **contrail-vrouter-agent** can be configured to send flow messages and other messages to the system log (syslog). To send flow messages to syslog, configure the following parameters in `/etc/contrail/contrail-vrouter-agent.conf`.

The following parameters are under the section **DEFAULT**:

- **log_flow=1**—Enables logging of all flow messages.
- **use_syslog=1**—Enables sending of all messages, including flow messages, to syslog.
- **syslog_facility=LOG_LOCAL0**—Enables sending messages from the **contrail-vrouter-agent** to the syslog, using the facility **LOCAL0**. You can configure **LOCAL0** to your required facility.
- **log_level=SYS_INFO**—Changes the logging level of **contrail-vrouter-agent** to INFO.

If syslog is enabled, flow messages are *not* sent to Contrail Analytics because the two destinations are mutually exclusive.

Flow log sampling settings apply regardless of the flow log destination specified. If sampling is enabled, the syslog messages will be sampled using the same rules that would apply to Contrail Analytics. If non-sampled flow data is required, sampling must be disabled by means of configuration settings.

Flow events for termination will include both the appropriate tear-down fields and the appropriate setup fields.

The flow messages will be sent to the syslog with a severity of INFO.

The user can configure the remote system log (**rsyslog**) on the compute node to send syslog messages with facility LOCAL0, severity of INFO (and lower), to the remote syslog server. Messages with a higher severity than INFO can be logged to a local file to allow for debugging.

Flow messages appear in the syslog in a format similar to the following log example:

```
May 24 14:40:13 a7s10 contrail-vrouter-agent[29930]: 2016-05-24 Tue 14:40:13:921.098 PDT a7s10
[Thread 139724471654144, Pid 29930]: [SYS_INFO]: FlowLogDataObject: flowdata= [ [ [ flowuuid =
7ea8bf8f-b827-496e-b93e-7622a0c8eeea direction_ing = 1 sourcevn = default-domain:mock-gen-test:vn8
sourceip = 1.0.0.9 destvn = default-domain:mock-gen-test:vn58 destip = 1.0.0.59 protocol = 1 sport =
-29520 dport = 20315 setup_time = 1464125225556930 bytes = 1035611592 packets = 2024830
diff_bytes = 27240 diff_packets = 40 ], ] ]
```

NOTE: Several individual flow messages might be packed into a single syslog message for improved efficiency.

User Configuration for Analytics Alarms and Log Statistics

IN THIS SECTION

- [Configuring Alarms Based on User-Visible Entities Data | 131](#)
- [Examples: Detecting Anomalies | 133](#)
- [Configuring the User-Defined Log Statistic | 135](#)
- [Implementing the User-Defined Log Statistic | 138](#)

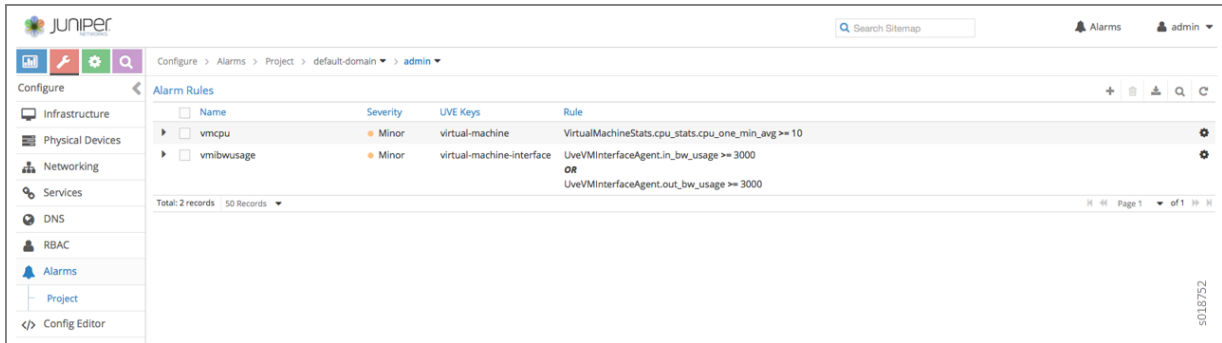
Configuring Alarms Based on User-Visible Entities Data

Starting with Contrail 3.1, users can dynamically configure alarms based on the user-visible entities (UVE) data. An alarm configuration object is created based on the alarm configuration XSD schema. The alarm configuration object is added to the Contrail configuration database, using the Contrail API server REST API interface.

An alarm configuration object can be anchored in the configuration data model under **global-system-config** or **project**, depending on the alarm type. Under **global-system-config**, you should configure virtual network system-wide alarms, such as those for the analytics node, the config node, and so on. Under **project**, you should configure alarms related to project objects, such as virtual networks and similar objects.

To configure and monitor alarms using the Contrail UI:

1. Navigate to **Configure > Alarms > Project**, and select the desired project to access the **Alarm Rules** page.



2. Click the Gear icon to add a new alarm configuration or to edit an existing alarm configuration. Use the **Edit** screen to define descriptions and to set up alarm rules. See [Table 32 on page 133](#) for field descriptions.

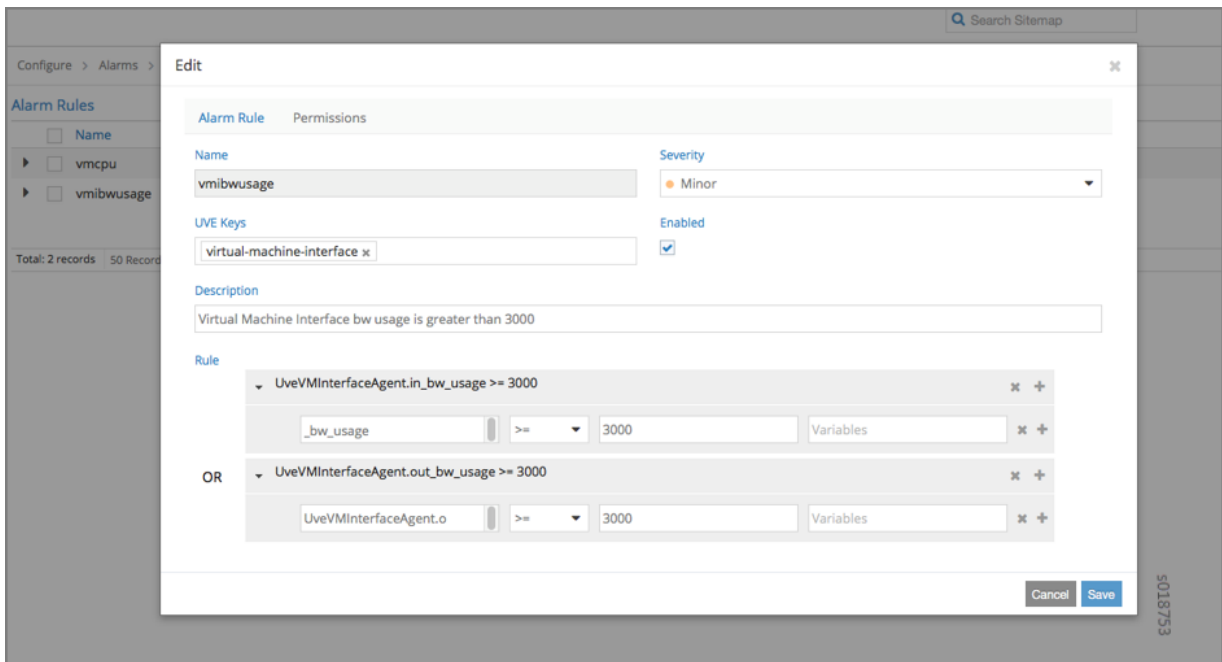


Table 32: Alarm Rules Fields

Field	Description
Name	Enter a name for the alarm.
Severity	Select the severity level of the alarm from the list.
UVE Keys	Select the list of UVE types to apply to this alarm.
Description	Enter a description of the alarm.
Rule	Set up the alarm rules. Alarm rules are expressed as OR of AND terms. Each term has operand1, operand2, and the operation. Operand1 is the UVE attribute. Operand2 can be either another UVE attribute or a JSON value. The rules are evaluated in the contrail-alarm-gen service and an alarm is raised or cleared as needed on respective conditions.

- To monitor alarms, navigate to **Monitor > Alarms > Dashboard**. The **Dashboard** screen lists the active alarms in the system.



Examples: Detecting Anomalies

The purpose of anomaly detection in Contrail is to identify a condition in which a metric deviates from its expected value, within given parameters.

Contrail uses a statistical process control model for time-series anomaly detection that can be computed online, in real-time. Raw metrics are sent as statistics by Sandesh generators embedded inside the UVEs.

The model uses the running average and running standard deviation for a given raw metric. The model does not account for seasonality and linear trends in the metric.

The following example represents part of the UVE sent by the vRouter to the collector. The raw metrics are **phy_band_in_bps** and **phy_band_out_bps**.

The derived statistics are in **in_bps_ewm** and **out_bps_ewm**, which are generated when the model's EWM algorithm is applied to the raw metrics. The raw metrics and the derived statistics are part of the UVE and are sent to the collector.

```
struct EWMResult {
    3: u64 samples
    6: double mean
    7: double stddev
}

struct VrouterStatsAgent { // Agent stats

1: string name (key="ObjectVRouter")

2: optional bool deleted    ...

/** @display_name:Vrouter Physical Interface Input bandwidth Statistics*/

50: optional map<string,u64> phy_band_in_bps (tags="name:.__key")

/** @display_name:Vrouter Physical Interface Output bandwidth Statistics*/

51: optional map<string,u64> phy_band_out_bps (tags="name:.__key")

52: optional map<string,derived_stats_results.EWMResult> in_bps_ewm
(mstats="phy_band_in_bps:DSEWM:0.2")

53: optional map<string,derived_stats_results.EWMResult> out_bps_ewm
(mstats="phy_band_out_bps:DSEWM:0.2")
}
```

The following shows part of the UVE that lists the raw metric **phy_band_out_bps** and the derived statistic **out_bps_ewm**. The user can define an alarm based on the values in **sigma** or in **stddev**.

```
- out_bps_ewm: {
  - eth0: {
    sigma: -0.425095,
    samples: 177,
    stddev: 6348.16,
    mean: 206712
  }
},
- phy_band_out_bps: {
  eth0: "204013"
},
```

s018755

Configuring the User-Defined Log Statistic

Any deployment of Contrail cloud over an orchestration system requires tools for monitoring and troubleshooting the entire cloud deployment. Cloud data centers are built with a large collection of interconnected servers that provide computing and storage capacity for a variety of applications. The monitoring of the cloud and its infrastructure requires monitoring logs and messages sent to a variety of servers from many micro services.

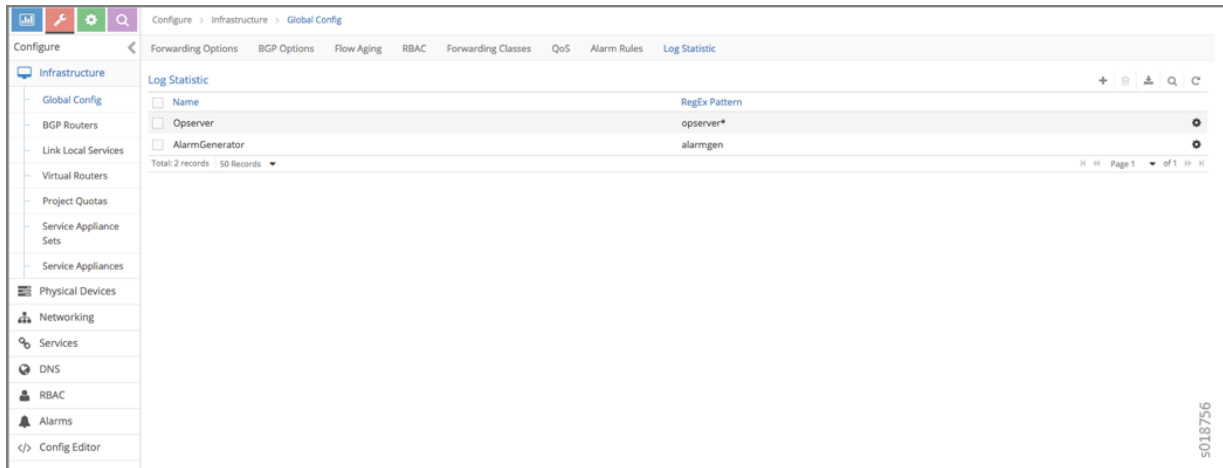
Contrail analytics stores all of the monitored messages in the Contrail database node, and the analytics generates a large amount of useful information that aids in monitoring and troubleshooting the network.

Starting with Contrail Release 3.1, the user-defined log statistic feature provides additional abilities for monitoring and troubleshooting by enabling the user to set a counter on any regular Perl-type expression. Each time the pattern is found in any system logs, UVEs, or object logs, the counter is incremented.

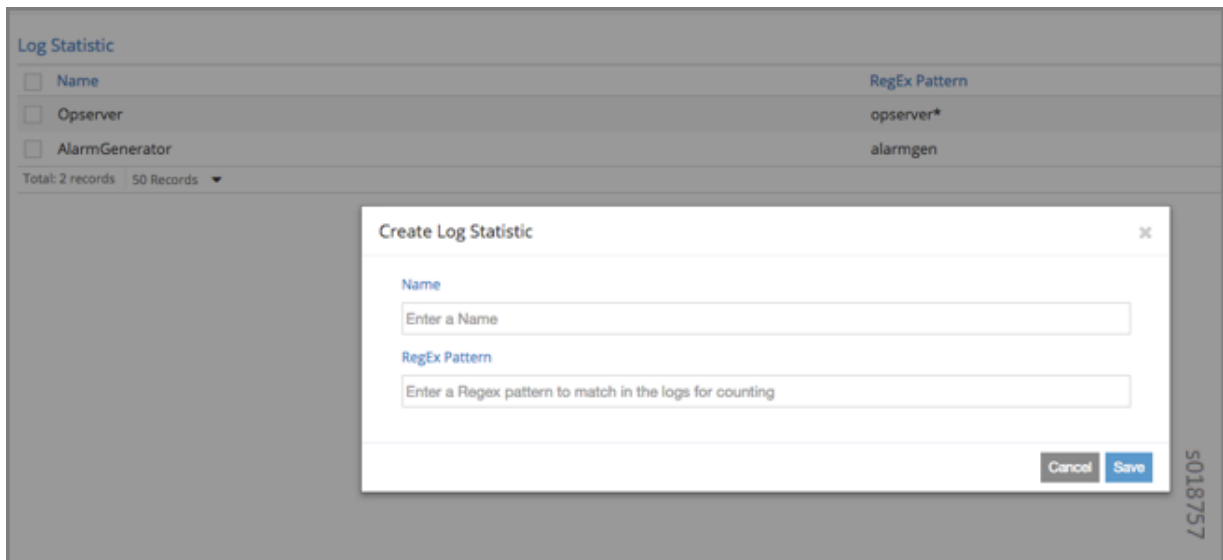
The user-defined log statistic can be configured from the Contrail UI or from the command line, using `vnc_api`.

To configure the user-defined log statistic from the Contrail UI:

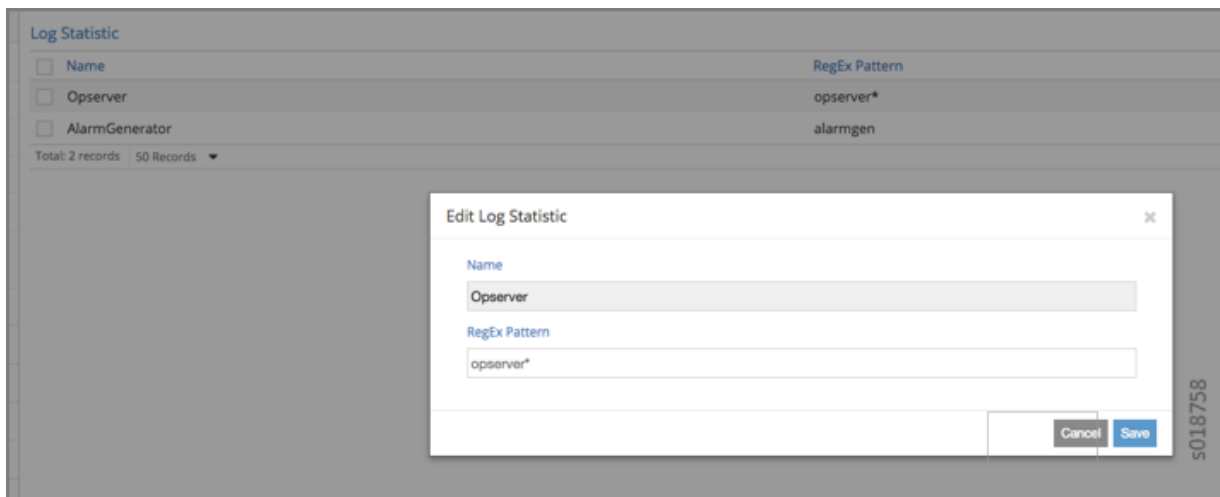
1. Navigate to **Configure > Infrastructure > Global Config** and select **Log Statistic**.



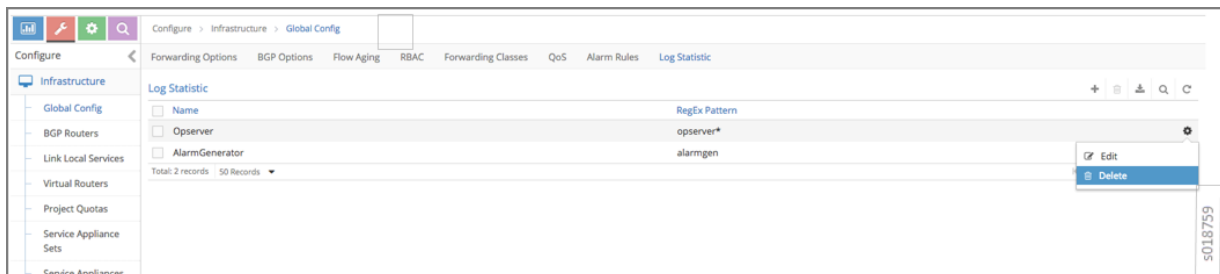
2. To create a log statistic, click the plus (+) icon to access the **Create Log Statistic** screen. Enter a name for the user-defined log statistic, and in the **RegEx Pattern** field, enter the Perl-type expression to look for and count.



- To edit an existing log statistic, select the name of the statistic and click the Gear icon, then select **Edit** to access the **Edit Log Statistic** screen.



- To delete a log statistic, select the name of the statistic and click the gear icon, then select the **Delete** option.



To configure the user-defined statistic from the **vnc_api**:

```
user@host:~# python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>> from vnc_api import vnc_api
>> from vnc_api.gen.resource_xsd import UserDefinedLogStat
>> from vnc_api.gen.resource_client import GlobalSystemConfig
>> vnc = vnc_api.VncApi('<username>', '<password>', '<tenant>')
>> gsc_uuid = vnc.global_system_configs_list()['global-system-configs'][0]['uuid']
>> gsc = vnc.global_system_config_read(id=gsc_uuid)
```

To list the counters:

```
>> [(x.name, x.pattern) for x in gsc.user_defined_log_statistics.statlist]

[('HostnameCounter', 'dummy'), ('MyIp', '10.84.14.38')]
```

To add a counter:

```
>> g=GlobalSystemConfig()
>> g.add_user_defined_counter(UserDefinedLogStat('Foo', 'Ba.*r'))
>> vnc.global_system_config_update(g)
```

To verify an addition:

```
>> gsc = vnc.global_system_config_read(id=gsc_uuid)
>> [(x.name, x.pattern) for x in gsc.user_defined_log_statistics.statlist]

[('HostnameCounter', 'dummy'), ('MyIp', '10.84.14.38'), ('Foo', 'Ba.*r')]
```

Implementing the User-Defined Log Statistic

The statistics are sent as a counter that has been aggregated over a time period of 60 seconds.

A current sample from your system can be obtained from the UVE at:

<http://<analytics-ip>:8081/analytics/uves/user-defined-log-statistic/<name>>

You can also use the statistics table **UserDefinedLogStatTable** to get historical data with all supported aggregations such as SUM, AVG, and the like.

The schema for the table is at the following location:

<http://<ip>:8081/analytics/table/StatTable.UserDefinedCounter.count/schema>

Schema for User-Defined Statistics Table

The following is the schema for the user-defined statistic table:

```
{
  "type": "STAT",
  "columns": [
    {
      "datatype": "string",
      "index": true,
```

```

    "name": "Source",
    "suffixes": null
  },
  {
    "datatype": "int",
    "index": false,
    "name": "T",
    "suffixes": null
  },
  {
    "datatype": "int",
    "index": false,
    "name": "CLASS(T)",
    "suffixes": null
  },
  {
    "datatype": "int",
    "index": false,
    "name": "T=",
    "suffixes": null
  },
  {
    "datatype": "int",
    "index": false,
    "name": "CLASS(T=)",
    "suffixes": null
  },
  {
    "datatype": "uuid",
    "index": false,
    "name": "UUID",
    "suffixes": null
  },
  {
    "datatype": "int",
    "index": false,
    "name": "COUNT(count)",
    "suffixes": null
  },
  {
    "datatype": "int",
    "index": false,
    "name": "count.previous",
    "suffixes": null
  }

```

```

},
{
  "datatype": "int",
  "index": false,
  "name": "SUM(count.previous)",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "CLASS(count.previous)",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "MAX(count.previous)",
  "suffixes": null
},
{
  "datatype": "int",
  "index": false,
  "name": "MIN(count.previous)",
  "suffixes": null
},
{
  "datatype": "percentiles",
  "index": false,
  "name": "PERCENTILES(count.previous)",
  "suffixes": null
},
{
  "datatype": "avg",
  "index": false,
  "name": "AVG(count.previous)",
  "suffixes": null
},
{
  "datatype": "string",
  "index": true,
  "name": "name",
  "suffixes": null
}

```

```
]
}
```

Alarms History

IN THIS SECTION

- [Viewing Alarms History | 141](#)

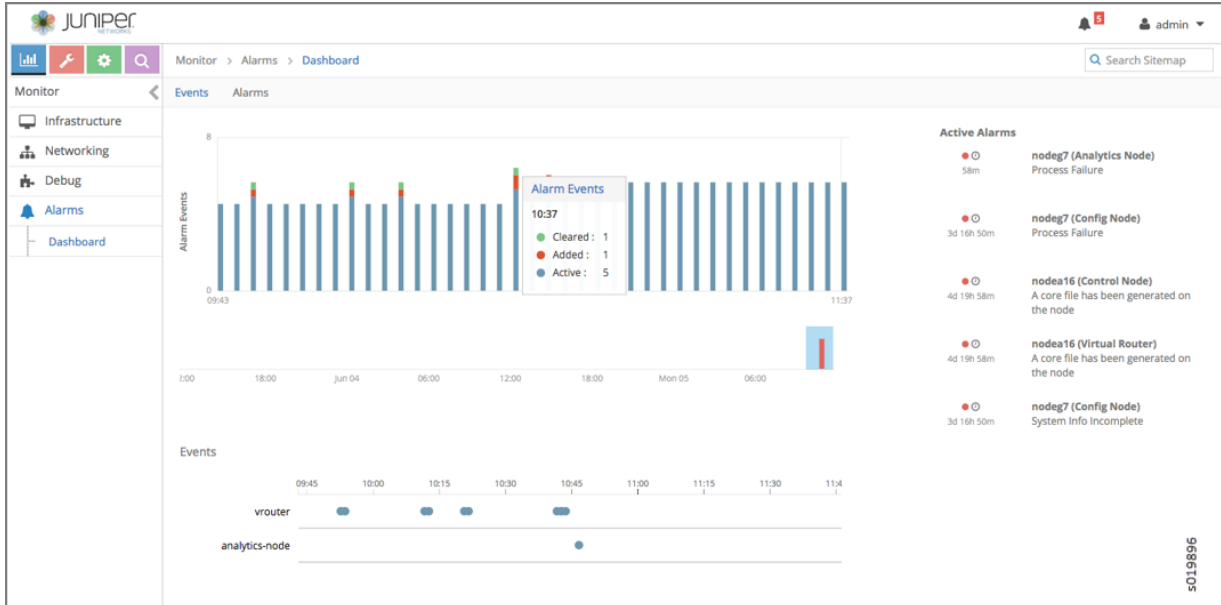
Starting with Contrail Release 4.0, you can view a history of alarms that were raised or reset. You can also view a history of user-visible entities (UVEs) that have been changed.

Viewing Alarms History

In the Contrail Web user interface, new fields at **Monitor > Alarms > Dashboard > Alarms History** now display alarms history, including alarms that were set or reset. [Figure 34 on page 142](#) shows the alarms history, identifying the volume and types of alarms by time and the node types in which events are occurring. The right side panel lists by name the nodes in which active events are occurring.

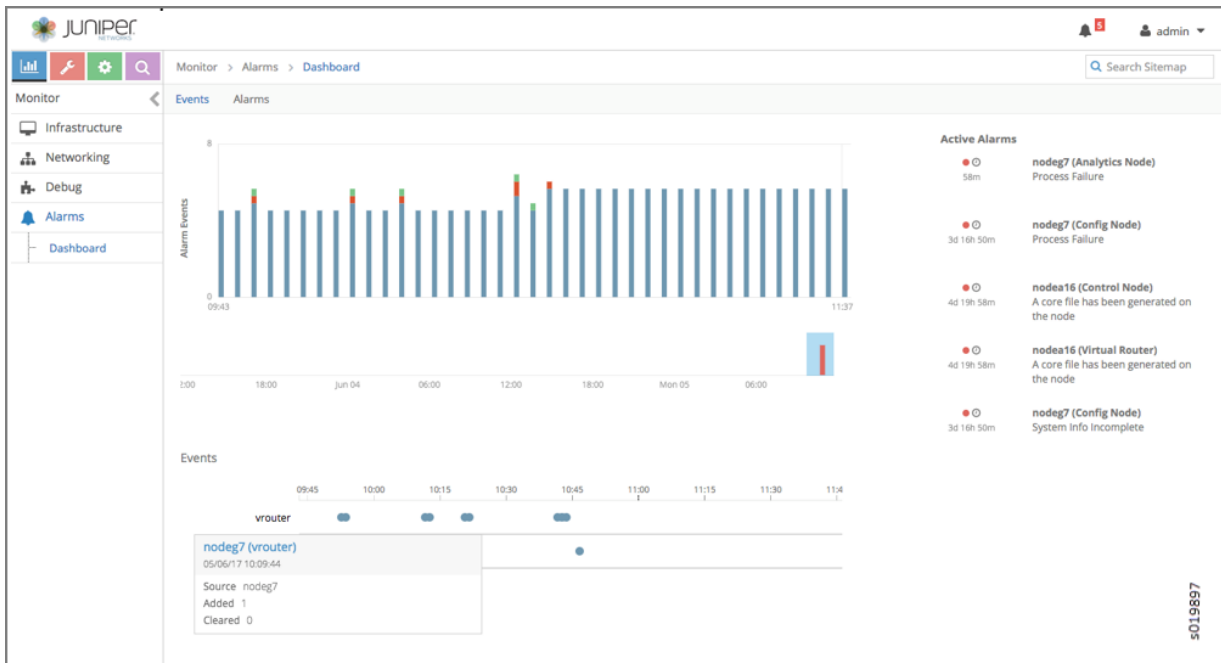
You can also use a **contrail-status** query to view the alarms history. Additionally, the **contrail-status** displays a history of added, updated, and removed information for UVEs in Contrail.

Figure 34: Alarms History Page



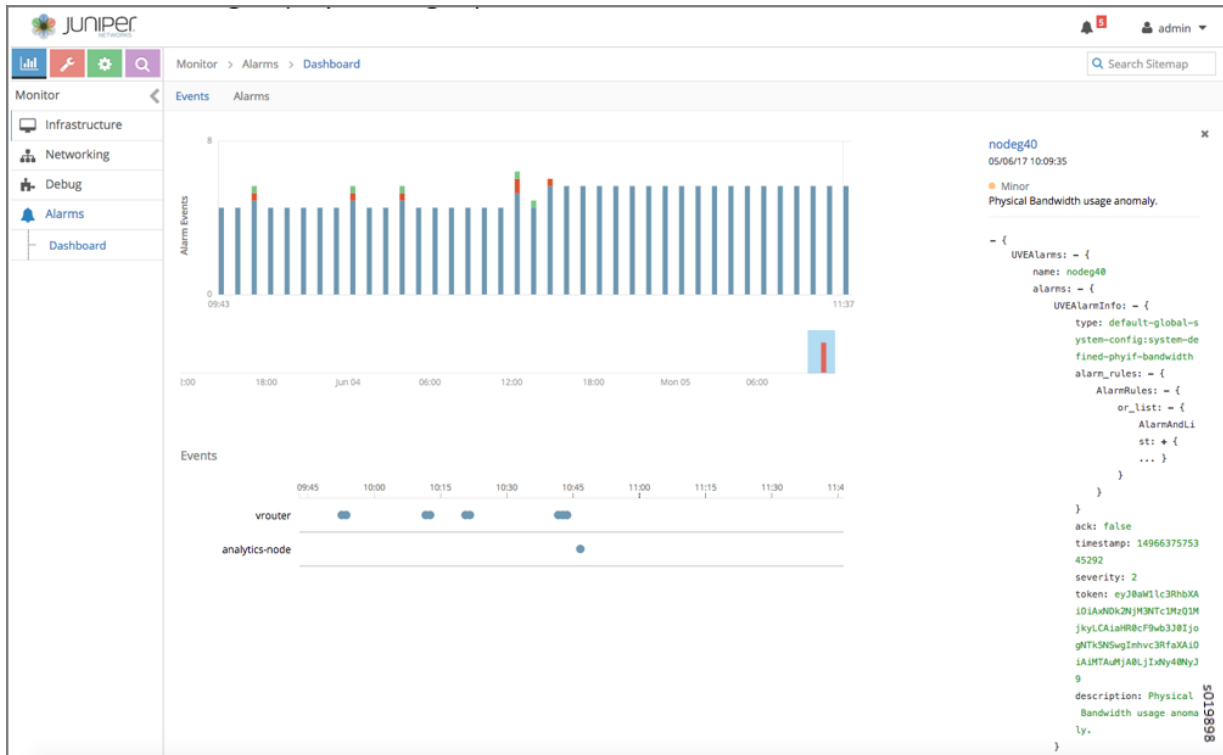
Tooltips are available on the Alarms History page. In the Events area, you can click on any node type listed to display a tooltip showing details of the events that have been added and cleared in that node, see [Figure 35 on page 142](#).

Figure 35: Events Log Tooltip



You can expand the event log in the right side panel to display a detailed event log. Click the name of any node in the list in the right panel, and the details of the current alarms are visible in the expanded view, see [Figure 36 on page 143](#).

Figure 36: Detailed Event Log



RELATED DOCUMENTATION

[User Configuration for Analytics Alarms and Log Statistics](#) | 131

Node Memory and CPU Information

To help in monitoring and debugging, the following statistics have been added for all node types. The statistics are updated every 60 seconds.

- System CPU info
- System memory and CPU usage
- Memory and CPU usage of all processes

You can see a current sample from the UVE in your system at:

`http://<analytics-ip>:8081/analytics/uves/<node-type>/<hostname>?flat`

You can also use the statistics tables to get historical data with all supported aggregations, such as SUM, AVG, and so on:

- **`NodeStatus.process_mem_cpu_usage`**
- **`NodeStatus.system_mem_cpu_usage`**

The schema for the tables are at the following locations on your system:

`http://<analytics-ip>:8081/analytics/table/StatTable.NodeStatus.process_mem_cpu_usage/schema`

`http://<analytics-ip>:8081/analytics/table/StatTable.NodeStatus.system_mem_cpu_usage/schema`

RELATED DOCUMENTATION

[User Configuration for Analytics Alarms and Log Statistics](#) | 131

Role- and Resource-Based Access Control for the Contrail Analytics API

In previous releases of Contrail, any user can access the Contrail analytics API by using queries to get historical information and by using UVEs to get state information.

Starting with Contrail Release 3.1, it is possible to restrict access such that only the **cloud-admin** user can access the Contrail analytics API.

Implementation details are as follows:

- An external user makes a REST API call to **contrail-analytics-api**, passing a token representing the user with the HTTP header **X-Auth-Token**.
- Based on the user role, **contrail-analytics-api** will only allow access for the **cloud-admin** user and reject the request (**HTTPUnauthorized**) for other users.

To set the **cloud_admin** user, use the following fields in **/etc/contrail/contrail-analytics-api.conf**:

- **aaa_mode**—Takes one of these values:
 - **no-auth**
 - **cloud-admin**
- **cloud_admin_role**—The user with this role has full access to everything. By default, this is set to "admin". This role must be configured in Keystone.

RELATED DOCUMENTATION

| [User Configuration for Analytics Alarms and Log Statistics | 131](#)

Configuring Analytics as a Standalone Solution

IN THIS SECTION

- [Overview: Contrail Analytics as a Standalone Solution | 145](#)
- [Configuration Examples for Standalone | 146](#)

Starting with Contrail 4.0, it is possible to configure Contrail Analytics as a standalone solution.

Overview: Contrail Analytics as a Standalone Solution

Starting with Contrail 4.0 (containerized Contrail), Contrail Analytics can be configured as a standalone solution.

The following services are necessary for a standalone solution:

- config
- webui
- analytics
- analyticsdb

A standalone Contrail Analytics solution consists of the following containers:

- controller container with only config and webui services enabled
- analytics container
- analyticsdb container

Configuration Examples for Standalone

IN THIS SECTION

- [Examples: Inventory File Controller Components | 146](#)
- [JSON Configuration Examples | 147](#)

The following are examples of default inventory file configurations for the controller container for standalone Contrail analytics.

Examples: Inventory File Controller Components

IN THIS SECTION

- [Single Node Cluster | 146](#)
- [Multi-Node Cluster | 146](#)

The following are example analytics standalone solution inventory file configurations for Contrail controller container components.

Single Node Cluster

```
[contrail-controllers]
10.xx.32.10          controller_components=[ 'config', 'webui' ]

[contrail-analyticsdb]
10.xx.32.10

[contrail-analytics]
10.xx.32.10
```

Multi-Node Cluster

```
[contrail-controllers]
10.xx.32.10          controller_components=[ 'config', 'webui' ]
10.xx.32.11          controller_components=[ 'config', 'webui' ]
10.xx.32.12          controller_components=[ 'config', 'webui' ]
```

```
[contrail-analyticsdb]
10.xx.32.10
10.xx.32.11
10.xx.32.12

[contrail-analytics]
10.xx.32.10
10.xx.32.11
10.xx.32.12
```

JSON Configuration Examples

IN THIS SECTION

- [Example: JSON Single Node Cluster | 147](#)
- [Example: JSON Multi-Node Cluster | 147](#)

The following are example JSON file configurations for (**server.json**) for Contrail analytics standalone solution.

Example: JSON Single Node Cluster

```
{
  "cluster_id": "cluster1",
  "domain": "sm-domain.com",
  "id": "server1",
  "parameters" : {
    "provision": {
      "contrail_4": {
        "controller_components": "['config','webui']"
      },
      ...
    }
  }
}
```

Example: JSON Multi-Node Cluster

```
{
  "cluster_id": "cluster1",
  "domain": "sm-domain.com",
```

```

    "id": "server1",
    "parameters" : {
      "provision": {
        "contrail_4": {
          "controller_components": "['config','webui']"
        },
        ...
      },
      {
        "cluster_id": "cluster1",
        "domain": "sm-domain.com",
        "id": "server2",
        "parameters" : {
          "provision": {
            "contrail_4": {
              "controller_components": "['config','webui']"
            },
            ...
          },
          {
            "cluster_id": "cluster1",
            "domain": "sm-domain.com",
            "id": "server3",
            "parameters" : {
              "provision": {
                "contrail_4": {
                  "controller_components": "['config','webui']"
                },
                ...
              },
              ...
            }
          }
        }
      }
    }
  }
}

```

RELATED DOCUMENTATION

[Configuring Secure Sandesh and Introspect for Contrail Analytics](#) | 149

[Understanding Contrail Analytics](#) | 97

Configuring Secure Sandesh and Introspect for Contrail Analytics

IN THIS SECTION

- [Configuring Secure Sandesh Connection | 149](#)
- [Configuring Secure Introspect Connection | 149](#)

Configuring Secure Sandesh Connection

All Contrail services use Sandesh, a southbound interface protocol based on Apache Thrift, to send analytics data such as system logs, object logs, UVEs, flow logs, and the like, to the collector service in the Contrail Analytics node. The Transport Layer Security (TLS) protocol is used for certificate exchange, mutual authentication, and negotiating ciphers to secure the Sandesh connection from potential tampering and eavesdropping.

To configure a secure Sandesh connection, configure the following parameters in all Contrail services that connect to the collector (Sandesh clients) and the Sandesh server.

Parameter	Description	Default
[SANDESH].sandesh_keyfile	Path to the node's private key	/etc/contrail/ssl/private/server-privkey.pem
[SANDESH].sandesh_certfile	Path to the node's public certificate	/etc/contrail/ssl/certs/server.pem
[SANDESH].sandesh_ca_cert	Path to the CA certificate	/etc/contrail/ssl/certs/ca-cert.pem
[SANDESH].sandesh_ssl_enable	Enable or disable secure Sandesh connection	false

Configuring Secure Introspect Connection

All Contrail services are embedded with a web server that can be used to query the internal state of the data structures, view trace messages, and perform other extensive debugging. The Transport Layer Security (TLS) protocol is used for certificate exchange, mutual authentication, and negotiating ciphers to secure the introspect connection from potential tampering and eavesdropping.

To configure a secure introspect connection, configure the following parameters in the Contrail service, see [Table 33 on page 150](#).

Table 33: Secure Introspect Parameters

Parameter	Description	Default
[SANDESH].sandesh_keyfile	Path to the node's private key	/etc/contrail/ssl/private/server-privkey.pem
[SANDESH].sandesh_certfile	Path to the node's public certificate	/etc/contrail/ssl/certs/server.pem
[SANDESH].sandesh_ca_cert	Path to the CA certificate	/etc/contrail/ssl/certs/ca-cert.pem
[SANDESH].introspect_ssl_enable	Enable or disable secure introspect connection	false

Using Contrail Analytics to Monitor and Troubleshoot the Network

IN THIS CHAPTER

- [Monitoring the System | 152](#)
- [Debugging Processes Using the Contrail Introspect Feature | 155](#)
- [Monitor > Infrastructure > Dashboard | 159](#)
- [Monitor > Infrastructure > Control Nodes | 162](#)
- [Monitor > Infrastructure > Virtual Routers | 171](#)
- [Monitor > Infrastructure > Analytics Nodes | 183](#)
- [Monitor > Infrastructure > Config Nodes | 189](#)
- [Monitor > Networking | 193](#)
- [Query > Flows | 203](#)
- [Query > Logs | 212](#)

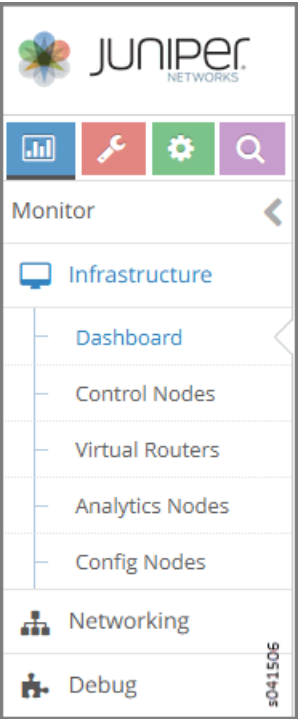
Monitoring the System

The **Monitor** icon on the Contrail Controller provides numerous options so you can view and analyze usage and other activity associated with all nodes of the system, through the use of reports, charts, and detailed lists of configurations and system activities.

Monitor pages support monitoring of infrastructure components—control nodes, virtual routers, analytics nodes, and config nodes. Additionally, users can monitor networking and debug components.

Use the menu options available from the **Monitor** icon to configure and view the statistics you need for better understanding of the activities in your system. See [Figure 37 on page 152](#)

Figure 37: Monitor Menu



See [Table 34 on page 153](#) for descriptions of the items available under each of the menu options from the **Monitor** icon.

Table 34: Monitor Menu Options

Option	Description
Infrastructure > Dashboard	Shows “at-a-glance” status view of the infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, and a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts. See “Monitor > Infrastructure > Dashboard” on page 159.
Infrastructure > Control Nodes	View a summary for all control nodes in the system, and for each control node, view: <ul style="list-style-type: none"> • Graphical reports of memory usage and average CPU load. • Console information for a specified time period. • A list of all peers with details about type, ASN, and the like. • A list of all routes, including next hop, source, local preference, and the like. See “Monitor > Infrastructure > Control Nodes” on page 162.
Infrastructure > Virtual Routers	View a summary of all vRouters in the system, and for each vRouter, view: <ul style="list-style-type: none"> • Graphical reports of memory usage and average CPU load. • Console information for a specified time period. • A list of all interfaces with details such as label, status, associated network, IP address, and the like. • A list of all associated networks with their ACLs and VRFs. • A list of all active flows with source and destination details, size, and time. See “Monitor > Infrastructure > Virtual Routers” on page 171.
Infrastructure > Analytics Nodes	View activity for the analytics nodes, including memory and CPU usage, analytics host names, IP address, status, and more. See “Monitor > Infrastructure > Analytics Nodes” on page 183.
Infrastructure > Config Nodes	View activity for the config nodes, including memory and CPU usage, config host names, IP address, status, and more. See “Monitor > Infrastructure > Config Nodes” on page 189.

Table 34: Monitor Menu Options (*continued*)

Option	Description
Networking > Networks	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> • Total traffic in and out. • Inter VN traffic in and out. • The most active ports, peers, and flows for a specified duration. • All traffic ingress and egress from connected networks, including their attached policies. <p>See “Monitor > Networking” on page 193.</p>
Networking > Dashboard	<p>For all virtual networks for all projects in the system, view graphical traffic statistics, including:</p> <ul style="list-style-type: none"> • Total traffic in and out. • Inter VN traffic in and out. <p>You can view the statistics in varying levels of granularity, for example, for a whole project, or for a single network. See “Monitor > Networking” on page 193.</p>
Networking > Projects	View essential information about projects in the system including name, associated networks, and traffic in and out.
Networking > Networks	View essential information about networks in the system including name and traffic in and out.
Networking > Instances	View essential information about instances in the system including name, associated networks, interfaces, vRouters, and traffic in and out.
Debug > Packet Capture	<ul style="list-style-type: none"> • Add and manage packet analyzers. • Attach packet captures and configure their details. • View a list of all packet analyzers in the system and the details of their configurations, including source and destination networks, ports, and IP addresses.

RELATED DOCUMENTATION

[Monitor > Infrastructure > Dashboard | 159](#)

[Monitor > Infrastructure > Control Nodes | 162](#)

[Monitor > Infrastructure > Virtual Routers | 171](#)

[Monitor > Networking | 193](#)

[Query > Logs | 212](#)

[Query > Flows | 203](#)

Debugging Processes Using the Contrail Introspect Feature

This topic describes how to use the Sandesh infrastructure and the Contrail Introspect feature to debug processes.

Introspect is a mechanism for taking a program object and querying information about it.

Sandesh is the name of a unified infrastructure in the Contrail Virtual Networking solution.

Sandesh is a way for the Contrail daemons to provide a request-response mechanism. Requests and responses are defined in Sandesh format and the Sandesh compiler generates code to process the requests and send responses.

Sandesh also provides a way to use a Web browser to send Sandesh requests to a Contrail daemon and get the Sandesh responses. This feature is used to debug processes by looking into the operational status of the daemons.

Each Contrail daemon starts an HTTP server, with the following page types:

- The main index.html listing all Sandesh modules and the links to them.
- Sandesh module pages that present HTML forms for each Sandesh request.
- XML-based dynamically-generated pages that display Sandesh responses.
- An automatically generated page that shows all code needed for rendering and all HTTP server-client interactions.

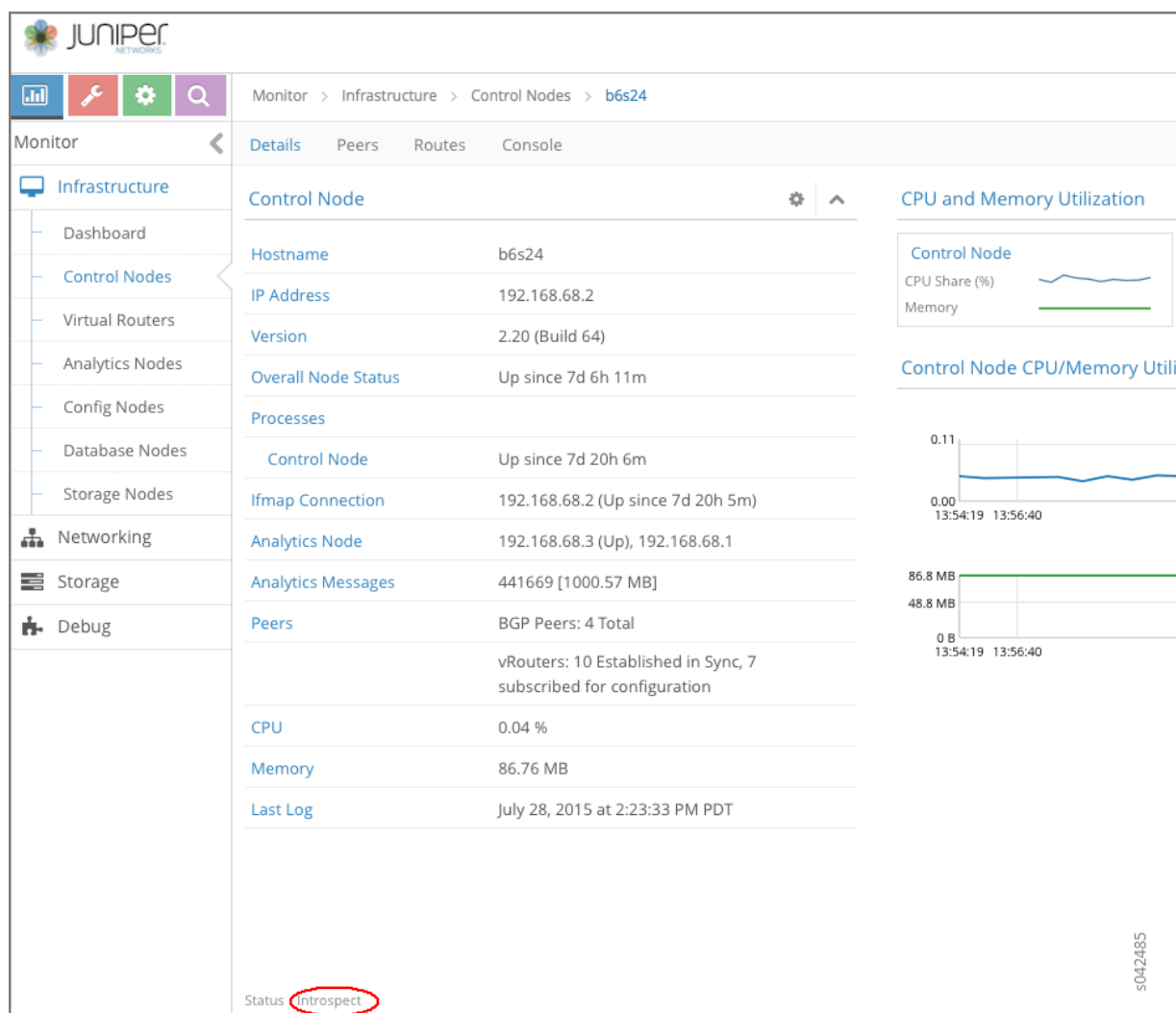
You can display the HTTP introspect of a Contrail daemon directly by accessing the following Introspect ports:

- `<controller-ip>:8083`. This port displays the *contrail-control* introspect port.
- `<compute-ip>:8085`. This port displays the *contrail-vrouter-agent* introspect port.

Another way to launch the Introspect page is by browsing to a particular node page using the Contrail Web user interface.

[Figure 38 on page 156](#) shows the contrail-control infrastructure page. Notice the Introspect link at the bottom of the Control Nodes Details tab window.

Figure 38: Control Nodes Details Tab Window



The following are the Sandesh modules for the Contrail control process (contrail-control) Introspect port.

- bgp_peer.xml
- control_node.xml
- cpuinfo.xml
- discovery_client_stats.xml
- ifmap_log.xml
- ifmap_server_show.xml
- rtarget_group.xml
- sandesh_trace.xml
- sandesh_uve.xml

- service_chaining.xml
- static_route.xml
- task.xml
- xmpp_server.xml

Figure 39 on page 157 shows the Controller Introspect window.

Figure 39: Controller Introspect Window

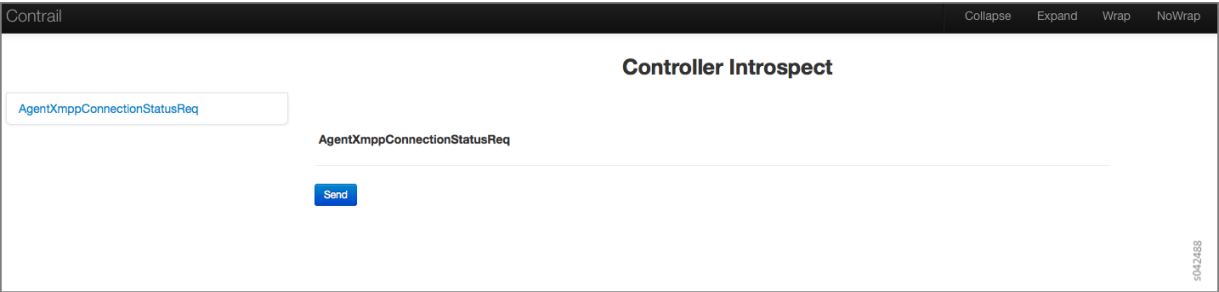


Figure 40 on page 157 shows an example of the BGP Peer (bgp_peer.xml) Introspect page.

Figure 40: BGP Peer Introspect Page

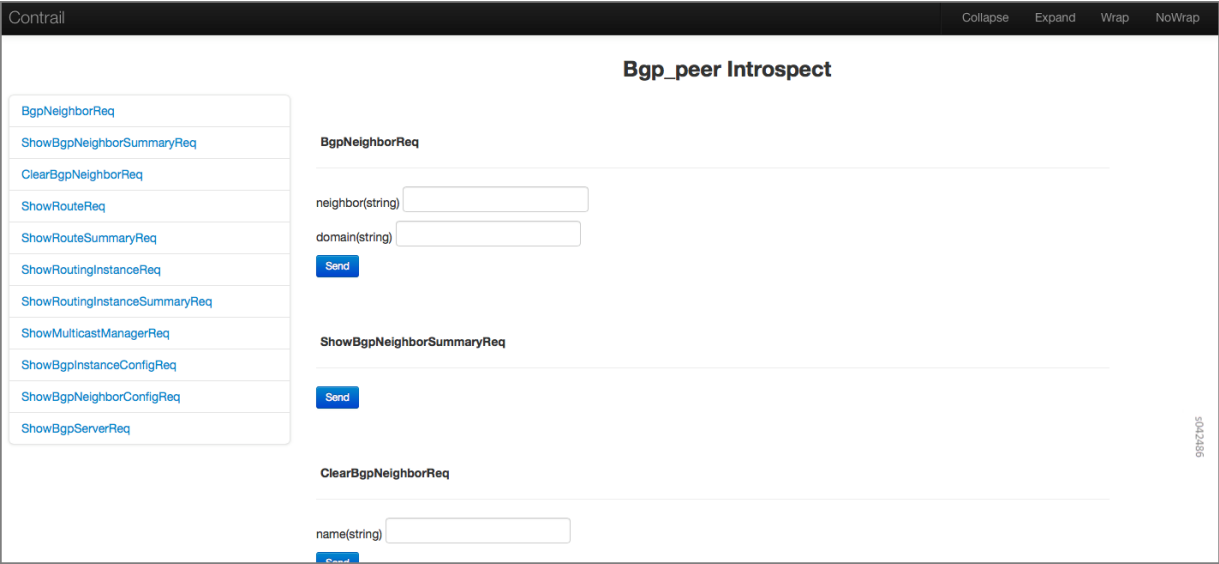


Figure 41 on page 158 shows an example of the BGP Neighbor Summary Introspect page.

Figure 41: BGP Neighbor Summary Introspect Page

Contrail										
ShowBgpNeighborSummaryResp										
neighbors										
peer	deleted	deleted_at	peer_address	peer_id	peer_asn	encoding	peer_type	state	local_address	local_id
b6s23	false	-	192.168.68.1	192.168.68.1	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
b6s25	false	-	192.168.68.3	192.168.68.3	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
mx1	false	-	192.168.100.1	192.168.100.1	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
mx2	false	-	192.168.100.2	192.168.100.2	64512	BGP	internal	Established	192.168.68.2	192.168.68.2
b6s28	false	-	192.168.68.6	-	0	XMPP	internal	Established	192.168.68.2	-
b6s18	false	-	192.168.69.5	-	0	XMPP	internal	Established	192.168.68.2	-
b6s13	false	-	192.168.69.8	-	0	XMPP	internal	Established	192.168.68.2	-
b6s7	false	-	192.168.69.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s33	false	-	192.168.68.11	-	0	XMPP	internal	Established	192.168.68.2	-
b6s9	false	-	192.168.69.10	-	0	XMPP	internal	Established	192.168.68.2	-
b6s26	false	-	192.168.68.4	-	0	XMPP	internal	Established	192.168.68.2	-

The following are the Sandesh modules for the Contrail vRouter agent (**contrail-vrouter-agent**) Introspect port.

- agent.xml
- agent_stats_interval.xml
- cfg.xml
- controller.xml
- cpuinfo.xml
- diag.xml
- discovery_client_stats.xml
- flow_stats_interval.xml
- ifmap_agent.xml
- kstate.xml
- multicast.xml
- pkt.xml
- port_ipc.xml
- sandesh_trace.xml
- sandesh_uve.xml
- services.xml
- stats_interval.xml
- task.xml
- xmpp_server.xml

Figure 42 on page 159 shows an example of the Agent (agent.xml) Introspect page.

Figure 42: Agent Introspect Page

Contrail

CollapseExpandWrapNoWrap

AgentXmppConnectionStatus

peer									
controller_ip	state	cfg_controller	mcast_controller	last_state	last_event	last_state_at	flap_count	flap_time	
192.168.68.3	Established	Yes	No	OpenSent	xmsm::EvXmppKeepalive	2015-Jul-21 01:20:57.616019	2	2015-Jul-21 01:20:57.555077	rx
192.168.68.2	Established	No	Yes	OpenSent	xmsm::EvXmppKeepalive	2015-Jul-21 01:20:59.599875	2	2015-Jul-21 01:20:59.548692	rx

Monitor > Infrastructure > Dashboard

IN THIS SECTION

- [Monitor Dashboard | 159](#)
- [Monitor Individual Details from the Dashboard | 160](#)
- [Using Bubble Charts | 161](#)
- [Color-Coding of Bubble Charts | 161](#)

Use **Monitor > Infrastructure > Dashboard** to get an “at-a-glance” view of the system infrastructure components, including the numbers of virtual routers, control nodes, analytics nodes, and config nodes currently operational, a bubble chart of virtual routers showing the CPU and memory utilization, log messages, system information, and alerts.

Monitor Dashboard

Click **Monitor > Infrastructure > Dashboard** on the left to view the **Dashboard**. See [Figure 43 on page 160](#).

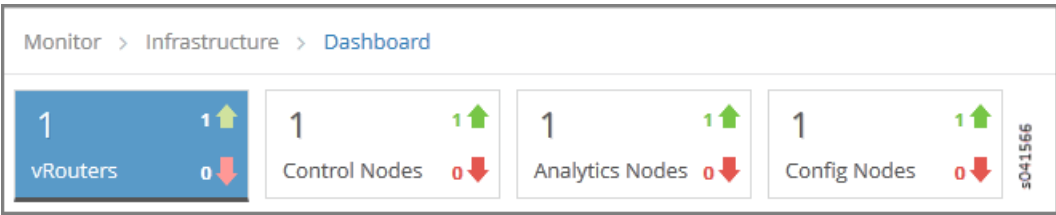
Figure 43: Monitor > Infrastructure > Dashboard



Monitor Individual Details from the Dashboard

Across the top of the **Dashboard** screen are summary boxes representing the components of the system that are shown in the statistics. See [Figure 44 on page 160](#). Any of the control nodes, virtual routers, analytics nodes, and config nodes can be monitored individually and in detail from the **Dashboard** by clicking an associated box, and drilling down for more detail.

Figure 44: Dashboard Summary Boxes



Detailed information about monitoring each of the areas represented by the boxes is provided in the links in [Table 35 on page 160](#).

Table 35: Dashboard Summary Boxes

Box	For More Information
vRouters	“Monitor > Infrastructure > Virtual Routers” on page 171
Control Nodes	“Monitor > Infrastructure > Control Nodes” on page 162

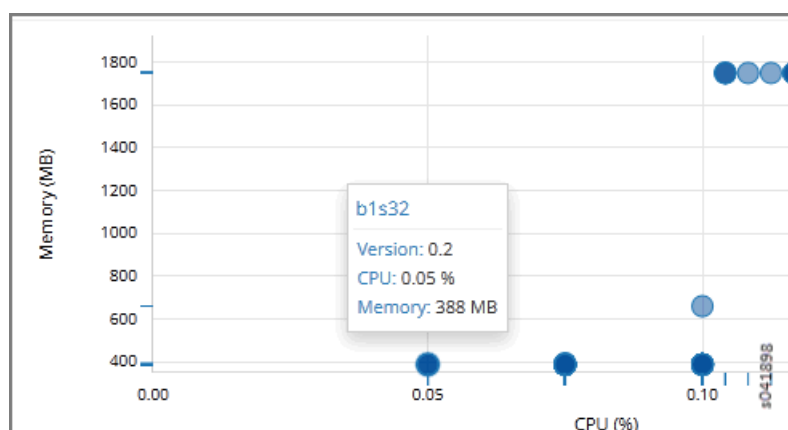
Table 35: Dashboard Summary Boxes (*continued*)

Box	For More Information
Analytics Nodes	“Monitor > Infrastructure > Analytics Nodes” on page 183
Config Nodes	“Monitor > Infrastructure > Config Nodes” on page 189

Using Bubble Charts

Bubble charts show the CPU and memory utilization of components contributing to the current analytics display, including vRouters, control nodes, config nodes, and the like. You can hover over any bubble to get summary information about the component it represents; see [Figure 45 on page 161](#). You can click through the summary information to get more details about the component.

Figure 45: Bubble Summary Information



Color-Coding of Bubble Charts

Bubble charts use the following color-coding scheme:

Control Nodes

- Blue—working as configured.
- Red—error, at least one configured peer is down.

vRouters

- Blue—working, but no instance is launched.
- Green—working with at least one instance launched.
- Red—error, there is a problem with connectivity or a vRouter is in a failed state.

RELATED DOCUMENTATION

[Monitor > Infrastructure > Virtual Routers | 171](#)

[Monitor > Infrastructure > Control Nodes | 162](#)

[Monitor > Infrastructure > Analytics Nodes | 183](#)

[Monitor > Infrastructure > Config Nodes | 189](#)

Monitor > Infrastructure > Control Nodes

IN THIS SECTION

- [Monitor Control Nodes Summary | 162](#)
- [Monitor Individual Control Node Details | 164](#)
- [Monitor Individual Control Node Console | 165](#)
- [Monitor Individual Control Node Peers | 168](#)
- [Monitor Individual Control Node Routes | 169](#)

Use **Monitor > Infrastructure > Control Nodes** to gain insight into usage statistics for control nodes.

Monitor Control Nodes Summary

Select **Monitor > Infrastructure > Control Nodes** to see a graphical chart of average memory usage versus average CPU percentage usage for all control nodes in the system. Also on this screen is a list of all control nodes in the system. See [Figure 46 on page 163](#). See [Table 36 on page 163](#) for descriptions of the fields on this screen.

Figure 46: Control Nodes Summary



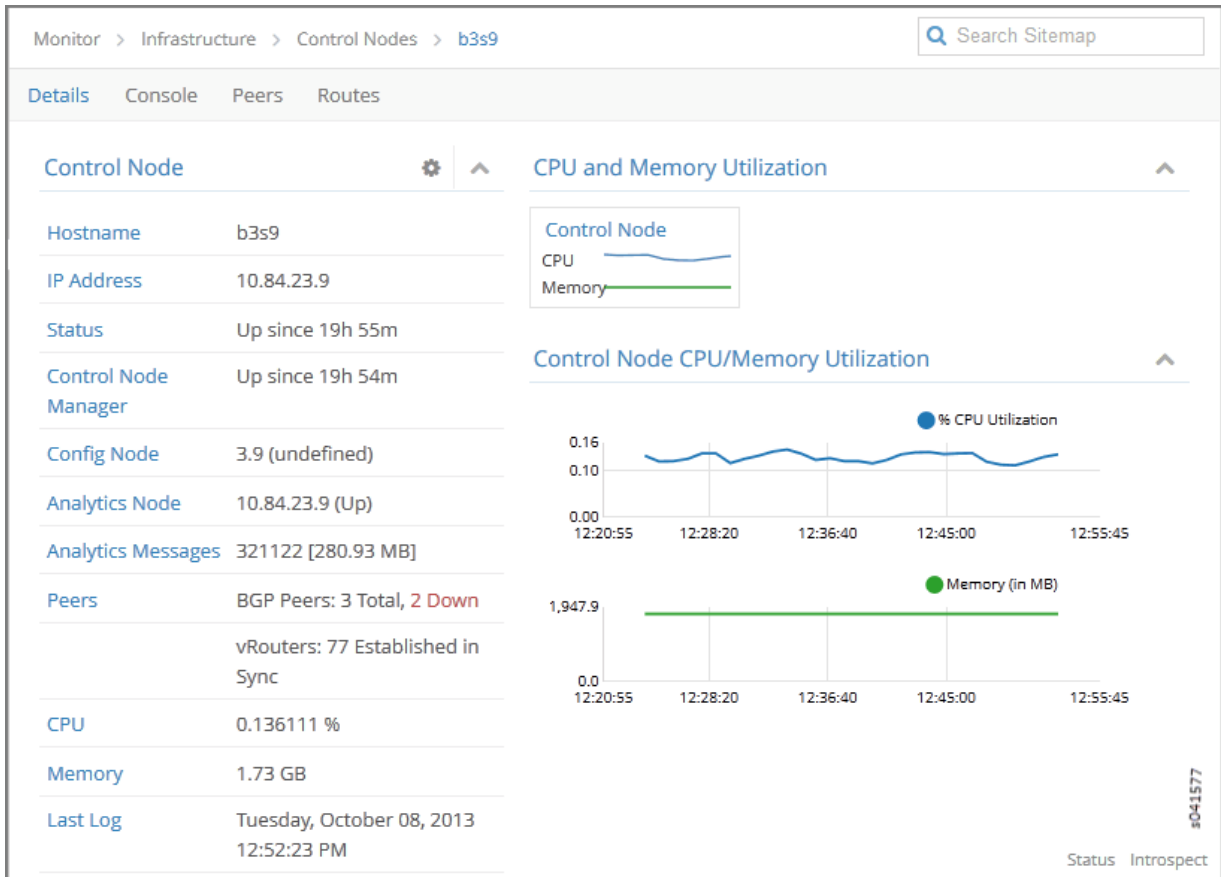
Table 36: Control Nodes Summary Fields

Field	Description
Host name	The name of the control node.
IP Address	The IP address of the control node.
Version	The software version number that is installed on the control node.
Status	The current operational status of the control node — Up or Down.
CPU (%)	The CPU percentage currently in use by the selected control node.
Memory	The memory in MB currently in use and the total memory available for this control node.
Total Peers	The total number of peers for this control node.
Established in Sync Peers	The total number of peers in sync for this control node.
Established in Sync vRouters	The total number of vRouters in sync for this control node.

Monitor Individual Control Node Details

Click the name of any control nodes listed under the **Control Nodes** title to view an array of graphical reports of usage and numerous details about that node. There are several tabs available to help you probe into more details about the selected control node. The first tab is the **Details** tab; see [Figure 47 on page 164](#).

Figure 47: Individual Control Node—Details Tab



The Details tab provides a summary of the status and activity on the selected node, and presents graphical displays of CPU and memory usage. See [Table 37 on page 164](#) for descriptions of the fields on this tab.

Table 37: Individual Control Node—Details Tab Fields

Field	Description
Hostname	The host name defined for this control node.
IP Address	The IP address of the selected node.
Status	The operational status of the control node.

Table 37: Individual Control Node—Details Tab Fields (*continued*)

Field	Description
Control Node Manager	The operational status of the control node manager.
Config Node	The IP address of the configuration node associated with this control node.
Analytics Node	The IP address of the node from which analytics (monitor) information is derived.
Analytics Messages	The total number of analytics messages in and out from this node.
Peers	The total number of peers established for this control node and how many are in sync and of what type.
CPU	The average percent of CPU load incurred by this control node.
Memory	The average memory usage incurred by this control node.
Last Log	The date and time of the last log message issued about this control node.
Control Node CPU/Memory Utilization	A graphic display x, y chart of the average CPU load and memory usage incurred by this control node over time.

Monitor Individual Control Node Console

Click the **Console** tab for an individual control node to display system logging information for a defined time period, with the last 5 minutes of information as the default display. See [Figure 48 on page 166](#).

Figure 48: Individual Control Node—Console Tab

Monitor > Infrastructure > Control Nodes > b3s9

Search Sitemap

Details
Console
Peers
Routes

Console Logs

Time Range

Custom

From Time

Oct 08, 2013 02:26:33 PM

To Time

Oct 08, 2013 02:31:33 PM

Log Category

All

Log Type

any

Log Level

SYS_DEBUG

Limit

Limit 10 mess

Auto Refresh

☒

Display Logs
Reset

Time	Category	Log Type	Log
2013-10-08 14:31:30:351:353	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.252 : P fsm::EvConnectTimerExp
2013-10-08 14:31:27:971:482	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.253 : P state Connect
2013-10-08 14:31:24:970:157	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.253 : P fsm::EvConnectTimerExp
2013-10-08 14:30:58:220:866	BGP	BgpStateMachineSessionMessageLog	Bgp Peer 10.84.23.252 : P state Connect

See [Table 38 on page 166](#) for descriptions of the fields on the **Console** tab screen.

Table 38: Control Node: Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the Last 5 mins through to the Last 24 hrs . The default display is for the Last 5 mins .
Log Category	Select a log category to display: <div> All _default_ XMPP TCP </div>
Log Type	Select a log type to display.

Table 38: Control Node: Console Tab Fields (*continued*)

Field	Description
Log Level	<p>Select a log severity level to display:</p> <p>SYS_EMERG</p> <p>SYS_ALERT</p> <p>SYS_CRIT</p> <p>SYS_ERR</p> <p>SYS_WARN</p> <p>SYS_NOTICE</p> <p>SYS_INFO</p> <p>SYS_DEBUG</p>
Search	Enter any text string to search and display logs containing that string.
Limit	<p>Select from a list an amount to limit the number of messages displayed:</p> <p>No Limit</p> <p>Limit 10 messages</p> <p>Limit 50 messages</p> <p>Limit 100 messages</p> <p>Limit 200 messages</p> <p>Limit 500 messages</p>
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor Individual Control Node Peers

The **Peers** tab displays the peers for an individual control node and their peering state. Click the expansion arrow next to the address of any peer to reveal more details. See [Figure 49 on page 168](#).

Figure 49: Individual Control Node—Peers Tab

Peer	Peer Type	Peer ASN	Status	Last flap	Messages (Recv/Sent)
10.84.23.252	BGP	64512	Active, -	-	0/ 0
10.84.23.8	BGP	64512	Established, in sync	-	3754/ 3758
10.84.23.253	BGP	64512	Connect, -	-	0/ 0
10.84.21.4	XMPP	-	Established, in sync	-	2751/ 5189
10.84.21.5	XMPP	-	Established, in sync	-	2753/ 5802
10.84.21.6	XMPP	-	Established, in sync	-	2752/ 4264
10.84.21.34	XMPP	-	Established, in sync	-	2753/ 5659

Details :

```

- {
  name: "b3s9:10.84.21.34",
  value: - {
    xmppPeerInfoData: - {
      state_info: - {
        last_state: "Active",
        state: "Established",
        last_state_at: 1381190447915913
      },
      peer_stats_info: - {

```

See [Table 39 on page 168](#) for descriptions of the fields on the **Peers** tab screen.

Table 39: Control Node: Peers Tab Fields

Field	Description
Peer	The hostname of the peer.
Peer Type	The type of peer.
Peer ASN	The autonomous system number of the peer.
Status	The current status of the peer.

Table 39: Control Node: Peers Tab Fields (*continued*)

Field	Description
Last flap	The last flap detected for this peer.
Messages (Recv/Sent)	The number of messages sent and received from this peer.

Monitor Individual Control Node Routes

The **Routes** tab displays active routes for this control node and lets you query the results. Use horizontal and vertical scroll bars to view more results. Click the expansion icon next to a routing table name to reveal more details about the selected route. See [Figure 50 on page 169](#).

Figure 50: Individual Control Node—Routes Tab

Details Console Peers **Routes**

Routing Instance: All Address Family: All Limit 50 Routes

Peer Source: All Prefix: Prefix Protocol: All

Display Routes Reset

Routes

Routing Table	Prefix	Protocol	Source	Next hop	Label	Secur...	Origin VN
bgp.l3vpn.0	10.84.21.1:13:192.168.30.240/32	XMPP	b1s1	10.84.21.1	28	3	default-domain:demo:v n30
		BGP	10.84.23.9	10.84.21.1	28	3	default-domain:demo:v n30
	10.84.21.1:14:192.168.31.242/32	XMPP	b1s1	10.84.21.1	29	3	default-domain:demo:v n31
		BGP	10.84.23.9	10.84.21.1	29	3	default-domain:demo:v n31
	10.84.21.1:1:192.168.2.231/32	XMPP	b1s1	10.84.21.1	16	3	default-domain:demo:v n2

See [Table 40 on page 169](#) for descriptions of the fields on the **Routes** tab screen.

Table 40: Control Node: Routes Tab Fields

Field	Description
Routing Instance	You can select a single routing instance from a list of all instances for which to display the active routes.

Table 40: Control Node: Routes Tab Fields (*continued*)

Field	Description
Address Family	Select an address family for which to display the active routes: All (default) I3vpn inet inetmcast
(Limit Field)	Select to limit the display of active routes: Limit 10 Routes Limit 50 Routes Limit 100 Routes Limit 200 Routes
Peer Source	Select from a list of available peers the peer for which to display the active routes, or select All.
Prefix	Enter a route prefix to limit the display of active routes to only those with the designated prefix.
Protocol	Select a protocol for which to display the active routes: All (default) XMPP BGP ServiceChain Static
Display Routes	Click this button to refresh the display of routes after selecting different display criteria.
Reset	Click this button to clear any selected criteria and return the display to default values.
<i>Column</i>	<i>Description</i>
Routing Table	The name of the routing table that stores this route.
Prefix	The route prefix for each active route displayed.
Protocol	The protocol used by the route.
Source	The host source for each active route displayed.

Table 40: Control Node: Routes Tab Fields (*continued*)

Field	Description
Next hop	The IP address of the next hop for each active route displayed.
Label	The label for each active route displayed.
Security	The security value for each active route displayed.
Origin VN	The virtual network from which the route originates.
AS Path	The AS path for each active route displayed.

Monitor > Infrastructure > Virtual Routers

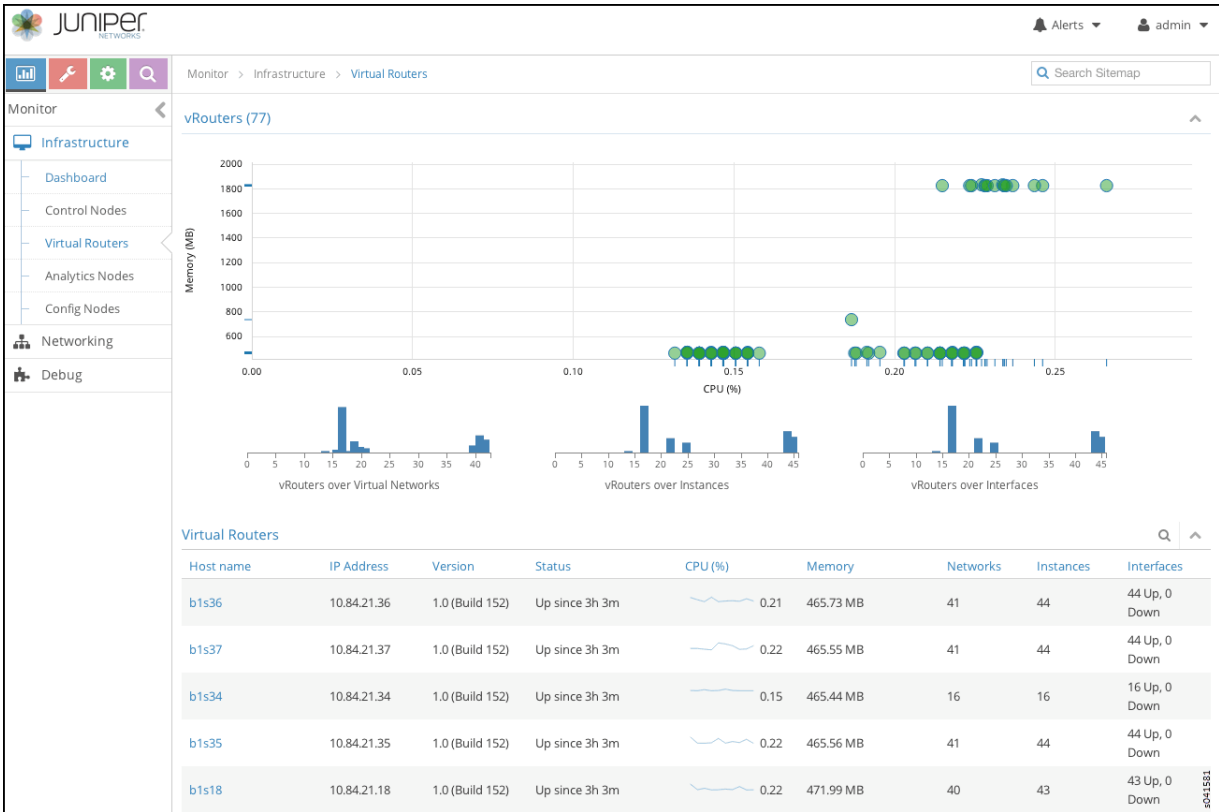
IN THIS SECTION

- [Monitor vRouters Summary | 171](#)
- [Monitor Individual vRouters Tabs | 173](#)
- [Monitor Individual vRouter Details Tab | 173](#)
- [Monitor Individual vRouters Interfaces Tab | 175](#)
- [Monitor Individual vRouters Networks Tab | 176](#)
- [Monitor Individual vRouters ACL Tab | 177](#)
- [Monitor Individual vRouters Flows Tab | 179](#)
- [Monitor Individual vRouters Routes Tab | 180](#)
- [Monitor Individual vRouter Console Tab | 181](#)

Monitor vRouters Summary

Click **Monitor > Infrastructure > Virtual Routers** to view the **vRouters** summary screen. See [Figure 51 on page 172](#).

Figure 51: vRouters Summary



See [Table 41 on page 172](#) for descriptions of the fields on the **vRouters Summary** screen.

Table 41: vRouters Summary Fields

Field	Description
Host name	The name of the vRouter. Click the name of any vRouter to reveal more details.
IP Address	The IP address of the vRouter.
Version	The version of software installed on the system.
Status	The current operational status of the vRouter – Up or Down.
CPU (%)	The CPU percentage currently in use by the selected vRouter.
Memory (MB)	The memory currently in use and the total memory available for this vRouter.
Networks	The total number of networks for this vRouter.
Instances	The total number of instances for this vRouter.

Table 41: vRouters Summary Fields (continued)

Field	Description
Interfaces	The total number of interfaces for this vRouter.

Monitor Individual vRouters Tabs

Click the name of any vRouter to view details about performance and activities for that vRouter. Each individual vRouters screen has the following tabs.

- **Details**—similar display of information as on individual control nodes **Details** tab. See [Figure 52 on page 174](#).
- **Console**—similar display of information as on individual control nodes **Console** tab. See [Figure 58 on page 181](#).
- **Interfaces**—details about associated interfaces. See [Figure 53 on page 175](#).
- **Networks**—details about associated networks. See [Figure 54 on page 177](#).
- **ACL**—details about access control lists. See [Figure 55 on page 178](#).
- **Flows**—details about associated traffic flows. See [Figure 56 on page 179](#).
- **Routes**—details about associated routes. See [Figure 57 on page 180](#).

Monitor Individual vRouter Details Tab

The **Details** tab provides a summary of the status and activity on the selected node, and presents graphical displays of CPU and memory usage; see [Figure 52 on page 174](#). See [Table 42 on page 174](#) for descriptions of the fields on this tab.

Figure 52: Individual vRouters—Details Tab

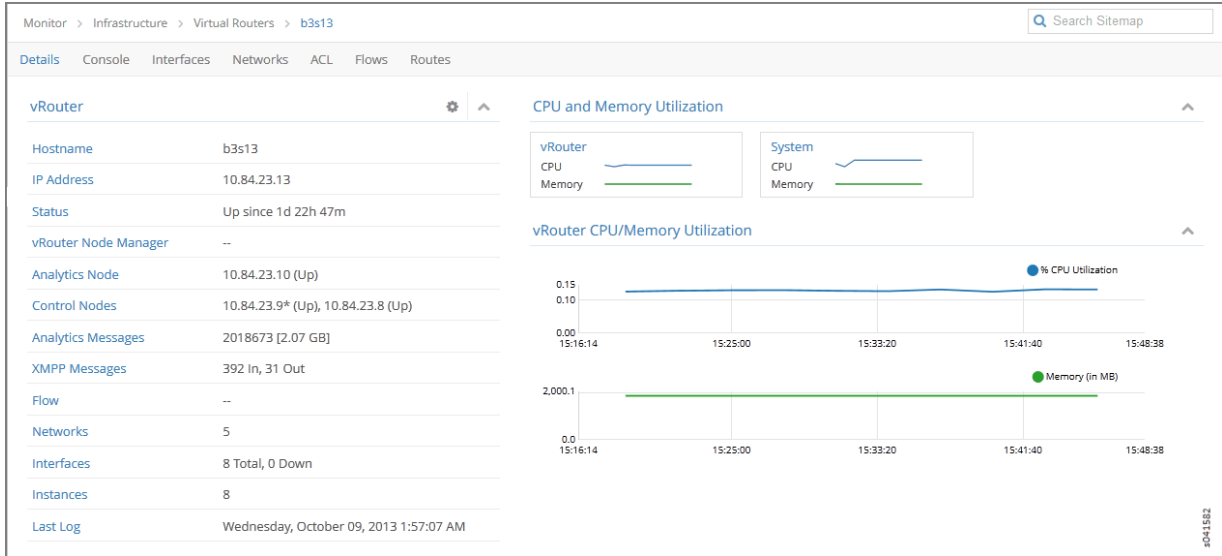


Table 42: vRouters Details Tab Fields

Field	Description
Hostname	The hostname of the vRouter.
IP Address	The IP address of the selected vRouter.
Status	The operational status of the vRouter.
vRouter Node Manager	The operational status of the vRouter node manager.
Analytics Node	The IP address of the node from which analytics (monitor) information is derived.
Control Nodes	The IP address of the configuration node associated with this vRouter.
Analytics Messages	The total number of analytics messages in and out from this node.
XMPP Messages	The total number of XMPP messages that have gone in and out of this vRouter.
Flow	The number of active flows and the total flows for this vRouter.
Networks	The number of networks associated with this vRouter.
Interfaces	The number of interfaces associated with this vRouter.
Instances	The number of instances associated with this vRouter.

Table 42: vRouters Details Tab Fields (*continued*)

Field	Description
Last Log	The date and time of the last log message issued about this vRouter.
vRouter CPU/Memory Utilization	Graphs (x, y) displaying CPU and memory utilization averages over time for this vRouter, in comparison to system utilization averages.

Monitor Individual vRouters Interfaces Tab

The **Interfaces** tab displays details about the interfaces associated with an individual vRouter. Click the expansion arrow next to any interface name to reveal more details. Use horizontal and vertical scroll bars to access all portions of the screen. See [Figure 53 on page 175](#). See [Table 43 on page 176](#) for descriptions of the fields on the **Interfaces** tab screen.

Figure 53: Individual vRouters—Interfaces Tab

Monitor > Infrastructure > Virtual Routers > b1s36

Search Sitemap

Details Console **Interfaces** Networks ACL Flows Routes

Interfaces

Name	Label	Status	Network	IP Address	Floating IP	Instance
▶ tap25e5cee3-07	18	Up	default-domain:demo:vn30	192.168.30.247	None	005132fd-0d83-4db7-88c8-bd49d68e9480
▶ tap4d91aab1-f1	25	Up	default-domain:demo:vn26	192.168.26.247	None	65d6c6e9-7a82-43d8-a706-f74d81715920
▶ tap5a8cd9dd-5b	27	Up	default-domain:demo:vn23	192.168.23.249	None	a159c518-4fb6-402a-ae0d-eb5b4457b551
▶ tap603a5e0b-8b	16	Up	default-domain:demo:vn19	192.168.19.247	None	fe622580-b0cf-4c6d-89e5-d2065e7e87e4
▲ tap68ad232c-76	19	Up	default-domain:demo:vn28	192.168.28.247	None	91089d89-76b5-46c2-abc9-b9693bcb37ac

Details:

```

- {
  index: "6",
  name: "tap68ad232c-76",
  uuid: "68ad232c-76d1-4fe2-a200-42182497545e",
  vrf_name: "default-domain:demo:vn28:vn28",
  active: "Active",
  dhcp_service: "Enable",

```

1041583

Table 43: vRouters: Interfaces Tab Fields

Field	Description
Name	The name of the interface.
Label	The label for the interface.
Status	The current status of the interface.
Network	The network associated with the interface.
IP Address	The IP address of the interface.
Floating IP	Displays any floating IP addresses associated with the interface.
Instance	The name of any instance associated with the interface.

Monitor Individual vRouters Networks Tab

The **Networks** tab displays details about the networks associated with an individual vRouter. Click the expansion arrow at the name of any network to reveal more details. See [Figure 54 on page 177](#). See [Table 44 on page 177](#) for descriptions of the fields on the **Networks** tab screen.

Figure 54: Individual vRouters—Networks Tab

Name	ACLs	VRF
default-domain:demo:vn24	a372751f-6497-41e9-b409-fa4ab5ce6b7f	default-domain:demo:vn24:vn24
default-domain:demo:vn22	195af177-0a28-49a1-9cf0-2ceac22af5a1	default-domain:demo:vn22:vn22
default-domain:demo:vn30	362cce6e-2894-42d6-ba03-3ee98cac8809	default-domain:demo:vn30:vn30
default-domain:demo:vn21	5918a068-1cd5-4993-9cff-386a807940ca	default-domain:demo:vn21:vn21
default-domain:demo:vn28	dd87c461-97c0-4d47-bff0-89040e7d6ab0	default-domain:demo:vn28:vn28
default-domain:demo:vn19	f0465432-6fc0-4fb3-967c-392100617408	default-domain:demo:vn19:vn19
default-domain:demo:vn2	1c46e7e0-f799-4bc6-ae09-e4654c263aa6	default-domain:demo:vn2:vn2

```

Details:
- {
  name: "default-domain:demo:vn2",
  uuid: "63d08f7a-b342-4892-9171-edab9f4c397f",
  acl_uuid: "1c46e7e0-f799-4bc6-ae09-e4654c263aa6",
  mirror_acl_uuid: - {},
  mirror_cfg_acl_uuid: - {},
  vrf_name: "default-domain:demo:vn2:vn2",
  ipam_data: - {
    list: - {

```

Table 44: vRouters: Networks Tab Fields

Field	Description
Name	The name of each network associated with this vRouter.
ACLs	The name of the access control list associated with the listed network.
VRF	The identifier of the VRF associated with the listed network.
Action	Click the icon to select the action: Edit, Delete

Monitor Individual vRouters ACL Tab

The **ACL** tab displays details about the access control lists (ACLs) associated with an individual vRouter. Click the expansion arrow next to the UUID of any ACL to reveal more details. See [Figure 55 on page 178](#). See [Table 45 on page 178](#) for descriptions of the fields on the **ACL** tab screen.

Figure 55: Individual vRouters—ACL Tab

The screenshot shows the 'ACL' tab for a specific vRouter. The interface includes a breadcrumb trail (Monitor > Infrastructure > Virtual Routers > b1s36), a search bar, and navigation tabs (Details, Console, Interfaces, Networks, ACL, Flows, Routes). The ACL list table has columns for UUID, Flows, Action, Protocol, Source Network or Prefix, Source Port, Destination Network or Prefix, and a 'D' column. Two ACLs are listed, with the second one selected. Below the table, the 'Details' section shows a JSON configuration for the selected ACL.

UUID	Flows	Action	Protocol	Source Network or Prefix	Source Port	Destination Network or Prefix	D
195af177-0a28-49a1-9cf0-2ceac22af5a1	8	pass	any	-	any	-	a
		pass	any	-	any	-	a
		pass	any	-	any	-	a
1c46e7e0-f799-4bc6-ae09-e4654c263aa6	8	pass	any	-	any	-	a

Details :

```

- {
  uuid: "1c46e7e0-f799-4bc6-ae09-e4654c263aa6",
  dynamic_acl: "false",
  entries: - {
    list: - {
      AclEntrySandeshData: - [
        - {
          ace_id: "1",

```

Table 45: vRouters: ACL Tab Fields

Field	Description
UUID	The universal unique identifier (UUID) associated with the listed ACL.
Flows	The flows associated with the listed ACL.
Action	The traffic action defined by the listed ACL.
Protocol	The protocol associated with the listed ACL.
Source Network or Prefix	The name or prefix of the source network associated with the listed ACL.
Source Port	The source port associated with the listed ACL.
Destination Network or Prefix	The name or prefix of the destination network associated with the listed ACL.
Destination Port	The destination port associated with the listed ACL.
ACE Id	The ACE ID associated with the listed ACL.

Monitor Individual vRouters Flows Tab

The **Flows** tab displays details about the flows associated with an individual vRouter. Click the expansion arrow next to any ACL/SG UUID to reveal more details. Use the horizontal and vertical scroll bars to access all portions of the screen. See [Figure 56 on page 179](#). See [Table 46 on page 179](#) for descriptions of the fields on the **Flows** tab screen.

Figure 56: Individual vRouters—Flows Tab

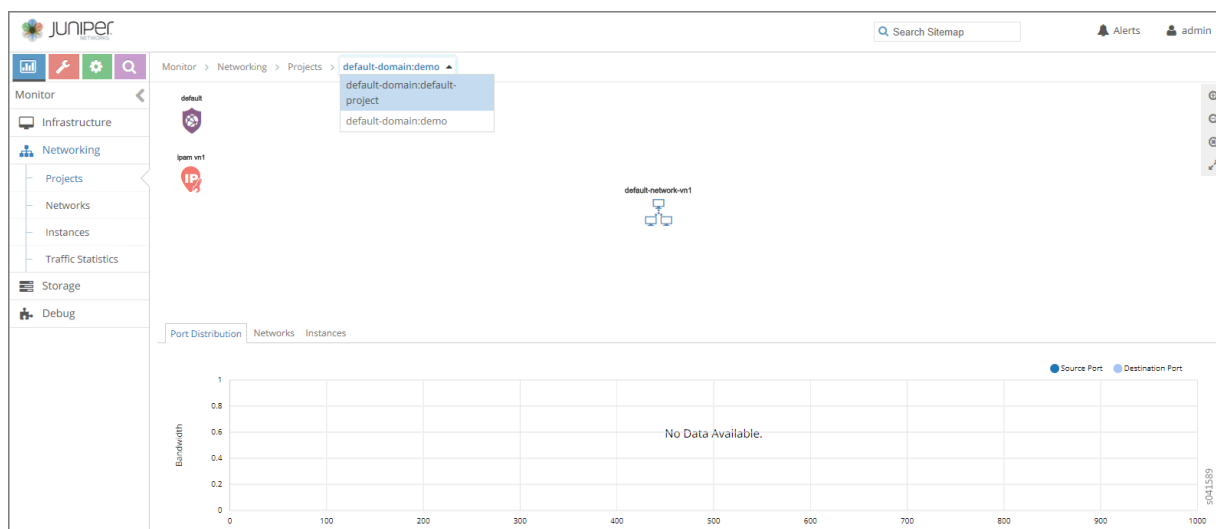


Table 46: vRouters: Flows Tab Fields

Field	Description
ACL UUID	The default is to show All flows, however, you can select from a drop down list any single flow to view its details.
ACL / SG UUID	The universal unique identifier (UUID) associated with the listed ACL or SG.
Protocol	The protocol associated with the listed flow.
Src Network	The name of the source network associated with the listed flow.
Src IP	The source IP address associated with the listed flow.
Src Port	The source port of the listed flow.
Dest Network	The name of the destination network associated with the listed flow.
Dest IP	The destination IP address associated with the listed flow.

Table 46: vRouters: Flows Tab Fields (*continued*)

Field	Description
Dest Port	The destination port associated with the listed flow.
Bytes/Pkts	The number of bytes and packets associated with the listed flow.
Setup Time	The setup time associated with the listed flow.

Monitor Individual vRouters Routes Tab

The **Routes** tab displays details about unicast and multicast routes in specific VRFs for an individual vRouter. Click the expansion arrow next to the route prefix to reveal more details. See [Figure 57 on page 180](#). See [Table 47 on page 180](#) for descriptions of the fields on the **Routes** tab screen.

Figure 57: Individual vRouters—Routes Tab

Monitor > Infrastructure > Virtual Routers > b1s36		
Details Console Interfaces Networks ACL Flows Routes		
VRF default-domain:default-project:ip-fabric:__default__		
Show Routes <input checked="" type="radio"/> Unicast <input type="radio"/> Multicast		
Routes		
Prefix	Next hop ...	Next hop details
▶ 0.0.0.0 / 0	arp	Interface: p2p0p0 Mac: 40:b4:f0:68:20:4e IP: 10.84.21.254
▶ 10.84.21.0 / 24	resolve	Source: Local Destination VN: default-domain:default-project:ip-fabric
▶ 10.84.21.1 / 32	arp	Interface: p2p0p0 Mac: 0:25:90:ab:b0:2c IP: 10.84.21.1
▶ 10.84.21.2 / 32	arp	Interface: p2p0p0 Mac: 0:25:90:ab:b0:38 IP: 10.84.21.2
▶ 10.84.21.3 / 32	arp	Interface: p2p0p0 Mac: 0:25:90:ab:af:ce IP: 10.84.21.3
▶ 10.84.21.4 / 32	arp	Interface: p2p0p0 Mac: 0:25:90:ab:ae:82 IP: 10.84.21.4
⌵ 10.84.21.5 / 32	arp	Interface: p2p0p0 Mac: 0:25:90:ab:b0:16 IP: 10.84.21.5
Details : <pre> - { dispPrefix: "10.84.21.5 / 32", path: - { nh: - { type: "arp", ref_count: "1", valid: "true", policy: "disabled", sip: "10.84.21.5", vrf: "default-domain:default-project:ip-fabric:__default__", </pre>		

Table 47: vRouters: Routes Tab Fields

Field	Description
VRF	Select from a drop down list the virtual routing and forwarding (VRF) to view.
Show Routes	Select to show the route type: Unicast or Multicast .

Table 47: vRouters: Routes Tab Fields (*continued*)

Field	Description
Prefix	The IP address prefix of a route.
Next hop	The next hop method for this route.
Next hop details	The next hop details for this route.

Monitor Individual vRouter Console Tab

Click the **Console** tab for an individual vRouter to display system logging information for a defined time period, with the last 5 minutes of information as the default display. See [Figure 58 on page 181](#). See [Table 48 on page 181](#) for descriptions of the fields on the **Console** tab screen.

Figure 58: Individual vRouter—Console Tab

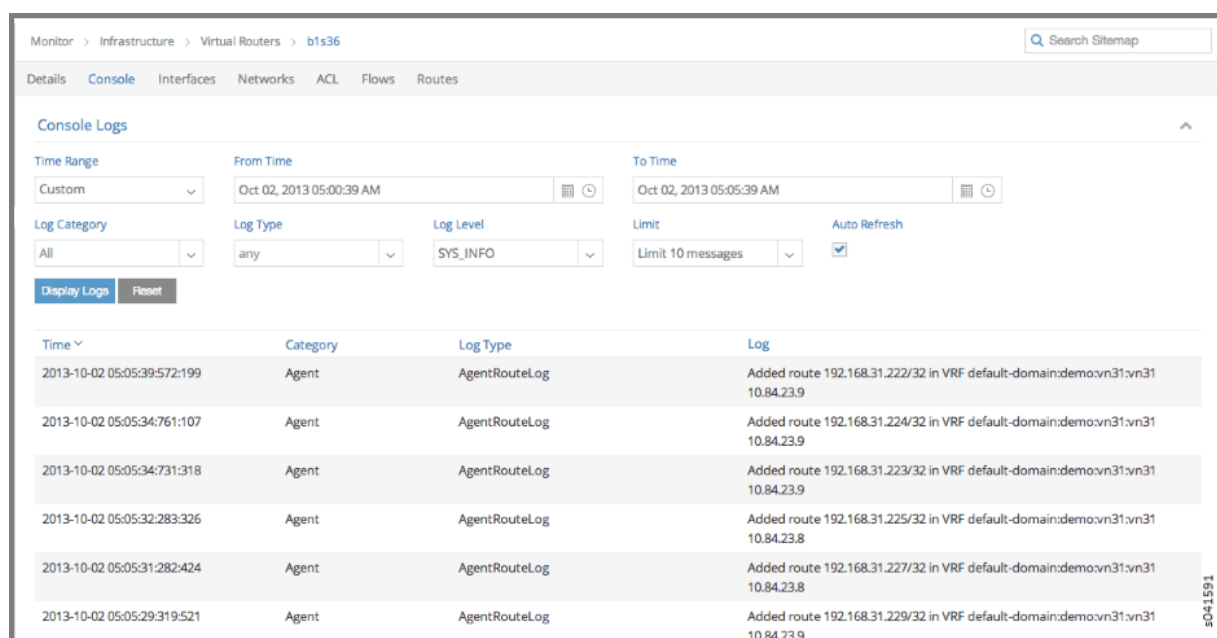


Table 48: Control Node: Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. There are several options, ranging from Last 5 mins through to the Last 24 hrs , plus a Custom time range.
From Time	If you select Custom in Time Range , enter the start time.

Table 48: Control Node: Console Tab Fields (*continued*)

Field	Description
To Time	If you select Custom in Time Range , enter the end time.
Log Category	<p>Select a log category to display:</p> <ul style="list-style-type: none"> • All • _default_ • XMPP • TCP
Log Type	Select a log type to display.
Log Level	<p>Select a log severity level to display:</p> <ul style="list-style-type: none"> • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_NOTICE • SYS_INFO • SYS_DEBUG
Limit	<p>Select from a list an amount to limit the number of messages displayed:</p> <ul style="list-style-type: none"> • No Limit • Limit 10 messages • Limit 50 messages • Limit 100 messages • Limit 200 messages • Limit 500 messages
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.
<i>Columns</i>	
Time	This column lists the time received for each log message displayed.

Table 48: Control Node: Console Tab Fields (continued)

Field	Description
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor > Infrastructure > Analytics Nodes

IN THIS SECTION

- [Monitor Analytics Nodes | 183](#)
- [Monitor Analytics Individual Node Details Tab | 184](#)
- [Monitor Analytics Individual Node Generators Tab | 186](#)
- [Monitor Analytics Individual Node QE Queries Tab | 186](#)
- [Monitor Analytics Individual Node Console Tab | 187](#)

Select **Monitor > Infrastructure > Analytics Nodes** to view the console logs, generators, and query expansion (QE) queries of the analytics nodes.

Monitor Analytics Nodes

Select **Monitor > Infrastructure > Analytics Nodes** to view a summary of activities for the analytics nodes; see [Figure 59 on page 184](#). See [Table 49 on page 184](#) for descriptions of the fields on the analytics summary.

Figure 59: Analytics Nodes Summary

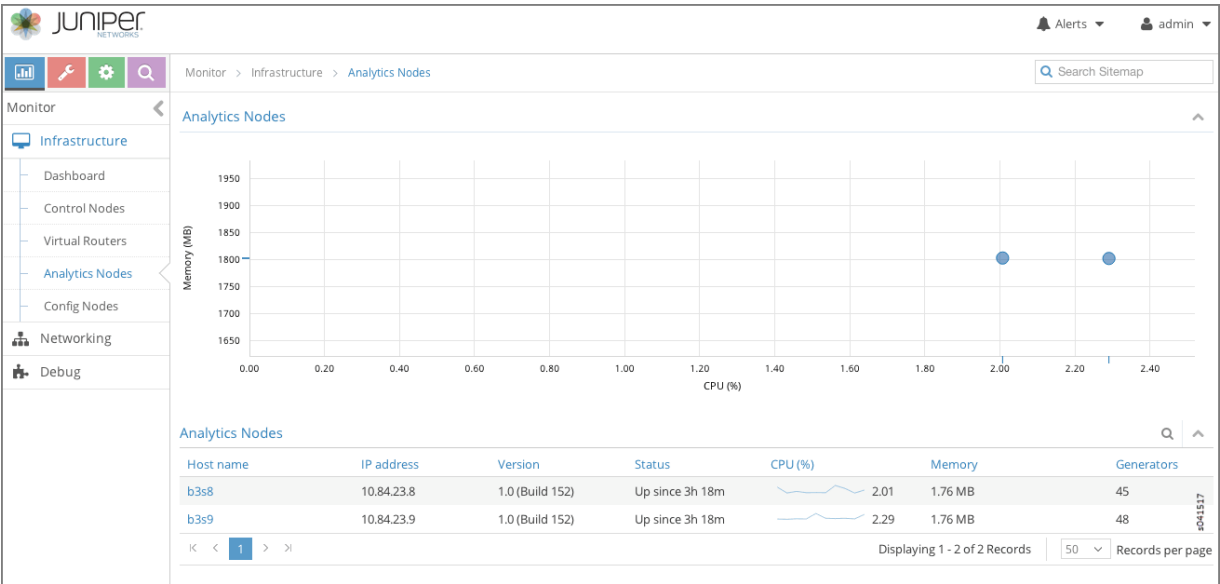


Table 49: Fields on Analytics Nodes Summary

Field	Description
Host name	The name of this node.
IP address	The IP address of this node.
Version	The version of software installed on the system.
Status	The current operational status of the node – Up or Down – and the length of time it is in that state.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage for this node.
Generators	The total number of generators for this node.

Monitor Analytics Individual Node Details Tab

Click the name of any analytics node displayed on the analytics summary to view the **Details** tab for that node. See [Figure 60 on page 185](#).

See [Table 50 on page 185](#) for descriptions of the fields on this screen.

Figure 60: Monitor Analytics Individual Node Details Tab

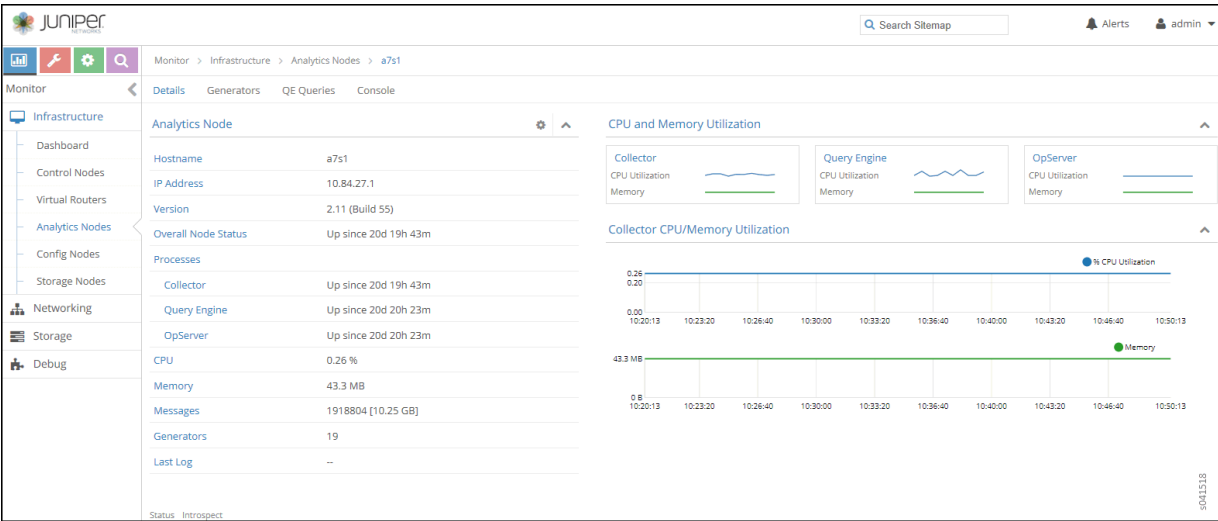


Table 50: Monitor Analytics Individual Node Details Tab Fields

Field	Description
Hostname	The name of this node.
IP Address	The IP address of this node.
Version	The installed version of the software.
Overall Node Status	The current operational status of the node – Up or Down – and the length of time in this state.
Processes	The current status of each analytics process, including Collector, Query Engine, and OpServer.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage of this node.
Messages	The total number of messages for this node.
Generators	The total number of generators associated with this node.
Last Log	The date and time of the last log message issued about this node.

Monitor Analytics Individual Node Generators Tab

The **Generators** tab displays information about the generators for an individual analytics node; see [Figure 61 on page 186](#). Click the expansion arrow next to any generator name to reveal more details. See [Table 51 on page 186](#) for descriptions of the fields on the **Peers** tab screen.

Figure 61: Individual Analytics Node—Generators Tab

Name	Status	Messages	Bytes
a7s1:Analytics:contrail-analytics-api:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	476046	1.25 GB
a7s1:Analytics:contrail-analytics-node mgr:0	Up since 20d 23h 56m, Connected since 20d 23h 16m	5	14.32 KB
a7s1:Analytics:contrail-collector:0	Up since 20d 23h 16m, Connected since 20d 23h 16m	1932437	10.25 GB
a7s1:Analytics:contrail-query-engine:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	928348	1.62 GB
a7s1:Analytics:contrail-snmp-collector:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	3	4.5 KB
a7s1:Analytics:contrail-topology:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	3	4.46 KB
a7s1:Compute:Storage-Stats-mgr:0	Up since 20d 23h 15m, Connected since 20d 23h 15m	947488	1.22 GB
a7s1:Compute:contrail-vrouter-agent:0	Up since 20d 23h 57m, Connected since 20d 23h 16m	314603	1.03 GB

Table 51: Monitor Analytics Individual Node Generators Tab Fields

Field	Description
Name	The host name of the generator.
Status	The current status of the peer— Up or Down — and the length of time in that state.
Messages	The number of messages sent and received from this peer.
Bytes	The total message size in bytes.

Monitor Analytics Individual Node QE Queries Tab

The **QE Queries** tab displays the number of query expansion (QE) messages that are in the queue for this analytics node. See [Figure 62 on page 187](#).

See [Table 52 on page 187](#) for descriptions of the fields on the **QE Queries** tab screen.

Figure 62: Individual Analytics Node—QE QueriesTab

Enqueue Time	Query	Progress
No QE Queries to display		

Table 52: Analytics Node QE Queries Tab Fields

Field	Description
Enqueue Time	The length of time this message has been in the queue waiting to be delivered.
Query	The query message.
Progress (%)	The percentage progress for the message delivery.

Monitor Analytics Individual Node Console Tab

Click the **Console** tab for an individual analytics node to display system logging information for a defined time period. See [Figure 63 on page 187](#). See [Table 53 on page 188](#) for descriptions of the fields on the **Console** tab screen.

Figure 63: Analytics Individual Node—Console Tab

Time	Category	Log Type	Log
No Records Found.			

Table 53: Monitor Analytics Individual Node Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. There are 11 options, ranging from the Last 5 mins through to the Last 24 hrs . The default display is for the Last 5 mins .
Log Category	Select a log category to display: All _default_ XMPP TCP
Log Type	Select a log type to display.
Log Level	Select a log severity level to display: SYS_EMERG SYS_ALERT SYS_CRIT SYS_ERR SYS_WARN SYS_NOTICE SYS_INFO SYS_DEBUG
Keywords	Enter any text string to search for and display logs containing that string.
(Limit field)	Select the number of messages to display: No Limit Limit 10 messages Limit 50 messages Limit 100 messages Limit 200 messages Limit 500 messages
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.

Table 53: Monitor Analytics Individual Node Console Tab Fields (continued)

Field	Description
Time	This column lists the time received for each log message displayed.
Category	This column lists the log category for each log message displayed.
Log Type	This column lists the log type for each log message displayed.
Log	This column lists the log message for each log displayed.

Monitor > Infrastructure > Config Nodes

IN THIS SECTION

- [Monitor Config Nodes | 189](#)
- [Monitor Individual Config Node Details | 190](#)
- [Monitor Individual Config Node Console | 191](#)

Select **Monitor > Infrastructure > Config Nodes** to view the information about the system config nodes.

Monitor Config Nodes

Select **Monitor > Infrastructure > Config Nodes** to view a summary of activities for the analytics nodes.
See [Figure 64 on page 190](#).

Figure 64: Config Nodes Summary

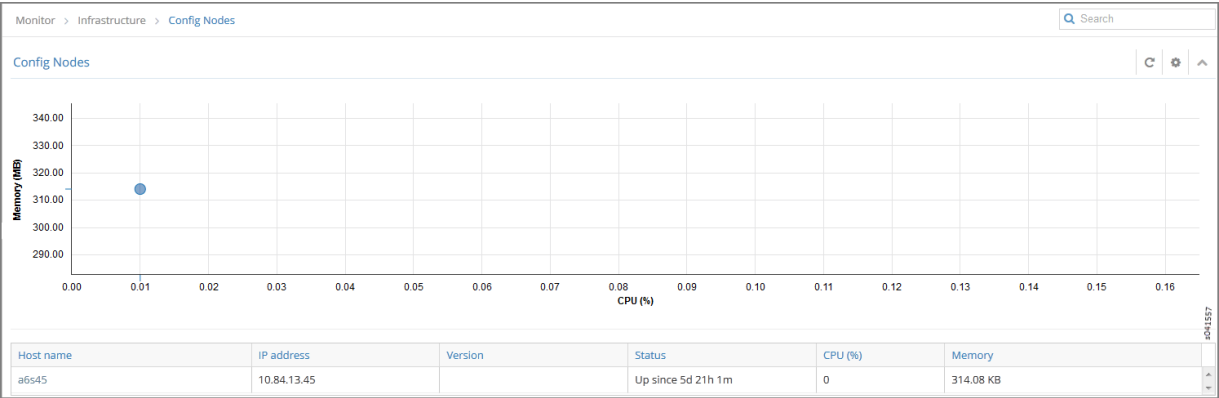


Table 54 on page 190 describes the fields in the Config Nodes summary.

Table 54: Config Nodes Summary Fields

Field	Description
Host name	The name of this node.
IP address	The IP address of this node.
Version	The version of software installed on the system.
Status	The current operational status of the node — Up or Down — and the length of time it is in that state.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage for this node.

Monitor Individual Config Node Details

Click the name of any config node displayed on the config nodes summary to view the **Details** tab for that node; see Figure 65 on page 191.

Figure 65: Individual Config Nodes— Details Tab

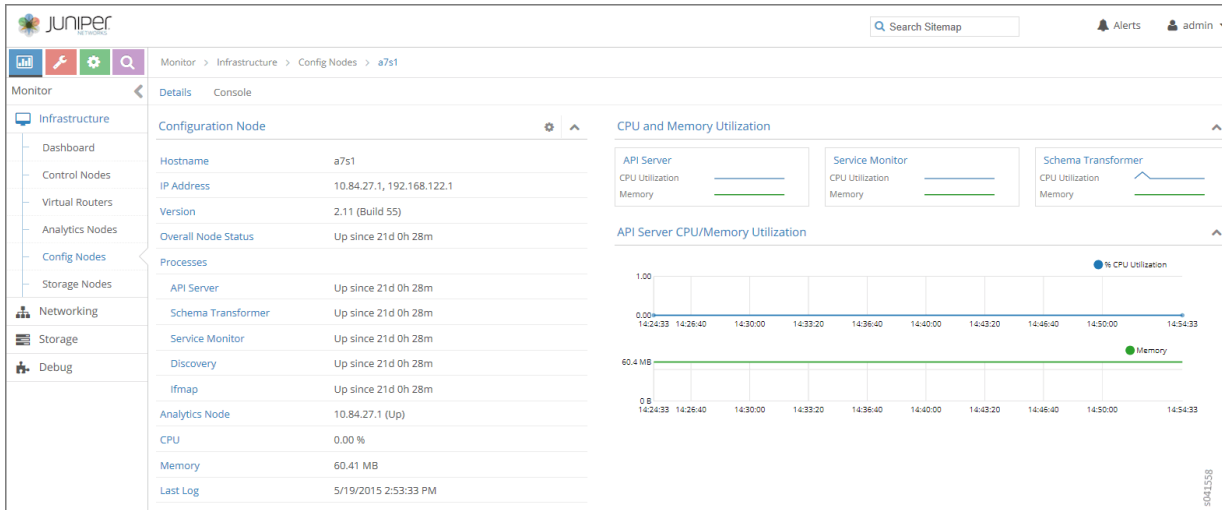


Table 55 on page 191 describes the fields on the Details screen.

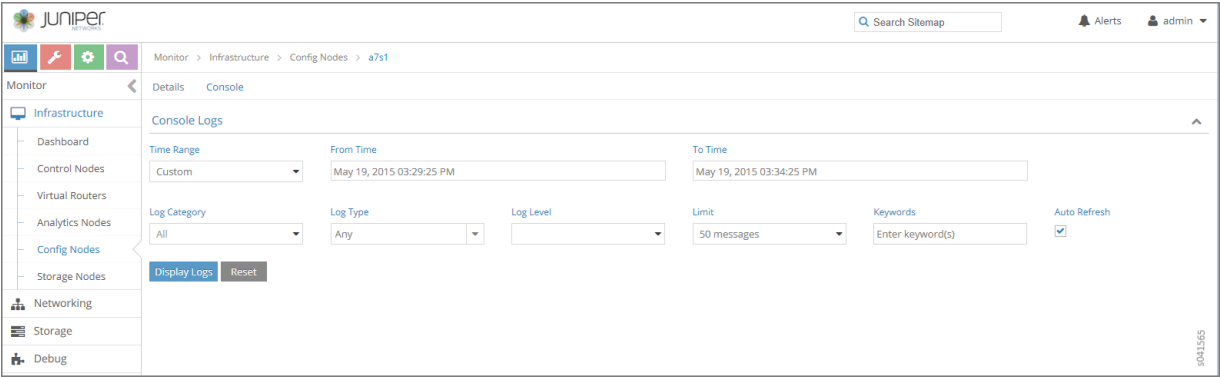
Table 55: Individual Config Nodes— Details Tab Fields

Field	Description
Hostname	The name of the config node.
IP Address	The IP address of this node.
Version	The installed version of the software.
Overall Node Status	The current operational status of the node — Up or Down — and the length of time it is in this state.
Processes	The current operational status of the processes associated with the config node, including AI Server, Schema Transformer, Service Monitor, and the like.
Analytics Node	The analytics node associated with this node.
CPU (%)	The average CPU percentage usage for this node.
Memory	The average memory usage by this node.

Monitor Individual Config Node Console

Click the **Console** tab for an individual config node to display system logging information for a defined time period. See [Figure 66 on page 192](#).

Figure 66: Individual Config Node—Console Tab



See [Table 56 on page 192](#) for descriptions of the fields on the **Console** tab screen.

Table 56: Individual Config Node-Console Tab Fields

Field	Description
Time Range	Select a timeframe for which to review logging information as sent to the console. Use the drop down calendar in the fields From Time and To Time to select the date and times to include in the time range for viewing.
Log Category	Select from the drop down menu a log category to display. The option to view All is also available.
Log Type	Select a log type to display.
Log Level	Select a log severity level to display:
Limit	Select from a list an amount to limit the number of messages displayed: <ul style="list-style-type: none"> All Limit 10 messages Limit 50 messages Limit 100 messages Limit 200 messages Limit 500 messages
Keywords	Enter any key words by which to filter the log messages displayed.
Auto Refresh	Click the check box to automatically refresh the display if more messages occur.
Display Logs	Click this button to refresh the display if you change the display criteria.

Table 56: Individual Config Node-Console Tab Fields *(continued)*

Field	Description
Reset	Click this button to clear any selected display criteria and reset all criteria to their default settings.

Monitor > Networking

IN THIS SECTION

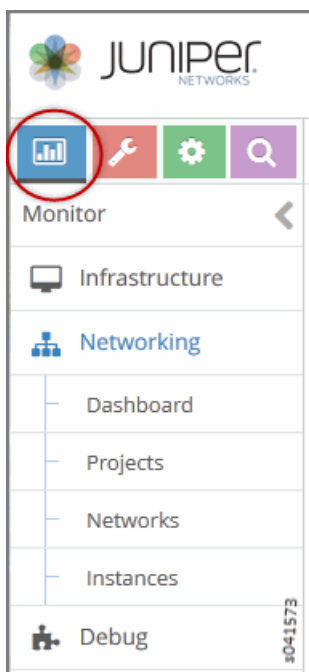
- [Monitor > Networking Menu Options | 193](#)
- [Monitor -> Networking -> Dashboard | 194](#)
- [Monitor > Networking > Projects | 196](#)
- [Monitor Projects Detail | 196](#)
- [Monitor > Networking > Networks | 199](#)

The **Monitor -> Networking** pages give an overview of the networking traffic statistics and health of domains, projects within domains, virtual networks within projects, and virtual machines within virtual networks.

Monitor > Networking Menu Options

[Figure 67 on page 194](#) shows the menu options available under **Monitor > Networking**.

Figure 67: Monitor Networking Menu Options



Monitor -> Networking -> Dashboard

Select **Monitor -> Networking -> Dashboard** to gain insight into usage statistics for domains, virtual networks, projects, and virtual machines. When you select this option, the Traffic Statistics for Domain window is displayed as shown in [Figure 68 on page 195](#).

Figure 68: Traffic Statistics for Domain Window

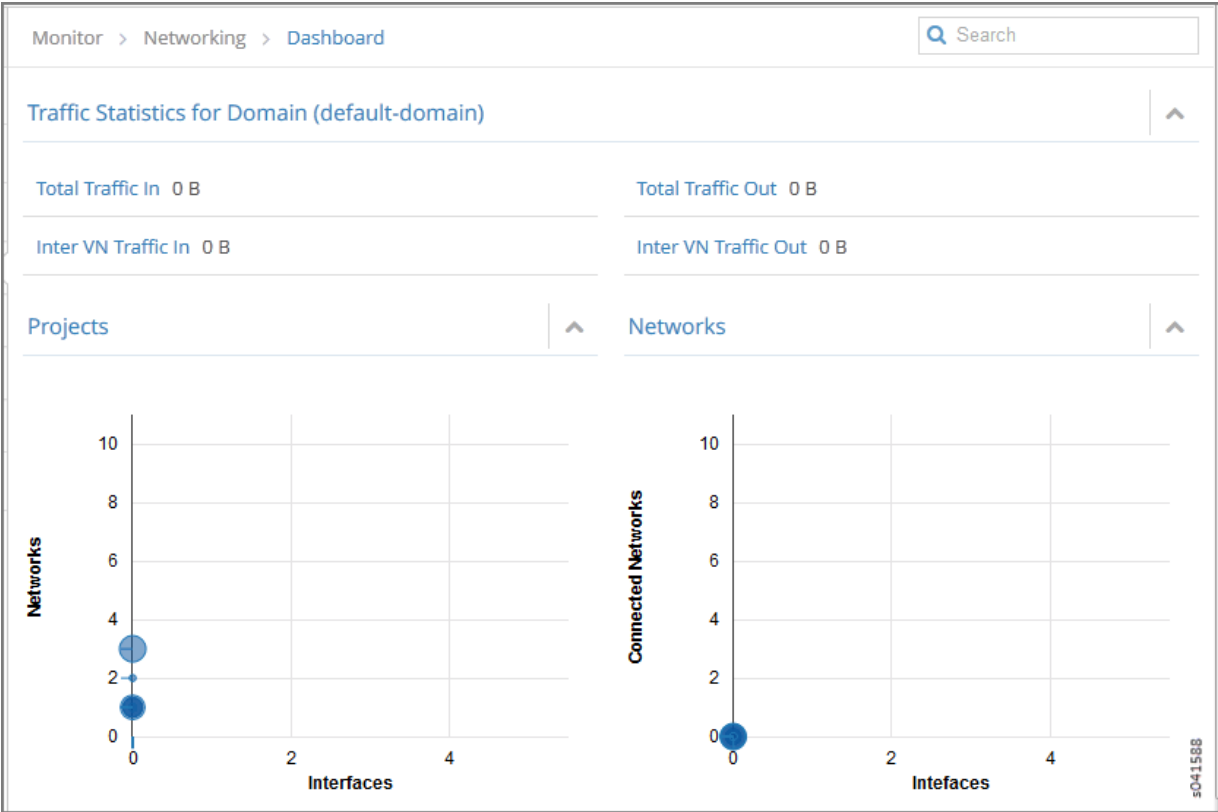


Table 57 on page 195 describes the fields in the Traffic Statistics for Domain window.

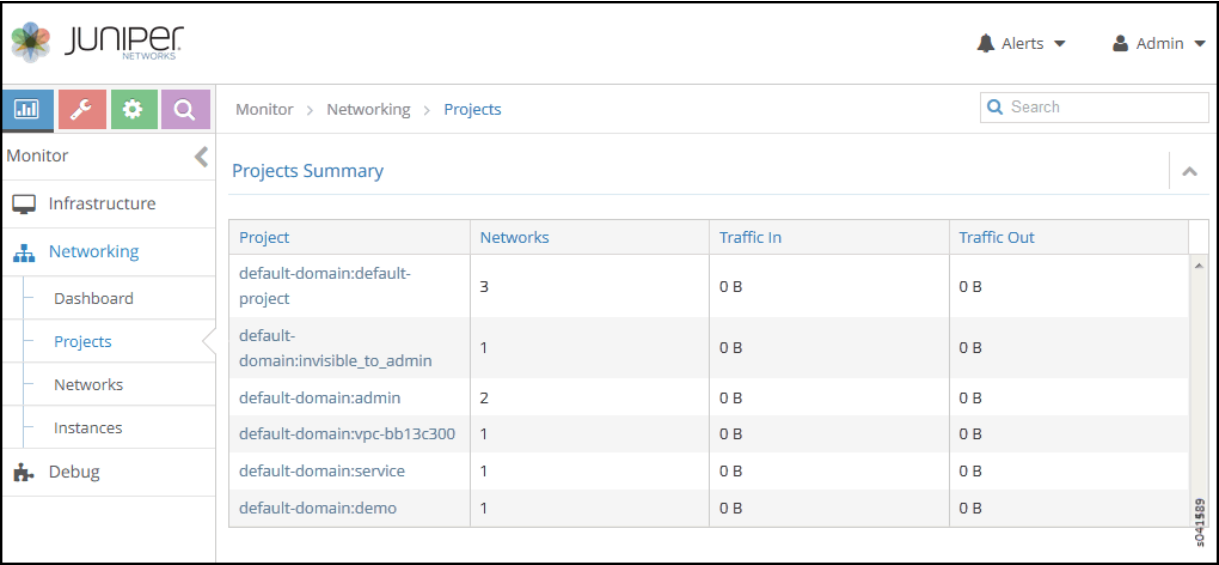
Table 57: Projects Summary Fields

Field	Description
Total Traffic In	The volume of traffic into this domain
Total Traffic Out	The volume of traffic out of this domain.
Inter VN Traffic In	The volume of inter-virtual network traffic into this domain.
Inter VN Traffic Out	The volume of inter-virtual network traffic out of this domain.
Projects	This chart displays the networks and interfaces for projects with the most throughput over the past 30 minutes. Click Projects then select Monitor > Networking > Projects , to display more detailed statistics.
Networks	This chart displays the networks for projects with the most throughput over the past 30 minutes. Click Networks then select Monitor > Networking > Networks , to display more detailed statistics.

Monitor > Networking > Projects

Select **Monitor > Networking > Projects** to see information about projects in the system. See [Figure 69 on page 196](#).

Figure 69: Monitor > Networking > Projects



See [Table 58 on page 196](#) for descriptions of the fields on this screen.

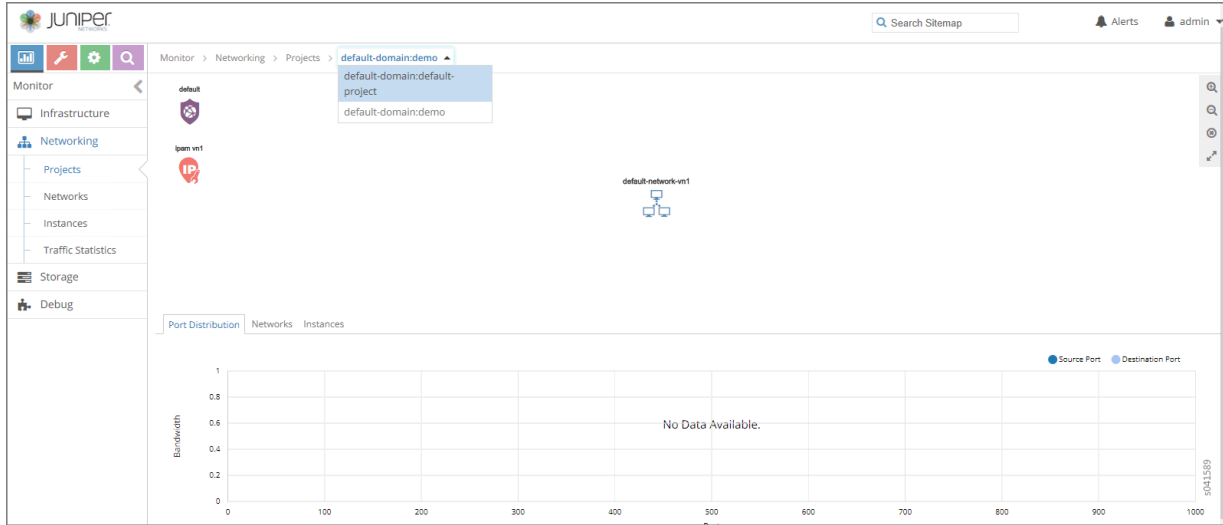
Table 58: Projects Summary Fields

Field	Description
Projects	The name of the project. You can click the name to access details about connectivity for this project.
Networks	The volume of inter-virtual network traffic out of this domain.
Traffic In	The volume of traffic into this domain.
Traffic Out	The volume of traffic out of this domain.

Monitor Projects Detail

You can click any of the projects listed on the Projects Summary to get details about connectivity, source and destination port distribution, and instances. When you click an individual project, the Summary tab for Connectivity Details is displayed as shown in [Figure 70 on page 197](#). Hover over any of the connections to get more details.

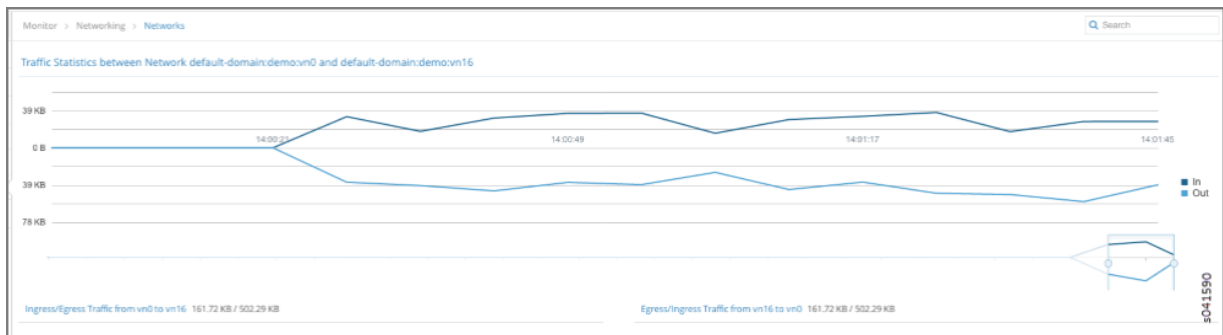
Figure 70: Monitor Projects Connectivity Details



In the Connectivity Details window you can click the links between the virtual networks to view the traffic statistics between the virtual networks.

The Traffic Statistics information is also available when you select **Monitor > Networking > Networks** as shown in [Figure 71 on page 197](#).

Figure 71: Traffic Statistics Between Networks



In the Connectivity Details window you can click the Instances tab to get a summary of details for each of the instances in this project.

Figure 72: Projects Instances Summary

	Instance	Virtual Network	Interfaces	vRouter	IP Address	Floating IP	Traffic (In/Out)
▶	out	default-domain:admin:right	1	hp1	2.2.2.252		129.87 KB / 119.83 KB
▶	NAT1_1	default-domain:admin:right	1	hp1	2.2.2.253 250.250.1.253 (1 more)		3.69 MB / 1.15 MB
▶	in	default-domain:admin:left	1	hp1	1.1.1.252		132.75 KB / 122.02 KB

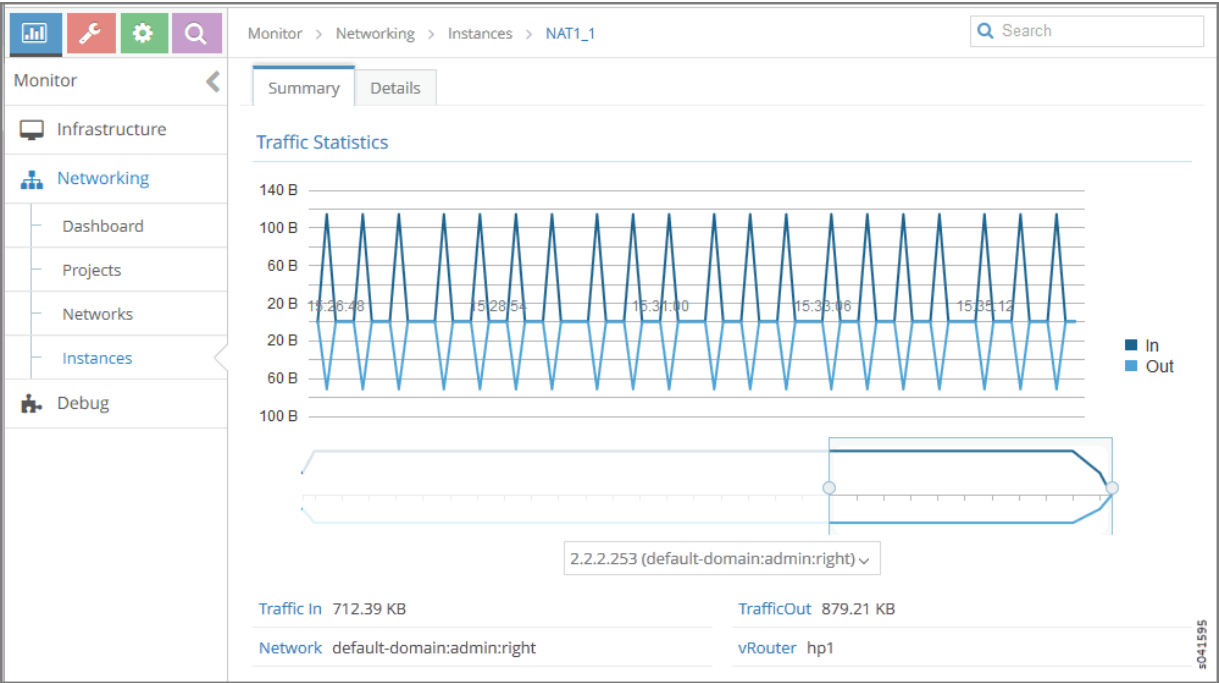
See Table 3 for a description of the fields on this screen.

Table 59: Projects Instances Summary Fields

Field	Description
Instance	The name of the instance. Click the name then select Monitor > Networking > Instances to display details about the traffic statistics for this instance.
Virtual Network	The virtual network associated with this instance.
Interfaces	The number of interfaces associated with this instance.
vRouter	The name of the vRouter associated with this instance.
IP Address	Any IP addresses associated with this instance.
Floating IP	Any floating IP addresses associated with this instance.
Traffic (In/Out)	The volume of traffic in KB or MB that is passing in and out of this instance.

Select **Monitor > Networking > Instances** to display instance traffic statistics as shown in [Figure 73 on page 199](#).

Figure 73: Instance Traffic Statistics



Monitor > Networking > Networks

Select **Monitor > Networking > Networks** to view a summary of the virtual networks in your system. See [Figure 74 on page 199](#).

Figure 74: Network Summary

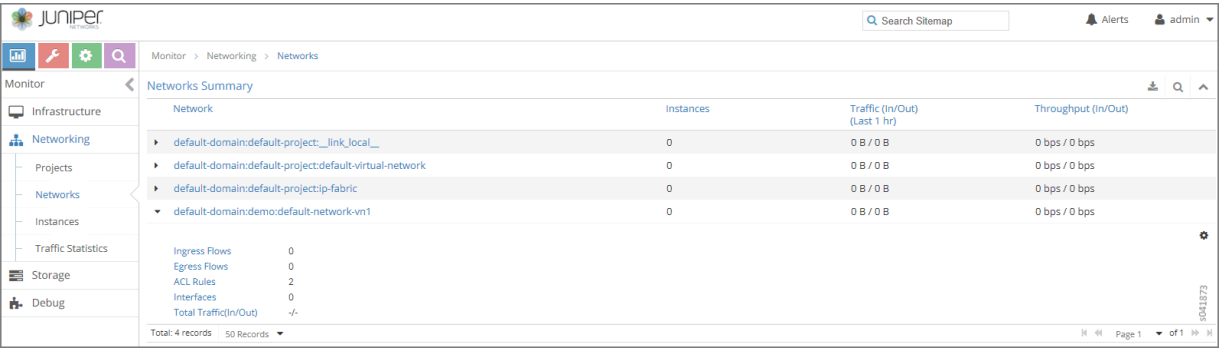


Table 60: Network Summary Fields

Field	Description
Network	The domain and network name of the virtual network. Click the arrow next to the name to display more information about the network, including the number of ingress and egress flows, the number of ACL rules, the number of interfaces, and the total traffic in and out.
Instances	The number of instances launched in this network.
Traffic (In/Out)	The volume of inter-virtual network traffic in and out of this network.
Throughput (In/Out)	The throughput of inter-virtual network traffic in and out of this network.

At **Monitor > Networking > Networks** you can click on the name of any of the listed networks to get details about the network connectivity, traffic statistics, port distribution, instances, and other details, by clicking the tabs across the top of the page.

Figure 75 on page 200 shows the **Summary** tab for an individual network, which displays connectivity details and traffic statistics for the selected network.

Figure 75: Individual Network Connectivity Details—Summary Tab

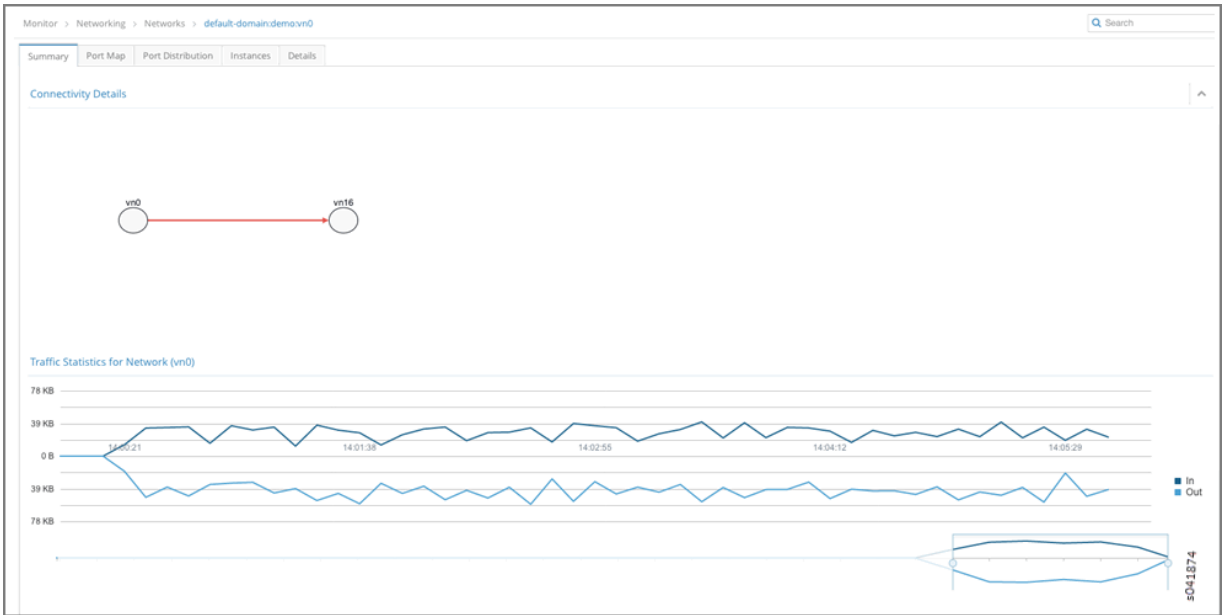


Figure 76 on page 201 shows the **Port Map** tab for an individual network, which displays the relative distribution of traffic for this network by protocol, by port.

Figure 76: Individual Network-- Port Map Tab

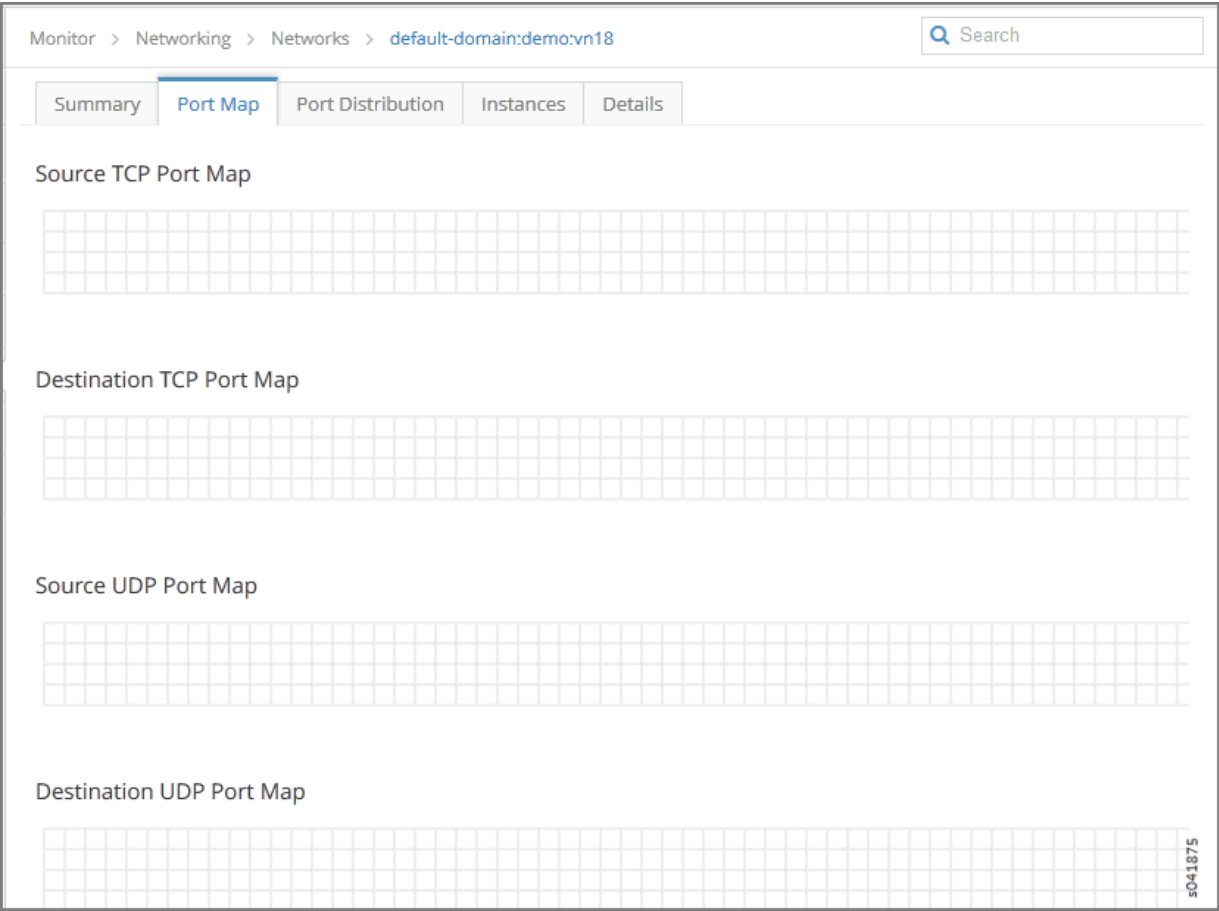


Figure 77 on page 201 shows the **Port Distribution** tab for an individual network, which displays the relative distribution of traffic in and out by source port and destination port.

Figure 77: Individual Network-- Port Distribution Tab

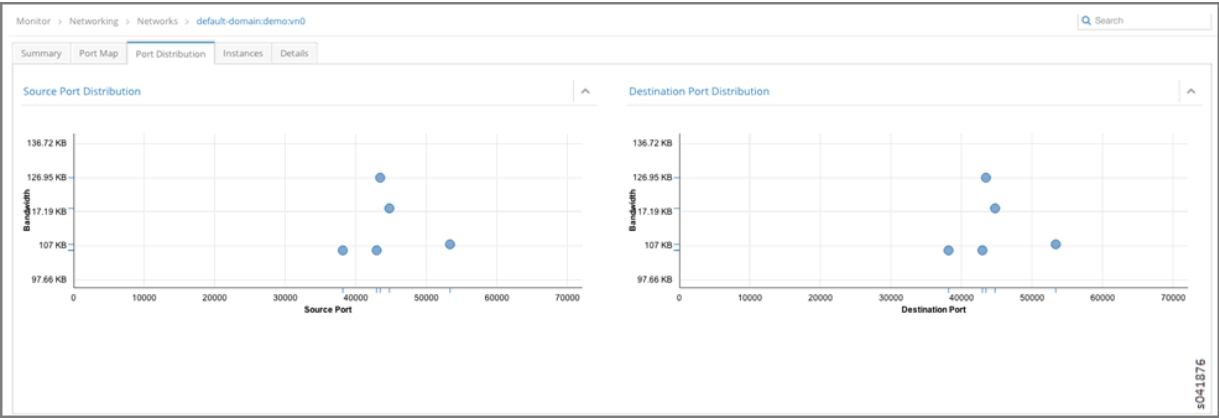


Figure 78 on page 202 shows the **Instances** tab for an individual network, which displays details for each instance associated with this network, including the number of interfaces, the associated vRouter, the instance IP address, and the volume of traffic in and out.

Additionally, you can click the arrow near the instance name to reveal even more details about the instance—the interfaces and their addresses, UUID, CPU (usage), and memory used of the total amount available.

Figure 78: Individual Network Instances Tab

Monitor > Networking > Networks > default-domain:demo:vn18

Q

Search

Summary

Port Map

Port Distribution

Instances

Details

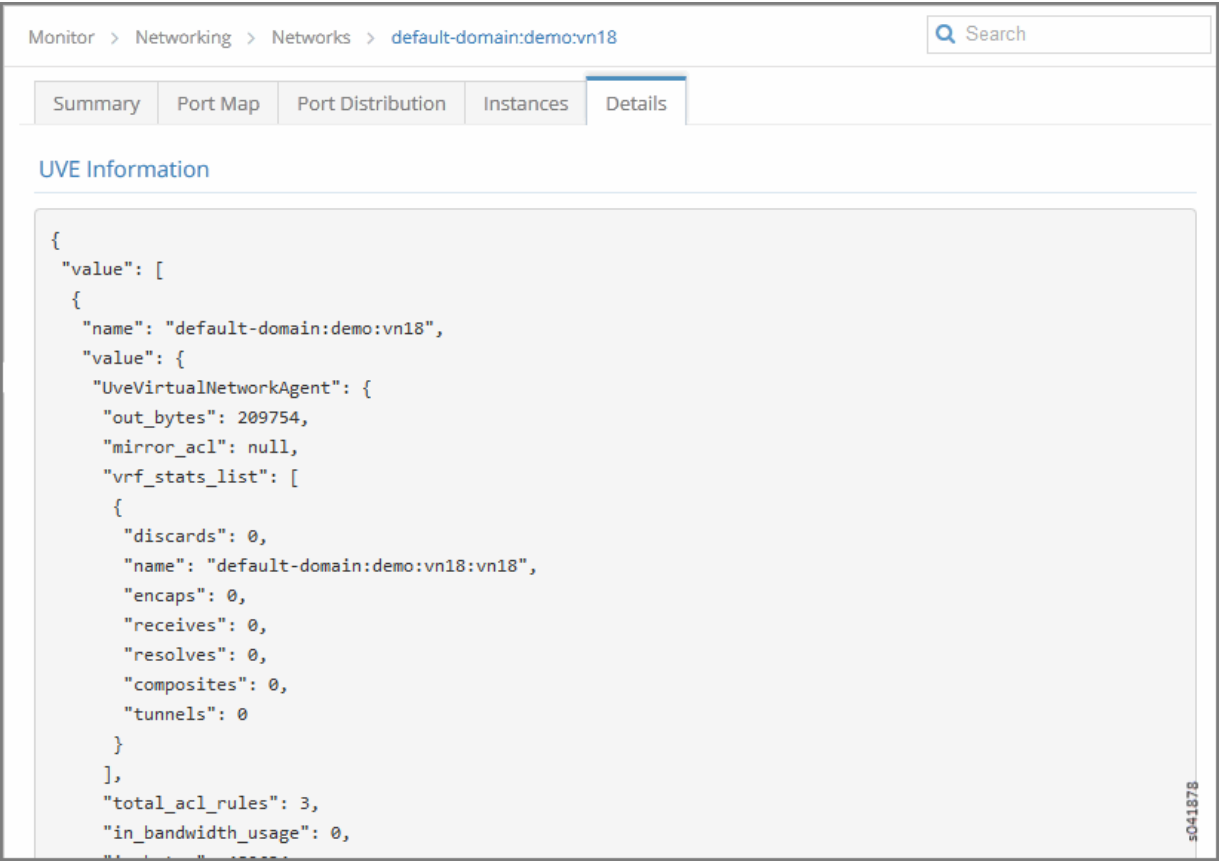
Instances Summary

	Instance	Interfaces	vRouter	IP Address	Floating IP	Traffic (In/Out)
▸	vn18_vm-b342ca93-9acd-4275-acb8-df7b5843884c	1	b1s29	192.168.18.225		1.13 KB / 712.00 B
▴	vn18_vm-22a42bf6-fccc-4db3-b5ac-80082bbefbef	1	b1s42	192.168.18.236		1.13 KB / 712.00 B
	<div> <div>Interfaces</div> <div> <div>IP Address: 192.168.18.236</div> <div>Label: 17</div> <div>Mac Address: 02:e9:94:e7:0e:56</div> <div>Network: default-domain:demo:vn18</div> <div>Traffic (In/Out): 1.13 KB/712.00 B</div> </div> </div> <div> <div>UUID</div> <div>22a42bf6-fccc-4db3-b5ac-80082bbefbef</div> </div> <div> <div>CPU</div> <div>0.01</div> </div> <div> <div>Memory (Used/Total)</div> <div>1.23 GB / 15.63 GB</div> </div>					
▸	vn18_vm-f676567a-826f-4e9d-9a81-b4649b7fcde2	1	b1s15	192.168.18.235		1.13 KB / 712.00 B

s041877

Figure 79 on page 203 shows the **Details** tab for an individual network, which displays the code used to define this network --the User Virtual Environment (UVE) code.

Figure 79: Individual Network Details Tab



Query > Flows

IN THIS SECTION

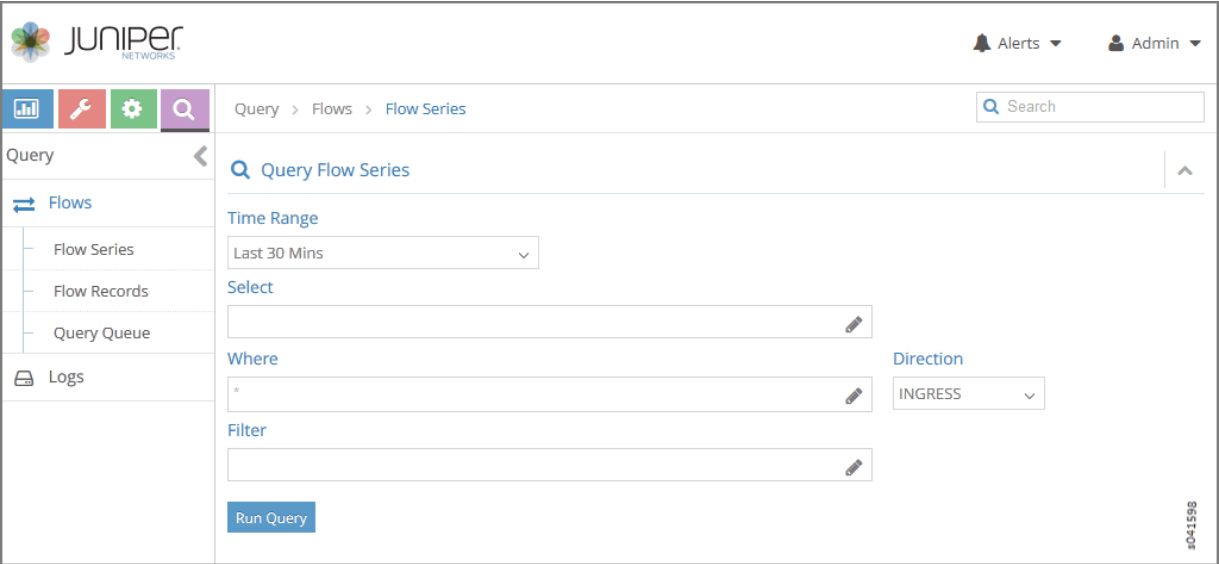
- [Query > Flows > Flow Series | 204](#)
- [Example: Query Flow Series | 206](#)
- [Query > Flow Records | 208](#)
- [Query > Flows > Query Queue | 210](#)

Select **Query > Flows** to perform rich and complex SQL-like queries on flows in the Contrail Controller. You can use the query results for such things as gaining insight into the operation of applications in a virtual network, performing historical analysis of flow issues, and pinpointing problem areas with flows.

Query > Flows > Flow Series

Select **Query > Flows > Flow Series** to create queries of the flow series table. The results are in the form of time series data for flow series. See [Figure 80 on page 204](#)

Figure 80: Query Flow Series Window



The query fields available on the screen for the **Flow Series** tab are described in [Table 61 on page 204](#). Enter query data into the fields to create a SQL-like query to display and analyze flows.

Table 61: Query Flow Series Fields

Field	Description
Time Range	Select a range of time to display the flow series: <ul style="list-style-type: none">• Last 10 Mins• Last 30 Mins• Last 1 Hr• Last 6 Hrs• Last 12 Hrs• Custom Click Custom to enter a specific custom time range in two fields: From Time and To Time .

Table 61: Query Flow Series Fields (*continued*)

Field	Description
Select	Click the edit button (pencil icon) to open a Select window (Figure 81 on page 206), where you can click one or more boxes to select the fields to display from the flow series, such as Source VN , Dest VN , Bytes , Packets , and more.
Where	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as sourcevn , sourceip , destvn , destip , protocol , sport , dport .
Direction	Select the desired flow direction: INGRESS or EGRESS .
Filter	Click the edit button (pencil icon) to open a Filter window (Figure 82 on page 206), where you can select filter items to sort by, the sort order, and limits to the number of results returned.
Run Query	Click Run Query to retrieve the flows that match the query you created. The flows are listed on the lower portion of the screen in a box with columns identifying the selected fields for each flow.
(graph buttons)	When Time Granularity is selected, you have the option to view results in graph or flowchart form. Graph buttons appear on the screen above the Export button. Click a graph button to transform the tabular results into a graphical chart display.
Export	The Export button is displayed after you click Run Query . This allows you to export the list of flows to a text .csv file.

The **Select** window allows you to select one or more attributes of a flow series by clicking the check box for each attribute desired, see [Figure 81 on page 206](#). The upper section of the **Select** window includes field names, and the lower portion lets you select units. Select **Time Granularity** and then select **SUM(Bytes)** or **SUM(Packets)** to aggregate bytes and packets in intervals.

Figure 81: Flow Series Select

Select

☐ Source VN

☐ Destination VN

☐ Time Granularity

☐ Source IP

☐ Destination IP

☐ Protocol

☐ Source Port

☐ Destination Port

☐ Virtual Router

☐ Bytes

☐ SUM(Bytes)

☐ Packets

☐ SUM(Packets)

Cancel

Apply

Use the **Filter** window to refine the display of query results for flows, by defining an attribute by which to sort the results, the sort order of the results, and any limit needed to restrict the number of results. See [Figure 82 on page 206](#).

Figure 82: Flow Series Filter

Filter

Sort By

☐ Source VN

☐ Destination VN

☐ Protocol

☐ Source IP

☐ Destination IP

☐ Virtual Router

☐ Source Port

☐ Destination Port

☐ Bytes

☐ Sum(Bytes)

☐ Packets

☐ Sum(Packets)

Sort Order

Limit By

ASC

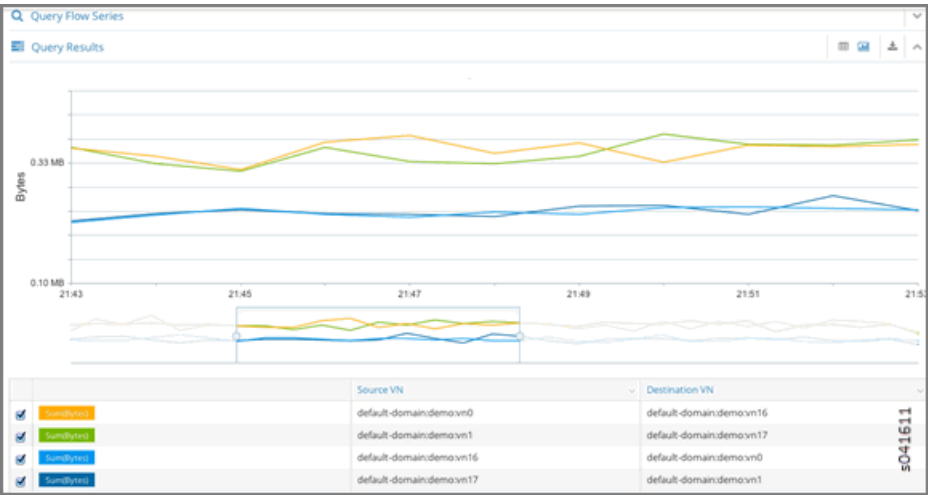
Cancel

Apply

Example: Query Flow Series

The following is an example flow series query that returns the time series of the summation traffic in bytes for all combinations of source VN and destination VN for the last 10 minutes, with the bytes aggregated in 10 second intervals. See [Figure 83 on page 207](#).

Figure 85: Query Flow Series Graphical Results



Query > Flow Records

Select **Query > Flow Records** to create queries of individual flow records for detailed debugging of connectivity issues between applications and virtual machines. Queries at this level return records of the active flows within a given time period.

Figure 86: Flow Records

The figure shows the 'Flow Records' query interface. It has a sidebar with 'Query', 'Flows', and 'Logs' sections. The 'Flows' section is expanded, showing 'Flow Series', 'Flow Records', and 'Query Queue'. The 'Flow Records' tab is selected. The main area contains a search bar, a 'Query Flow Records' title, and several input fields: 'Time Range' (set to 'Last 10 Mins'), 'Select' (empty), 'Where' (empty), and 'Direction' (set to 'INGRESS'). A 'Run Query' button is at the bottom. A vertical ID 's041601' is on the right side.

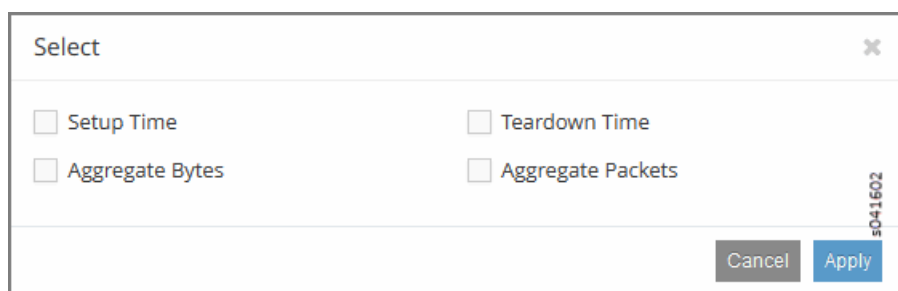
The query fields available on the screen for the **Flow Records** tab are described in [Table 62 on page 209](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 62: Query Flow Records Fields

Field	Description
Time Range	<p>Select a range of time for the flow records:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>Click Custom to enter a specified custom time range in two fields: From Time and To Time.</p>
Select	Click the edit button (pencil icon) to open a Select window (Figure 87 on page 209), where you can click one or more boxes to select attributes to display for the flow records, including Setup Time , Teardown Time , Aggregate Bytes , and Aggregate Packets .
Where	Click the edit button (pencil icon) to open a query-writing window where you can specify query values for sourcevn , sourceip , destvn , destip , protocol , sport , dport . .
Direction	Select the desired flow direction: INGRESS or EGRESS .
Run Query	Click Run Query to retrieve the flow records that match the query you created. The records are listed on the lower portion of the screen in a box with columns identifying the fields for each flow.
Export	The Export button is displayed after you click Run Query , allowing you to export the list of flows to a text .csv file.

The **Select** window allows you to select one or more attributes to display for the flow records selected, see [Figure 87 on page 209](#).

Figure 87: Flow Records Select Window

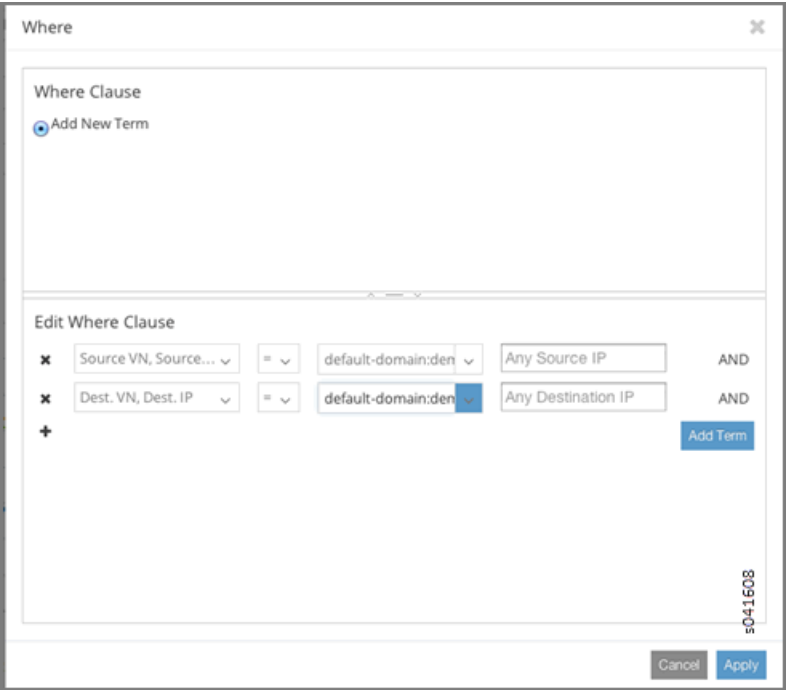


You can restrict the query to a particular source VN and destination VN combination using the **Where** section.

The **Where Clause** supports logical AND and logical OR operations, and is modeled as a logical OR of multiple AND terms. For example: ((term1 AND term2 AND term3..) OR (term4 AND term5) OR...).

Each term is a single variable expression such as **Source VN = VN1**.

Figure 88: Where Clause Window



Query > Flows > Query Queue

Select **Query > Flows > Query Queue** to display queries that are in the queue waiting to be performed on the data. See [Figure 89 on page 211](#).

Figure 89: Flows Query Queue

Query > Flows > Query Queue							Search Sitemap
Flow Query Queue							
Date	Query	Progress	Records	Status	Time Taken		
2013-10-09 18:07:06	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	150 secs		
2013-10-09 17:55:48	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	145 secs		
2013-10-09 17:29:39	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	170 secs		
2013-10-09 16:57:10	{ "table": "FlowSeriesTable", "start_time": 1381267020000000, "end_time": 1381277820000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	180	completed	270 secs		
2013-10-09 16:39:48	{ "table": "FlowSeriesTable", "start_time": 1381360140000000, "end_time": 1381361940000000, "select_fields": ["flow_class_id", "direction_ing", "T=60", "sum(bytes)"], "dir": 1 }	100%	30	completed	60 secs		
2013-10-09 11:07:29	{ "table": "FlowSeriesTable", "start_time": 1381338420000000, "end_time": 1381342020000000, "select_fields": ["flow_class_id", "direction_ing", "sum(bytes)", "T=60"], "dir": 1 }	100%	7	completed	15 secs		
Displaying 1 - 6 of 31 Records							

The query fields available on the screen for the **Flow Records** tab are described in [Table 63 on page 211](#). Enter query data into the fields to create an SQL-like query to display and analyze flows.

Table 63: Query Flow Records Fields

Field	Description
Date	The date and time the query was started.
Query	A display of the parameters set for the query.
Progress	The percentage completion of the query to date.
Records	The number of records matching the query to date.
Status	The status of the query, such as completed .
Time Taken	The amount of time in seconds it has taken the query to return the matching records.
(Action icon)	Click the Action icon and select View Results to view a list of the records that match the query, or click Delete to remove the query from the queue.

RELATED DOCUMENTATION

Understanding Flow Sampling
Fat Flows

Query > Logs

IN THIS SECTION

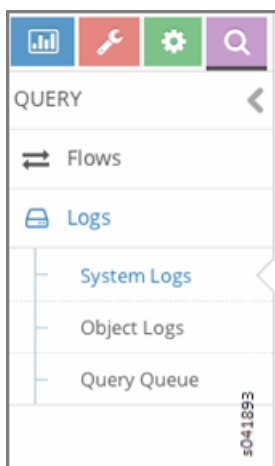
- Query > Logs Menu Options | 212
- Query > Logs > System Logs | 212
- Sample Query for System Logs | 214
- Query > Logs > Object Logs | 216

The **Query > Logs** option allows you to access the system log and object log activity of any Contrail Controller component from one central location.

Query > Logs Menu Options

Click **Query > Logs** to access the **Query Logs** menu, where you can select **System Logs** to view system log activity, **Object Logs** to view object logs activity, and **Query Queue** to create custom queries of log activity; see [Figure 90 on page 212](#).

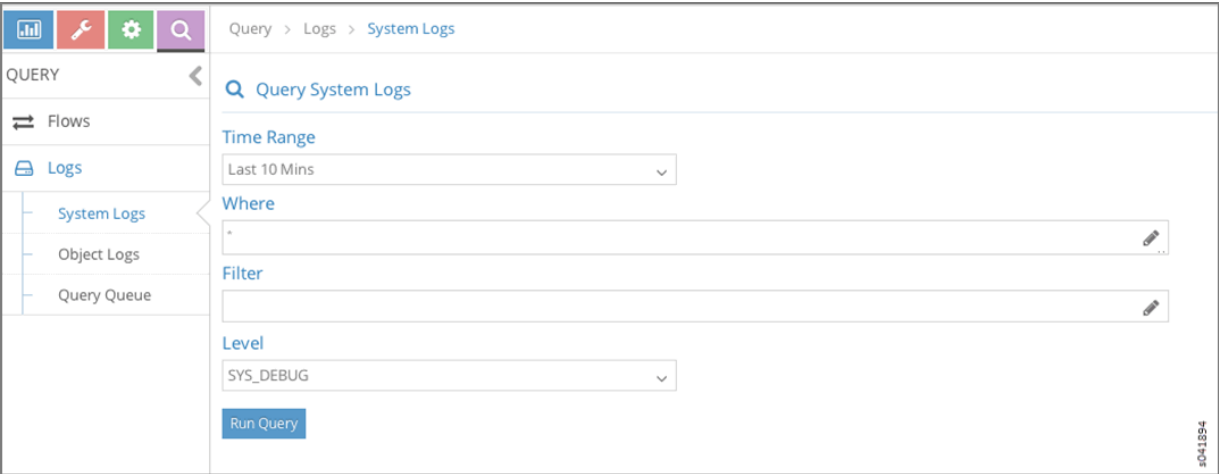
Figure 90: Query > Logs



Query > Logs > System Logs

Click **Query > Logs > System Logs** to access the **Query System Logs** menu, where you can view system logs according to criteria that you determine. See [Figure 91 on page 213](#).

Figure 91: Query > Logs > System Logs



The query fields available on the **Query System Logs** screen are described in [Table 64 on page 213](#).

Table 64: Query System Logs Fields

Field	Description
Time Range	Select a range of time for which to see the system logs: <ul style="list-style-type: none">• Last 10 Mins• Last 30 Mins• Last 1 Hr• Last 6 Hrs• Last 12 Hrs• Custom If you click Custom, enter a desired time range in two new fields: From Time and To Time .
Where	Click the edit button (pencil icon) to open a query-writing window, where you can specify query values for variables such as Source, Module, MessageType, and the like, in order to retrieve specific information.

Table 64: Query System Logs Fields (*continued*)

Field	Description
Level	<p>Select the message severity level to view:</p> <ul style="list-style-type: none"> • SYS_NOTICE • SYS_EMERG • SYS_ALERT • SYS_CRIT • SYS_ERR • SYS_WARN • SYS_INFO • SYS_DEBUG
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the Time , Source , Module Id , Category , Log Type , and Log message.
Export	This button appears after you click Run Query , allowing you to export the list of system messages to a text/csv file.

Sample Query for System Logs

This section shows a sample system logs query designed to show all **System Logs** from **ModuleId = VRouterAgent** on **Source = b1s16** and filtered by **Level = SYS_DEBUG**.

1. At the **Query System Logs** screen, click in the **Where** field to access the **Where** query screen and enter information defining the location to query in the **Edit Where Clause** section and click **OK**; see [Figure 92 on page 215](#).

Figure 92: Edit Where Clause

Where

Where Clause

Add New Term

Edit Where Clause

×

ModuleId

▼

=

▼

VRouterAgent

▼

AND

×

Source

▼

=

▼

b1s16

▼

AND

+

Add Term

OK

Cancel

sc041895

- 2. The information you defined at the Where screen displays on the **Query System Logs**. Enter any more defining information needed; see [Figure 93 on page 216](#). When finished, click **Run Query** to display the results.

Figure 93: Sample Query System Logs

Q Query System Logs

Time Range

Last 10 Mins

Where

(ModuleId = VRouterAgent AND Source = b1s16)

Filter

Level

SYS_DEBUG

Run Query

9041896

Query > Logs > Object Logs

Object logs allow you to search for logs associated with a particular object, for example, all logs for a specified virtual network. Object logs record information related to modifications made to objects, including creation, deletion, and other modifications; see [Figure 94 on page 216](#).

Figure 94: Query > Logs > Object Logs

Q Query Object Logs

Time Range

Last 12 Hrs

Object Type

Virtual Network

Object Id

default-domain:demo:vn14

Select

ObjectLog, SystemLog

Where

*

Filter

Run Query

9041897

The query fields available on the **Object Logs** screen are described in [Table 65 on page 217](#).

Table 65: Object Logs Query Fields

Field	Description
Time Range	<p>Select a range of time for which to see the logs:</p> <ul style="list-style-type: none"> • Last 10 Mins • Last 30 Mins • Last 1 Hr • Last 6 Hrs • Last 12 Hrs • Custom <p>If you click Custom, enter a desired time range in two new fields: From Time and To Time.</p>
Object Type	<p>Select the object type for which to show logs:</p> <ul style="list-style-type: none"> • Virtual Network • Virtual Machine • Virtual Router • BGP Peer • Routing Instance • XMPP Connection
Object Id	Select from a list of available identifiers the name of the object you wish to use.
Select	<p>Click the edit button (pencil icon) to open a window where you can select searchable types by clicking a checkbox:</p> <ul style="list-style-type: none"> • ObjectLog • SystemLog
Where	Click the edit button (pencil icon) to open the query-writing window, where you can specify query values for variables such as Source , ModuleId , and MessageType , in order to retrieve information as specific as you wish.
Run Query	Click this button to retrieve the system logs that match the query. The logs are listed in a box with columns showing the Time , Source , Module Id , Category , Log Type , and Log message.
Export	This button appears after you click Run Query , allowing you to export the list of system messages to a text/csv file.

Common Support Answers

IN THIS CHAPTER

- Debugging Ping Failures for Policy-Connected Networks | 218
- Debugging BGP Peering and Route Exchange in Contrail | 225
- Troubleshooting the Floating IP Address Pool in Contrail | 242
- Removing Stale Virtual Machines and Virtual Machine Interfaces | 273
- Troubleshooting Link-Local Services in Contrail | 278

Debugging Ping Failures for Policy-Connected Networks

This topic presents troubleshooting scenarios and steps for resolving reachability issues (ping failures) when working with policy-connected virtual networks.

These are the methods used to configure reachability for a virtual network or virtual machine:

- Use network policy to exchange virtual network routes.
- Use a floating IP address pool to associate an IP address from a destination virtual network to virtual machine(s) in the source virtual network.
- Use an ASN/RT configuration to exchange virtual network routes with an MX Series router gateway.
- Use a service instance static route configuration to route between service instances in two virtual networks.

This topic focuses on troubleshooting reachability for the first method --- using network policy to exchange routes between virtual networks.

Troubleshooting Procedure for Policy-Connected Network

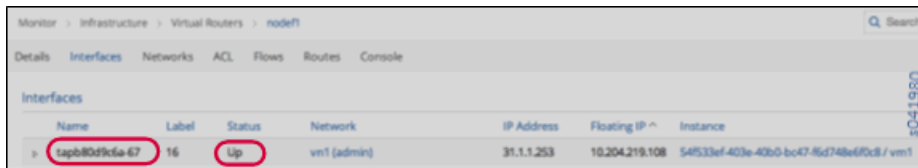
1. Check the state of the virtual machine and interface.

Before doing anything else, check the status of the source and destination virtual machines.

- Is the **Status** of each virtual machine **Up**?
- Are the corresponding tap interfaces **Active**?

Check the virtual machine status in the Contrail UI:

Figure 95: Virtual Machine Status Window



Name	Label	Status	Network	IP Address	Floating IP ^	Instance
tapb88d9c6a-67	16	Up	vn1 (admin)	31.1.1.253	19.204.219.108	54533ef-403e-40b0-bc47-6d748e60c8 / vn1

Check the tap interface status in the http agent introspect, for example:

http://nodet1.englab.juniper.net:8085/Snh_ItfReq?name=

Figure 96: Tap Interface Status Window



index	name	uuid	vrf_name	active
4	tapb88d9c6a-67	b88d9c6a-672e-4c1e-9b83-e053d6c9110b	default-domain:admin:vn1:vn1	Active

When the virtual machine status is verified **Up**, and the tap interface is **Active**, you can focus on other factors that affect traffic, including routing, network policy, security policy, and service instances with static routes.

2. Check reachability and routing.

Use the following troubleshooting guidelines whenever you are experiencing ping failures on virtual network routes that are connected by means of network policy.

Check the network policy configuration:

- Verify that the policy is attached to each of the virtual networks.
- Each attached policy should have either an explicit rule allowing traffic from one virtual network to the other, or an allow all traffic rule.
- Verify that the order of the actions in the policy rules is correct, because the actions are applied in the order in which they are listed.
- If there are multiple policies attached to a virtual network, verify that the policies are attached in a logical order. The first policy listed is applied first, and its rules are applied first, then the next policy is applied.
- Finally, if either of the virtual networks does not have an explicit rule to allow traffic from the other virtual network, the traffic flow will be treated as an **UNRESOLVED** or **SHORT** flow and all packets will be dropped.

Use the following sequence in the Contrail UI to check policies, attachments, and traffic rules:

Check VN1-VN2 ACL information from the compute node:

Figure 97: Policies, Attachments, and Traffic Rule Status Window

UUID	Flows	Action	Protocol	Source Network or Prefix	Source Port	Destination Network or Prefix	Destination Port	ACE Id
8b0329d7-a29e-41ac-a2e-30f48c2b5ae	100000	pass	1 - 1	default-domainadminvn1	any	default-domainadminvn2	any	1
		pass	1 - 1	default-domainadminvn2	any	default-domainadminvn1	any	2
		pass	any	default-domainadminvn1	any	default-domainadminvn1	any	3
		deny	any	default-domainadminvn1	any	default-domainadminvn2	any	4
		deny	any	default-domainadminvn2	any	default-domainadminvn1	any	5

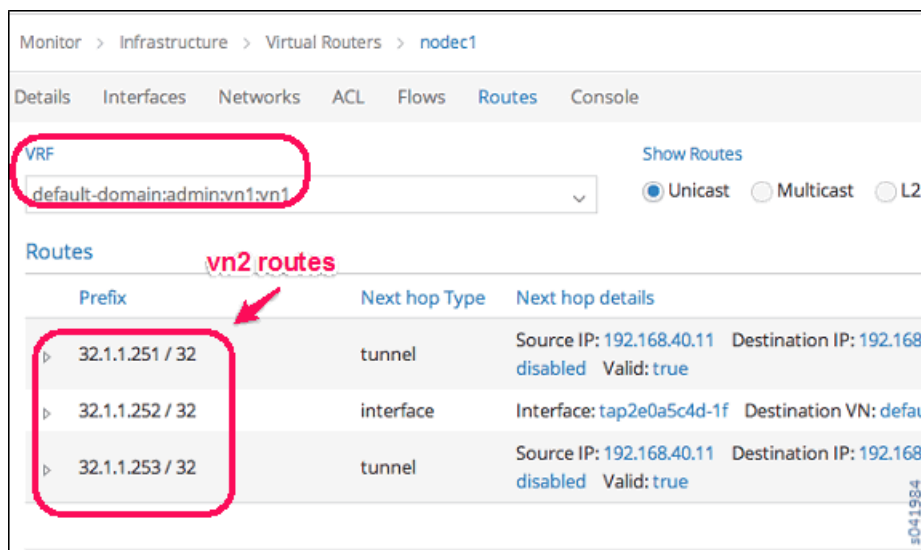
Check the virtual network policy configuration with route information:

Figure 98: Virtual Network Policy Configuration Window

Network	Attached Policies	IP Blocks
vn1	default-analyzer-analyzer-policy vn1-vn2	31.1.1.0/24
vn2	allow_all	32.1.1.0/24

Check the VN1 route information for VN2 routes:

Figure 99: Virtual Network Route Information Window



If a route is missing, ping fails. Flow inspection in the compute node displays **Action: D(drop)**.

Repeated dropstats commands confirms the drop by incrementing the **Flow Action Drop** counter with each iteration of dropstats.

Flow and dropstats commands issued at the compute node:

Figure 100: Flow and Dropstats Command List

```

root@nodefl1:~# flow -l | grep 32.1.1 -A1
root@nodefl1:~# flow -l | grep 32.1.1 -A1
root@nodefl1:~# flow -l | grep 32.1.1 -A1
73348          32.1.1.252:1911          31.1.1.253:0          1 (1)
                (Action:D, S(nh):0, Statistics:0/0)
--
423404          31.1.1.253:1911          32.1.1.252:0          1 (1)
                (Action:D, S(nh):8, Statistics:1/84)
root@nodefl1:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1588
root@nodefl1:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1589
root@nodefl1:~# dropstats | grep "Flow Action Drop "
Flow Action Drop          1590
root@nodefl1:~#

```

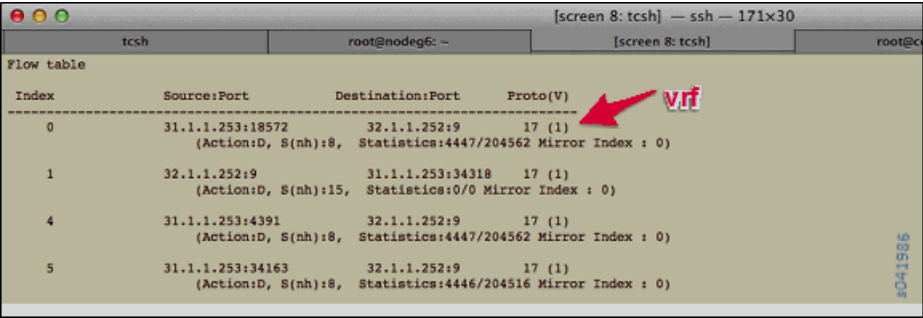
To help in debugging flows, you can use the detailed flow query from the agent introspect page for the compute node.

Fields of interest include:

- Inputs [from **flow -l** output]: **src/dest ip**, **src/dest ports**, **protocol**, and **vrf**
- Output from detailed flow query: **short_flow**, **src_vn**, **action_str->action**

Flow command output:

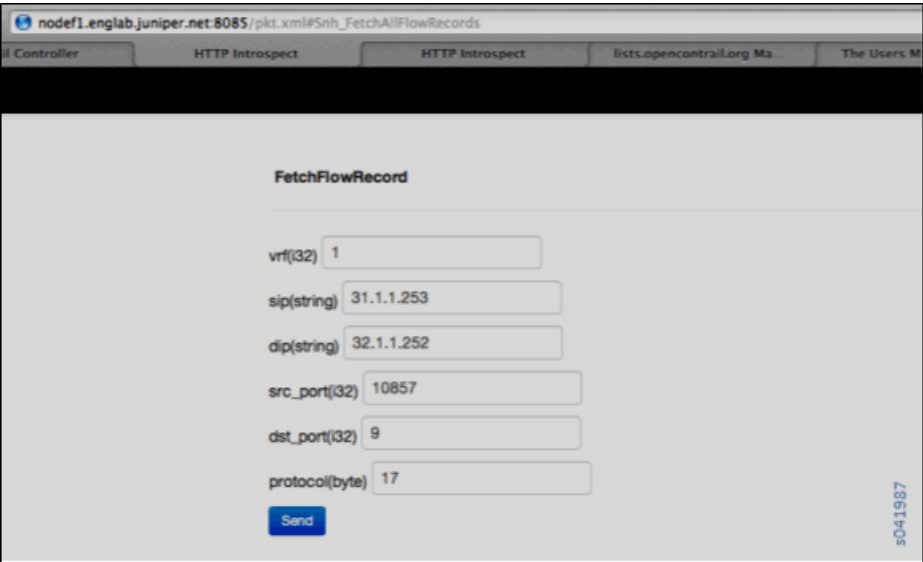
Figure 101: Flow Command Output Window



Index	Source:Port	Destination:Port	Proto(V)
0	31.1.1.253:18572 (Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0)	32.1.1.252:9	17 (1) vrf
1	32.1.1.252:9 (Action:D, S(nh):15, Statistics:0/0 Mirror Index : 0)	31.1.1.253:34318	17 (1)
4	31.1.1.253:4391 (Action:D, S(nh):8, Statistics:4447/204562 Mirror Index : 0)	32.1.1.252:9	17 (1)
5	31.1.1.253:34163 (Action:D, S(nh):8, Statistics:4446/204516 Mirror Index : 0)	32.1.1.252:9	17 (1)

Fetching details of a single flow:

Figure 102: Fetch Flow Record Window



nodef1.englab.juniper.net:8085/pkt.xml#Snh_FetchAllFlowRecords

Controller HTTP Introspect HTTP Introspect lists.opencontrail.org Ma... The Users M

FetchFlowRecord

vrf(32)

sip(string)

dip(string)

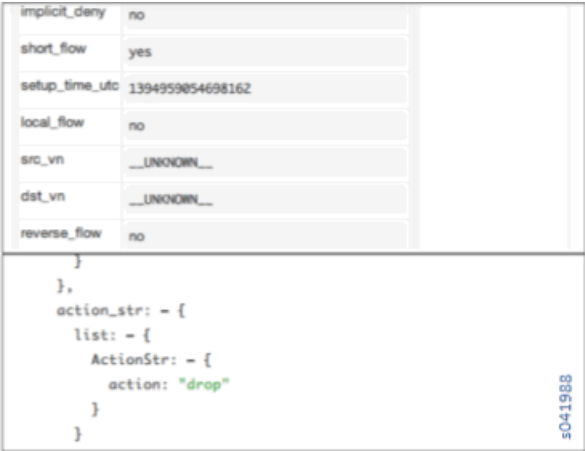
src_port(32)

dst_port(32)

protocol(byte)

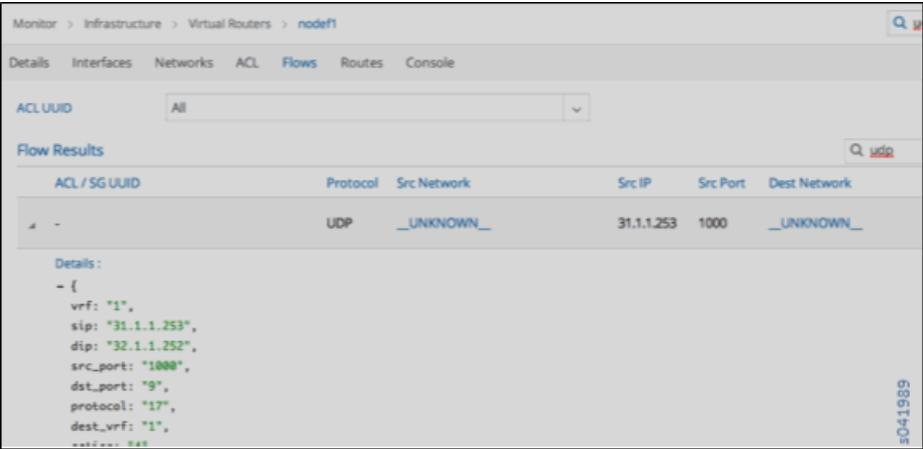
Output from **FetchFlowRecord** shows unresolved IP addresses:

Figure 103: Unresolved IP Address Window



You can also retrieve information about unresolved flows from the Contrail UI, as shown in the following:

Figure 104: Unresolved Flow Details Window

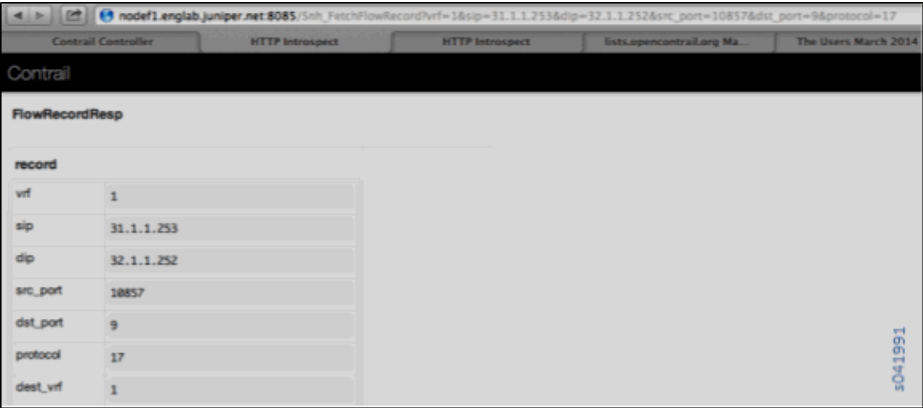


3. Check for protocol-specific network policy action.

If you are still experiencing reachability issues, troubleshoot any protocol-specific action, where routes are exchanged, but only specific protocols are allowed.

The following shows a sample query on a protocol-specific flow in the agent introspect:

Figure 105: Protocol-Specific Flow Sample



The following shows that although the virtual networks are resolved (not **__UNKNOWN__**), and not a short flow (the flow entry exists for a defined aging time), the policy action clearly displays **deny** as the action.

Figure 106: Protocol-Specific Flow Sample With Deny Action



Summary

This topic explores one area —debugging for policy-based routing. However, in a complex system, a virtual network might have one or more configuration methods combined that influence reachability and routing.

For example, an environment might have a virtual network VN-X configured with policy-based routing to another virtual network VN-Y. At the same time, there are a few virtual machines in VN-X that have a floating IP to another virtual network VN-Z, which is connected to VN-XX via a NAT service instance. This

is a complex scenario, and you need to debug step-by-step, taking into account all of the features working together.

Additionally, there are other considerations beyond routing and reachability that can affect traffic flow. For example, the rules of network policies and security groups can affect traffic to the destination. Also, if multi-path is involved, then ECMP and RPF need to be taken into account while debugging.

Debugging BGP Peering and Route Exchange in Contrail

IN THIS SECTION

- [Example Cluster | 225](#)
- [Verifying the BGP Routers | 226](#)
- [Verifying the Route Exchange | 228](#)
- [Debugging Route Exchange with Policies | 231](#)
- [Debugging Peering with an MX Series Router | 232](#)
- [Debugging a BGP Peer Down Error with Incorrect Family | 234](#)
- [Configuring MX Peering \(iBGP\) | 237](#)
- [Checking Route Exchange with an MX Series Peer | 239](#)
- [Checking the Route in the MX Series Router | 240](#)

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail BGP peering and route exchange.

Example Cluster

Examples in this document refer to a virtual cluster that is set up as follows:

```
Config Nodes   : [ 'nodea22', 'nodea20' ]  
  
Control Nodes : [ 'nodea22', 'nodea20' ]  
  
Compute Nodes : [ 'nodea22', 'nodea20' ]  
  
Collector     : [ 'nodea22' ]
```

```
WebU : nodea22

Openstack : nodea22
```

Verifying the BGP Routers

Use this procedure to launch various introspects to verify the setup of the BGP routers in your system.

Use this procedure to launch various introspects to verify the setup of the BGP routers in your system.

1. Verify the BGP routers.

All of the configured control nodes and external BGP routers are visible from the following location, shown using the sample node setup.

http: //<host ip address>:8082/bgp-routers

NOTE: Throughout this procedure, replace <host ip address> with the correct location for your system to see the setup in your system.

Figure 107: Sample Output, BGP Routers:

```
{
  - bgp-routers: [
    - {
      href: "http://nodea22.englab.juniper.net:8082/bgp-router/1da579c5-0907-4c98-a7ad-37671f00cf60",
      - fq_name: [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "nodea20"
      ],
      uuid: "1da579c5-0907-4c98-a7ad-37671f00cf60"
    },
    - {
      href: "http://nodea22.englab.juniper.net:8082/bgp-router/9702853f-5e48-417f-bd72-c00a12cc0200",
      - fq_name: [
        "default-domain",
        "default-project",
        "ip-fabric",
        "__default__",
        "nodea22"
      ],
      uuid: "9702853f-5e48-417f-bd72-c00a12cc0200"
    }
  ]
}
```

Sample Output

2. Verify the BGP peering.

The following statement is entered to check the **bgp_router_refs** object on the API server to validate the peering on the sample setup.

http: //<host ip address>:8082/bgp-router/1da579c5-0907-4c98-a7ad-37671f00cf60

Figure 108: Sample Output, BGP Router References:

```
- bgp_router_parameters: {
  vendor: "contrail",
  autonomous_system: 64512,
  vnc_managed: null,
  address: "10.204.216.16",
  identifier: "10.204.216.16",
  port: 179,
  address_families: {
    - family: {
      "inet-vpn",
      "e-vpn"
    }
  },
  href: "http://nodes22.englab.juniper.net:8082/bgp-router/9702853f-5e48-417f-bd72-c00a12cc0200",
  attr: {
    - session: {
      - attributes: {
        - {
          bgp_router: null,
          address_families: {
            - family: {
              "inet-vpn",
              "e-vpn"
            }
          }
        }
      },
      uuid: null
    }
  },
  uuid: "9702853f-5e48-417f-bd72-c00a12cc0200"
},
},
},
```

3. Verify the command line arguments that are passed to the control-node.

On the control-node, use `ps aux | grep control-node` to see the arguments that are passed to the control-node.

Example

```
/usr/bin/control-node --map-user <ip address> --map-password <ip address>--hostname
nodea22 --host-ip <ip address> --bgp-port 179 --discovery-server <ip address>
```

The hostname is the **bgp-router** name. Ensure that the bgp-router config can be found for the hostname, using the procedure in Step 1.

4. Validate the BGP neighbor config and the BGP peering config object.

`http://<host ip address>:8083/Snh_ShowBgpNeighborConfigReq?`

Figure 109: Sample Output, BGP Neighbor Config:

ShowBgpNeighborConfigResp						
neighbors						
instance_name	name	vendor	autonomous_system	identifier	address	address_families
default-domain:default-project:ip-fabric:___default___	default-domain:default-project:ip-fabric:___default___:nodea22	contrail	64512	10.204.216.16	10.204.216.16	address_families
						inet-vpn
						e-vpn

http: //<host ip address>:8083/Snh_ShowBgpPeeringConfigReq?

Figure 110: Sample Output, BGP Peering Config:

instance_name	name	neighbor_count	sessions
default-domain:default-project:ip-fabric:default...	attr(default-domain:default-project:ip-fabric:default...;nodes28,default-domain:default-project:ip-fabric:default...;nodes22)	1	<ul style="list-style-type: none"> attributes bgp_router address_families inet-vpn e-vpn

5. Check the BGP neighbor states on the sample setup.

http: //<host ip address>:8083/Snh_BgpNeighborReq?ip_address=&domain=

Figure 111: Sample Output, BGP Neighbor States:

peer	peer_address	peer_asn	local_address	local_asn	encoding	peer_type	state	send_state	last_event	last_state	last_state_at	last_error
10.204.216.16	10.204.216.16	64512	10.204.216.18	64512	BGP	Internal	Established	In sync	fan::EvBgpKeepalive	OpenConfirm	2014-Feb-10 07:08:21.177692	Unknown
10.204.216.253	10.204.216.253	64512	10.204.216.18	64512	BGP	Internal	Established	In sync	fan::EvBgpUpdate	OpenConfirm	2014-Feb-10 11:24:45.855632	Cease:Administrator reset the peer

If the peer is not in an established state, check the **last_error** and the **flap_count**. Debug the BGP state machine by using information displayed in the output, such as **last_state** and **last_event**.

NOTE: The image displayed is truncated to fit this page. On the console screen you can scroll horizontally to see more columns and data.

Verifying the Route Exchange

The following two virtual networks are used in the sample debugging session for route exchange.

```
vn1 -> 1.1.1.0/24

vn2 -> 2.2.2.0/24
```

Example Procedure for Verifying Route Exchange

- 1. Validate the presence of the routing instance for each virtual network in the sample system.

http ://<host ip address>:8083/Snh_ShowRoutingInstanceReq?name=

NOTE: Throughout this example, replace **<host ip address>** with the correct location for the control node on your system.

Figure 112: Sample Output, Show Routing Instance:

default-domain:demo:vn1:vn1	default-domain:demo:vn1	4	import_target	export_target	tables	peers
			target:64512:1	target:64512:1		
					default-domain:demo:vn1:vn1.inet.0	peers
						nodes28
					default-domain:demo:vn1:vn1.lnet.0	peers
						nodes28
					default-domain:demo:vn1:vn1.lnetcast.0	peers
						nodes28
default-domain:demo:vn2:vn2	default-domain:demo:vn2	5	import_target	export_target	tables	peers
			target:64512:2	target:64512:2		
					default-domain:demo:vn2:vn2.inet.0	peers
						nodes22
					default-domain:demo:vn2:vn2.lnet.0	peers
						nodes22
					default-domain:demo:vn2:vn2.lnetcast.0	peers
						nodes22

In the sample output, you can see the **import_target** and the **export_target** configured on the routing instance. Also shown are the **xmpp peers (vroutes)** registered to the table.

The user can click on the **inet** table of the required routing instance to display the routes that belong to the instance.

Use the information in Step 2 to validate a route.

- 2. Validate a route in a given routing instance in the sample setup:

http ://<host ip address>:8083/Snh_ShowRouteReq?x=default-domain:demo:vn1:vn1.inet.0

In the following sample output (truncated), the user can validate the BGP paths for the protocol and for the source of the route to verify which XMPP agent or vRouter has pushed the route. If the path source is BGP, the route is imported to the VRF table from a BGP peer, either another control-node or an external bgp router such as an MX Series router. BGP paths are displayed in the order of path selection.

The label (MPLS) and tunnel encap columns can be used for debugging data path issues.

Figure 115: Sample Output, Validate L3vpn Table, Scrolled:

source	as_path	next_hop	tunnel_encap	label	replicated	primary_table	communities	origin_vn	flags
node20	-	10.204.216.16	tunnel_encap	16	true	default-domain:demo:vn1:inet.0	communities	default-domain:demo:vn1	0
			gre				security group: 5		
			udp				originvn:64512:4		
							target:64512:1		
10.204.216.16	-	10.204.216.16	tunnel_encap	16	false		communities	default-domain:demo:vn1	0
			gre				security group: 5		
			udp				originvn:64512:4		
							target:64512:1		

source	as_path	next_hop	tunnel_encap	label	replicated	primary_table	communities	origin_vn	flags
node22	-	10.204.216.18	tunnel_encap	16	true	default-domain:demo:vn2:inet.0	communities	default-domain:demo:vn2	0
			gre				security group: 5		
			udp				originvn:64512:5		
							target:64512:2		
10.204.216.16	-	10.204.216.18	tunnel_encap	16	false		communities	default-domain:demo:vn2	0
			gre				security group: 5		
			udp				originvn:64512:5		
							target:64512:2		

Debugging Route Exchange with Policies

This section uses the sample output and the sample vn1 and vn2 to demonstrate methods of debugging route exchange with policies.

- 1. Create a network policy to allow vn1 and vn2 traffic and associate the policy to the virtual networks.

Figure 116: Create Policy Window

Create Policy

Policy Name

any_any

Policy Rules

Action	Protocol	Source Network	Source Ports	Direction	Destination Network	Destination Ports	Apply Service	Mirror to	
PAS	ANY	default-domain:	Source	<>	default-domain:	Destinat			

Cancel Save

- 2. Validate that the routing instances have the correct import_target configuration.

http: //<host ip address>:8083/Snh_ShowRoutingInstanceReq?name=

Figure 117: Sample Output, Validate Import Target:

default-domain:demo:vn1:vn1	default-domain:demo:vn1	4	import_target target:64512:1 target:64512:2	export_target target:64512:1
default-domain:demo:vn2:vn2	default-domain:demo:vn2	5	import_target target:64512:1 target:64512:2	export_target target:64512:2

3. Validate that the routes are imported from VRF.
- Use the BGP path attribute to check the replication status of the path. The route from the destination VRF should be replicated and validate the origin-vn.

Figure 118: Sample Output, Route Import:

ShowRouteResp									
tables									
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes		
default-domain:demo:vn2:vn2	default-domain:demo:vn2:vn2:inet.0	2	4	1	3	0	prefix	last_modified	paths
							1.1.1.253/32	2014-Feb-18 12:02:47.261344	protocol XMPP
									BGP
							2.2.2.253/32	2014-Feb-18 11:34:35.469899	protocol XMPP
									BGP

Debugging Peering with an MX Series Router

This section sets up an example BGP MX Series peer and provides some troubleshooting scenarios.

1. Set the Global AS number of the control-node for an MX Series BGP peer, using the Contrail WebUI (eBGP).

Figure 119: Edit Global ASN Window

2. Configure the eBGP peer for the MX Series router. Use the Contrail Web UI or Python provisioning.

Figure 120: Create BGP Peer Window

Configuring the MX Series BGP peer with the Python provision utility:

```
python ./provision_mx.py --router_name mx --router_ip <ip address> --router_asn
12345 --api_server_ip <ip address> --api_server_port 8082 --oper add --admin_user
admin --admin_password <password> --admin_tenant_name admin
```

3. Configure a control-node peer on the MX Series router, using Junos CLI:

```
set protocols bgp group contrail-control-nodes type external

set protocols bgp group contrail-control-nodes local-address <ip address>

set protocols bgp group contrail-control-nodes keep all

set protocols bgp group contrail-control-nodes peer-as 54321
```

```
set protocols bgp group contrail-control-nodes local-as 12345

set protocols bgp group contrail-control-nodes neighbor <ip address>
```

Debugging a BGP Peer Down Error with Incorrect Family

Use this procedure to identify and resolve errors that arise from *families* mismatched configurations.

NOTE: This example uses locations at **http://<host ip address>:**. Be sure to replace **<host ip address>** with the correct address for your environment.

1. Check the BGP peer UVE.

http://<host ip address>:8081/analytics/uves/bgp-peers

2. Search for the MX Series BGP peer by name in the list.

In the sample output, **families** is the family advertised by the peer and **configured_families** is what is provisioned. In the sample output, the families configured on the peer has a mismatch, thus the peer doesn't move to an established state. You can verify it in the peer UVE.

Figure 121: Sample BGP Peer UVE

```
{
  - BgpPeerInfoData: {
    - state_info: {
      last_state: "Idle",
      state: "Idle",
      last_state_at: 1394778927107639
    },
    - families: [
      "IPv4:Unicast"
    ],
    peer_type: "external",
    local_asn: 54321,
    - configured_families: [
      "inet-vpn"
    ],
    - event_info: {
      last_event_at: 1394778927107880,
      last_event: "fsm::EvStart"
    },
    local_id: 181196816,
    send_state: "not advertising",
    peer_address: "10.204.216.253",
    peer_id: 181197053,
    hold_time: 90,
    peer_asn: 12345
  }
}
```

3. Fix the **families** mismatch in the sample by updating the configuration on the MX Series router, using Junos CLI:

```
set protocols bgp group contrail-control-nodes family inet-vpn unicast
```

4. After committing the CLI configuration, the peer comes up. Verify this with UVE.

```
http: //<host ip address>:8081/analytics/uves/bgp-peers
```

Figure 122: Sample Established BGP Peer UVE

```
{
- BgpPeerInfoData: {
- state_info: {
  last_state: "OpenConfirm",
  state: "Established",
  last_state_at: 1394779652932460
},
- families: [
  "IPv4:Vpn"
],
peer_type: "external",
local_asn: 54321,
- configured_families: [
  "inet-vpn"
],
- event_info: {
  last_event_at: 1394779652992071,
  last_event: "fsm::EvBgpUpdate"
},
local_id: 181196816,
send_state: "in sync",
peer_address: "10.204.216.253",
peer_id: 181197053,
peer_asn: 12345
}
}
```

5. Verify the peer status on the MX Series router, using Junos CLI:

```
run show bgp neighbor <ip address>
```

```
Peer: <ip address> AS 54321 Local: <ip address> AS 12345
```

```
Type: External      State: Established      Flags: <ImportEval Sync>
```

```
Last State: OpenConfirm  Last Event: RecvKeepAlive
```

```
Last Error: None
```

```
Options: <Preference LocalAddress KeepAll AddressFamily PeerAS LocalAS Rib-group
Refresh>
```

```
Address families configured: inet-vpn-unicast
```

```
Local Address: <ip address> Holdtime: 90 Preference: 170 Local AS: 12345 Local
```

System AS: 64512

Number of flaps: 0

Error: 'Cease' Sent: 0 Recv: 2

Peer ID: <ip address> Local ID: <ip address> Active Holdtime: 90

Keepalive Interval: 30 Group index: 1 Peer index: 0

BFD: disabled, down

Local Interface: ge-1/0/2.0

NLRI for restart configured on peer: inet-vpn-unicast

NLRI advertised by peer: inet-vpn-unicast

NLRI for this session: inet-vpn-unicast

Peer does not support Refresh capability

Stale routes from peer are kept for: 300

Peer does not support Restarter functionality

Peer does not support Receiver functionality

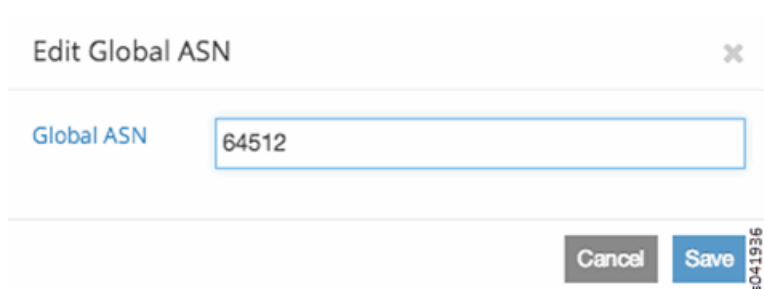
Peer does not support 4 byte AS extension

Peer does not support Addpath

Configuring MX Peering (iBGP)

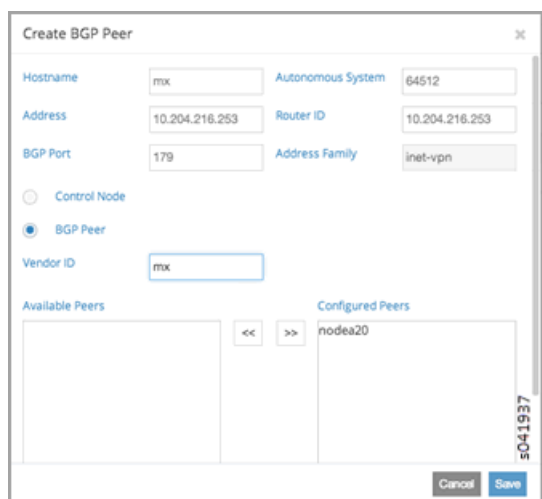
1. Edit the Global ASN.

Figure 123: Edit Global ASN Window



2. Configure the MX Series IBGP peer, using Contrail WebUI or Python provisioning.

Figure 124: Create BGP Peer Window



Configuring the MX Series BGP peer with the Python provision utility:

```
python ./provision_mx.py --router_name mx--router_ip <ip address> --router_asn 64512 --api_server_ip
<ip address> --api_server_port 8082 --oper add --admin_user admin --admin_password <password>
--admin_tenant_name admin
```

3. Verify the peer from UVE.

<http://<host ip address>:8081/analytics/uves/bgp-peers>

Figure 125: Sample Established IBGP Peer UVE

```
{
  - BgpPeerInfoData: {
    - state_info: {
      last_state: "OpenConfirm",
      state: "Established",
      last_state_at: 1394788178225128
    },
    - families: {
      "IPv4:Vpn"
    },
    peer_type: "internal",
    local_asn: 64512,
    - configured_families: {
      "inet-vpn"
    },
    - event_info: {
      last_event_at: 1394788178267208,
      last_event: "fsm::EvBgpUpdate"
    },
    local_id: 181196816,
    send_state: "in sync",
    peer_address: "10.204.216.253",
    peer_id: 181197053,
    peer_asn: 64512
  }
}
```

4. You can verify the same information at the HTTP introspect page of the control node (8443 in this example).

http://<host ip address>:8083/Snh_BgpNeighborReq?ip_address=&domain=

Figure 126: Sample Established IBGP Peer Introspect Window

neighbors											
peer	peer_address	peer_asn	local_address	local_asn	encoding	peer_type	state	send_state	last_event	last_state	
10.204.216.253	10.204.216.253	64512	10.204.216.16	64512	BGP	internal	Established	in sync	fsm::EvBgpKeepalive	OpenConfirm	

Checking Route Exchange with an MX Series Peer

4. Launch a virtual machine in the public network and verify the route in the public.inet.0 table.
- http: //<host ip address>:8083/ Snh_ShowRouteReq?x=default-domain:admin:public:public.inet.0**

Figure 130: Virtual Machine Routing Instance Public IPv4 Route Table

tables									
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes		
default-domain:admin:public:public	default-domain:admin:public:public.inet.0	3	3	1	2	0	routes		
							prefix	last_modified	paths
							0.0.0.0/0	2014-Mar-14 10:05:05.719526	paths
							protocol		
							BGP		
							10.204.218.0/24	2014-Mar-14 10:05:05.720617	paths
							protocol		
							BGP		
							11.2.3.253/32	2014-Mar-14 10:18:48.797958	paths
							protocol		
							XXPP		

5. Verify the route in the bgp.l3vpn.0 table.
- http: //<host ip address>:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0**

Figure 131: BGP Routing Instance Route Table

tables									
routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	infeasible_paths	routes		
default-domain:default-project:ip-fabric:default...	bgp.l3vpn.0	3	3	2	1	0	routes		
							prefix		
							10.204.216.253:5:0.0.0.0/0		
							10.204.216.253:5:10.204.218.0/24		
							10.204.216.70:1:11.2.3.253/32		

Checking the Route in the MX Series Router

Use Junos CLI show commands from the router to check the route.

```
run show route table public.inet.0

public.inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
```

+ = Active Route, - = Last Active, * = Both

0.0.0.0/0 *[Static/5] 15w6d 08:50:34

> to <ip address> via ge-1/0/1.0

<ip address> *[Direct/0] 15w6d 08:50:35

> via ge-1/0/1.0

<ip address> *[Local/0] 15w6d 08:50:51

Local via ge-1/0/1.0

<ip address> *[BGP/170] 01:13:34, localpref 100, from <ip address>

AS path: ?, validation-state: unverified

> via gr-1/0/0.32771, Push 16

[BGP/170] 01:13:34, localpref 100, from <ip address>

AS path: ?, validation-state: unverified

> via gr-1/0/0.32771, Push 16

<ip address> *[BGP/170] 00:03:20, localpref 100, from <ip address>

AS path: ?, validation-state: unverified

> via gr-1/0/0.32769, Push 16

run show route table bgp.l3vpn.0 receive-protocol bgp <ip address> detail

bgp.l3vpn.0: 92 destinations, 130 routes (92 active, 0 holddown, 0 hidden)

* <ip address> (1 entry, 0 announced)

Import Accepted

```

Route Distinguisher: <ip address>

VPN Label: 16

Nexthop: <ip address>

Localpref: 100

AS path: ?

Communities: target:64512:1 target:64512:10003 unknown iana 30c unknown iana
30c unknown type 8004 value fc00:1 unknown type 8071 value fc00:4

```

Troubleshooting the Floating IP Address Pool in Contrail

IN THIS SECTION

- [Example Cluster | 243](#)
- [Example | 244](#)
- [Example: MX80 Configuration for the Gateway | 245](#)
- [Ping the Floating IP from the Public Network | 248](#)
- [Troubleshooting Details | 248](#)
- [Get the UUID of the Virtual Network | 249](#)
- [View the Floating IP Object in the API Server | 249](#)
- [View floating-ips in floating-ip-pools in the API Server | 254](#)
- [Check Floating IP Objects in the Virtual Machine Interface | 257](#)
- [View the BGP Peer Status on the Control Node | 261](#)
- [Querying Routes in the Public Virtual Network | 262](#)
- [Verification from the MX80 Gateway | 264](#)
- [Viewing the Compute Node Vns Agent | 266](#)
- [Advanced Troubleshooting | 270](#)

This document provides troubleshooting methods to use when you have errors with the floating IP address pool when using Contrail.

Example Cluster

Examples in this document refer to a virtual cluster that is set up as follows:

```
Config Nodes   : ['nodec6', 'nodec7', 'nodec8']

Control Nodes  : ['nodec7', 'nodec8']

Compute Nodes  : ['nodec9', 'nodec10']

Collector      : ['nodec6', 'nodec8']

WebUI          : nodec7

Openstack      : nodec6
```

The following virtual networks are used in the examples in this document:

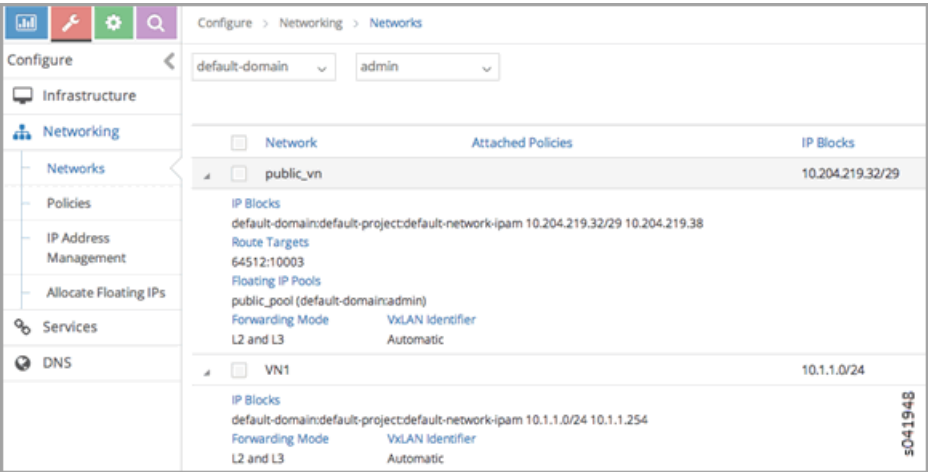
Public virtual network:

- Virtual network name: **public_vn**
- Public addresses range: **10.204.219.32 to 10.204.219.37**
- Route Target: **64512:10003**
- Floating IP pool name: **public_pool**

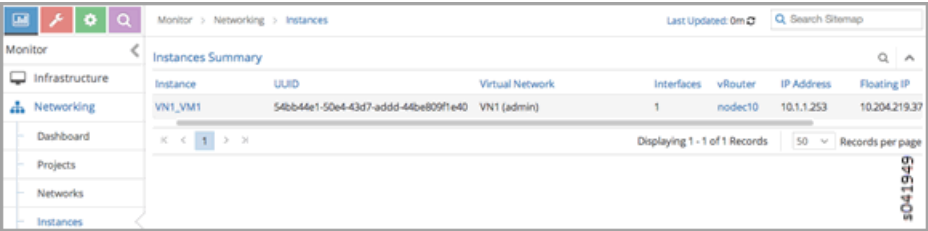
Private virtual network:

- Virtual network name: **vn1**
- Subnet: **10.1.1.0/24**

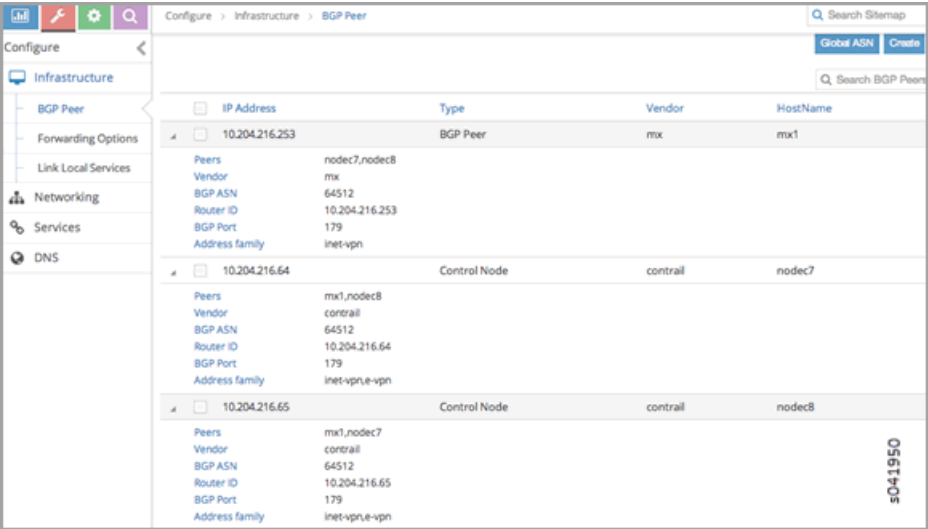
Example



A virtual machine is created in the virtual network VN1 with the name VN1_VM1 and with the IP address 10.1.1.253. A floating IP address of 10.204.219.37 is associated to the VN1_VM1 instance.



An MX80 router is configured as a gateway to peer with control nodes nodec7 and nodec8.



Example: MX80 Configuration for the Gateway

The following is the Junos OS configuration for the MX80 gateway. The route 10.204.218.254 is the route to the external world.

```
chassis {  
    fpc 1 {  
        pic 0 {  
            tunnel-services;  
        }  
    }  
}  
  
interfaces {  
    ge-1/0/1 {  
        unit 0 {  
            family inet {  
                address 10.204.218.1/24;  
            }  
        }  
    }  
  
    ge-1/0/2 {  
        unit 0 {  
            family inet {  
                address 10.204.216.253/24;  
            }  
        }  
    }  
}
```

```

    }

}

}

routing-options {

    static {

        route 0.0.0.0/0 next-hop 10.204.216.254;

    }

    router-id 10.204.216.253;

    route-distinguisher-id 10.204.216.253;

    autonomous-system 64512;

    dynamic-tunnels {

        tun1 {

            source-address 10.204.216.253;

            gre;

            destination-networks {

                10.204.216.0/24;

                10.204.217.0/24;

            }

        }

    }

}

}

protocols {

```

```
bgp {  
  
    group control-nodes {  
  
        type internal;  
  
        local-address 10.204.216.253;  
  
        keep all;  
  
        family inet-vpn {  
  
            unicast;  
  
        }  
  
        neighbor 10.204.216.64;  
  
        neighbor 10.204.216.65;  
  
    }  
  
}  
  
routing-instances {  
  
    public {  
  
        instance-type vrf;  
  
        interface ge-1/0/1.0;  
  
        vrf-target target:64512:10003;  
  
        vrf-table-label;  
  
        routing-options {  
  
            static {  
  
                route 0.0.0.0/0 next-hop 10.204.218.254;  
  
            }  
  
        }  
  
    }  
  
}
```



```

    }

  }

}

}

```

Ping the Floating IP from the Public Network

From the public network, ping the floating IP 10.204.219.37.

```

user1-test:~ user1$ ping 10.204.219.37

PING 10.204.219.37 (10.204.219.37): 56 data bytes

64 bytes from 10.204.219.37: icmp_seq=0 ttl=54 time=62.439 ms

64 bytes from 10.204.219.37: icmp_seq=1 ttl=54 time=56.018 ms

64 bytes from 10.204.219.37: icmp_seq=2 ttl=54 time=55.915 ms

64 bytes from 10.204.219.37: icmp_seq=3 ttl=54 time=57.755 ms

^C

--- 10.204.219.37 ping statistics ---

5 packets transmitted, 4 packets received, 20.0% packet loss

round-trip min/avg/max/stddev = 55.915/58.032/62.439/2.647 ms

```

Troubleshooting Details

The following sections show details of ways to get related information, view, troubleshoot, and validate floating IP addresses in a Contrail system.

Get the UUID of the Virtual Network

Use the following to get the universal unique identifier (UUID) of the virtual network.

```
[root@nodec6 ~]# (source /etc/contrail/openstackrc; quantum net-list -F id -F name)
2>/dev/null
```

id	name
43707766-75f3-4d48-80d9-1b7240fb161d	public_vn
2ab7ea04-8f5f-4b8d-acbf-a7c29c9b4112	VN1
1c59ded0-38e8-4168-b91f-4c51aba10d30	default-virtual-network
5b0a1040-91e4-47ff-bd4c-0a81e1901a1f	ip-fabric
7efddf64-ff3c-44d2-aeb2-45d7472b7a64	__link_local__

View the Floating IP Object in the API Server

Use the following to view the floating IP pool information in the API server. API server requests can be made on http port 8082.

The Contrail API servers have the virtual-network public_vn object that contains floating IP pool information. Use the following to view the floating-ip-pools object information.

curl http://<API-Server_IP>:8082/virtual-network/<UUID_of_VN>

Example

```
root@nodec6 ~]# curl
http://nodec6:8082/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d | python -m
json.tool

{
```

```

    "virtual-network": {

        "floating_ip_pools": [

            {

                "href":
"http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3",

                "to": [

                    "default-domain",

                    "admin",

                    "public_vn",

                    "public_pool"

                ],

                "uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3"

            }

        ],

        "fq_name": [

            "default-domain",

            "admin",

            "public_vn"

        ],

        "href":
"http://127.0.0.1:8095/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d",

        "id_perms": {

            "created": "2014-02-07T08:58:40.892803",

```



```

        "subnet": {
            "ip_prefix": "10.204.219.32",
            "ip_prefix_len": 29
        }
    },
]

},

"href":
"http://127.0.0.1:8095/network-ipam/39b0e8da-fcd4-4b35-856c-8d18570b1483",

"to": [

    "default-domain",

    "default-project",

    "default-network-ipam"

],

"uuid": "39b0e8da-fcd4-4b35-856c-8d18570b1483"

}

],

"parent_href":
"http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",

"parent_type": "project",

"parent_uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",

"route_target_list": {

    "route_target": [

```

```

        "target:64512:10003"

    ]

},

"routing_instances": [

    {

        "href":
"http://127.0.0.1:8095/routing-instance/3c6254ac-cfde-417e-916d-e7a1c0efad92",

        "to": [

            "default-domain",

            "admin",

            "public_vn",

            "public_vn"

        ],

        "uuid": "3c6254ac-cfde-417e-916d-e7a1c0efad92"

    }

],

"uuid": "43707766-75f3-4d48-80d9-1b7240fb161d",

"virtual_network_properties": {

    "extend_to_external_routers": null,

    "forwarding_mode": "l2_l3",

    "network_id": 4,

    "vxlan_network_identifier": null

```

```

    }

  }

}

```

View floating-ips in floating-ip-pools in the API Server

Once you have located the floating-ip-pools object, use the following to review its floating-ips object.

The floating-ips object should display the floating IP that is shown in the Contrail UI. The floating IP should have a reference to the virtual machine interface (VMI) object that is bound to the floating IP.

Example

```

[root@nodec6 ~]#
curlhttp://nodec6:8082/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3 | python
-m json.tool

{

  "floating-ip-pool": {

    "floating_ips": [

      {

        "href":
"http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",

        "to": [

          "default-domain",

          "admin",

          "public_vn",

          "public_pool",

```

```

        "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"

    ],

    "uuid": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"

}

],

"fq_name": [

    "default-domain",

    "admin",

    "public_vn",

    "public_pool"

],

"href":
"http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3",

"id_perms": {

    "created": "2014-02-07T08:58:41.136572",

    "description": null,

    "enable": true,

    "last_modified": "2014-02-07T08:58:41.136572",

    "permissions": {

        "group": "admin",

        "group_access": 7,

        "other_access": 7,

```



```

        "owner": "admin",

        "owner_access": 7

    },

    "uuid": {

        "uuid_lslong": 10683309858715198403,

        "uuid_mslong": 7365417021744038143

    }

},

"name": "public_pool",

"parent_href":
"http://127.0.0.1:8095/virtual-network/43707766-75f3-4d48-80d9-1b7240fb161d",

"parent_type": "virtual-network",

"parent_uuid": "43707766-75f3-4d48-80d9-1b7240fb161d",

"project_back_refs": [

    {

        "attr": {},

        "href":
"http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",

        "to": [

            "default-domain",

            "admin"

        ],

        "uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f"
    }
]

```

```

    }

    ],

    "uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3"

}

}

```

Check Floating IP Objects in the Virtual Machine Interface

Use the following to retrieve the virtual machine interface of the virtual machine from either the quantum port-list command or from the Contrail UI. Then get the virtual machine interface identifier and check its floating IP object associations.

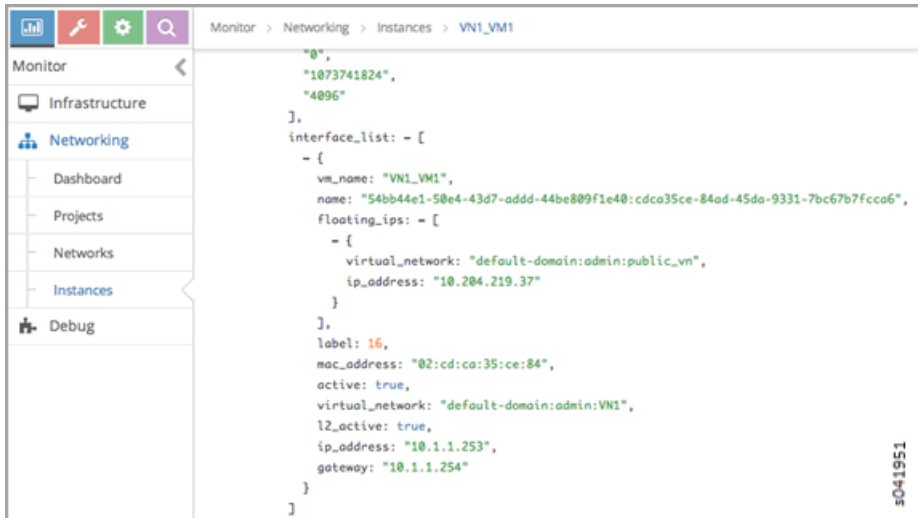
- Using **quantum port-list** to get the virtual machine interface:

Example

```
[root@nodec6 ~]# quantum port-list -F id -F fixed_ips
```

id	fixed_ips
cdca35ce-84ad-45da-9331-7bc67b7fcca6	{"subnet_id": "e80f480b-98d4-43cc-847c-711e637295db", "ip_address": "10.1.1.253"}

- Using Contrail UI to get the virtual machine interface:



Checking Floating IP Objects on the Virtual Machine Interface

Once you have obtained the virtual machine interface identifier, check the floating-ip objects that are associated with the virtual machine interface.

```
[root@nodec6 ~]# curl
http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0 |
python -m json.tool
```

```
{
  "floating-ip": {
    "floating_ip_address": "10.204.219.37",
    "fq_name": [
      "default-domain",
      "admin",
      "public_vn",
```

```

        "public_pool",

        "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0"

    ],

    "href":
"http://127.0.0.1:8095/floating-ip/f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",

    "id_perms": {

        "created": "2014-02-07T10:07:05.869899",

        "description": null,

        "enable": true,

        "last_modified": "2014-02-07T10:36:36.820926",

        "permissions": {

            "group": "admin",

            "group_access": 7,

            "other_access": 7,

            "owner": "admin",

            "owner_access": 7

        },

        "uuid": {

            "uuid_lslong": 12173378905373109408,

            "uuid_mslong": 17577202821367744163

        }
    }

```

```

    },

    "name": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",

    "parent_href":
"http://127.0.0.1:8095/floating-ip-pool/663737c1-f3ab-40ff-9442-bdb6c225e3c3",

    "parent_type": "floating-ip-pool",

    "parent_uuid": "663737c1-f3ab-40ff-9442-bdb6c225e3c3",

    "project_refs": [

        {

            "attr": null,

            "href":
"http://127.0.0.1:8095/project/deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f",

            "to": [

                "default-domain",

                "admin"

            ],

            "uuid": "deef6549-8e6c-4e3e-9cde-c9bc2b72ce6f"

        }

    ],

    "uuid": "f3eec4d6-889e-46a3-a8f0-879dfaff6ca0",

    "virtual_machine_interface_refs": [

        {

            "attr": null,

```

```

      "href":
"http://127.0.0.1:8095/virtual-machine-interface/cdca35ce-84ad-45da-9331-7bc67b7fcca6",

      "to": [

        "54bb44e1-50e4-43d7-addd-44be809f1e40",

        "cdca35ce-84ad-45da-9331-7bc67b7fcca6"

      ],

      "uuid": "cdca35ce-84ad-45da-9331-7bc67b7fcca6"

    }

  ]

}

}

```

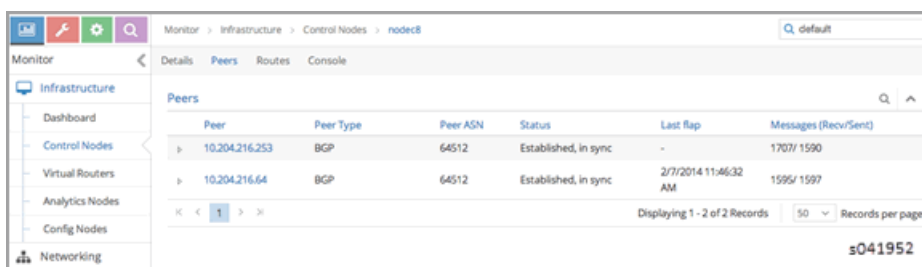
View the BGP Peer Status on the Control Node

Use the Contrail UI or the control node http introspect on port 8083 to view the BGP peer status. In the following example, the control nodes are **nodec7** and **nodec8**.

Ensure that the BGP peering state is displayed as **Established** for the control nodes and the gateway MX.

Example

- Using the Contrail UI:



Peer	Peer Type	Peer ASN	Status	Last flap	Messages (Recv/Sent)
10.204.216.253	BGP	64512	Established, in sync	-	1707/ 1590
10.204.216.64	BGP	64512	Established, in sync	2/7/2014 11:46:32 AM	1595/ 1597

Displaying 1 - 2 of 2 Records 50 Records per page

s041952

- Using the control-node Introspect:
http://nodec7:8083/Snh_BgpNeighborReq?ip_address=&domain=
http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=

Querying Routes in the Public Virtual Network

On each control-node, a query on the routes in the **public_vn** lists the routes that are pushed by the MX gateway, which in the following example are 0.0.0.0/0 and 10.204.218.0/24.

In the following results, the floating IP route of 10.204.217.32 is installed by the compute node (nodec10) that hosts that virtual machine.

Example

- Using the Contrail UI:

Routing Table	Prefix	Protocol	Source	Next hop	Label	Secur...	Origin VN
default-domainadminpublic_vnpublic_vn.inet.0	0.0.0.0/0	BGP	10.204.216.253	10.204.216.253	16	-	default-domainadminpublic_vn
	10.204.218.0/24	BGP	10.204.216.253	10.204.216.253	16	-	default-domainadminpublic_vn
	10.204.219.32/32	XMPP	nodec10	10.204.216.67	16	1	default-domainadminpublic_vn

- Using the http Introspect:
 Following is the format for using an introspect query.
http://<nodename/ip>:8083/Snh_ShowRouteReq?x=<RoutingInstance of public VN>.inet.0
 Example

http://nodec8:8083/Snh_BgpNeighborReq?ip_address=&domain=

nodec7-8083 / Snh_ShowRouteReq?x~default-domain:admin-public_vn-public_vn_inet.0

Contrail

Collapse Expand Filter Refresh

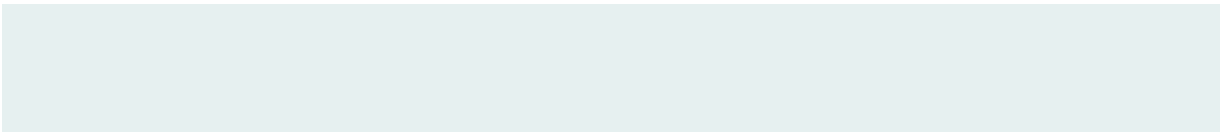
ShowRouteReq

Tables

routing_instance	routing_table_name	prefixes	paths	primary_paths	secondary_paths	invalid_paths	routes																
default-domain:admin-public_vn-public_vn_inet.0	default-domain:admin-public_vn-public_vn_inet.0	8.8.8.8/8	4	1	3	0	<table><thead><tr><th>prefix</th><th>last_modified</th><th>paths</th><th>protocol</th></tr></thead><tbody><tr><td>8.8.8.8/8</td><td>2014-Feb-07 08:10:41.204000</td><td>18.204.216.8/24</td><td>BGP</td></tr><tr><td>18.204.216.8/24</td><td>2014-Feb-07 08:10:41.204000</td><td>18.204.216.37/32</td><td>BGP</td></tr><tr><td>18.204.216.37/32</td><td>2014-Feb-07 18:10:16.800000</td><td></td><td>BGP</td></tr></tbody></table>	prefix	last_modified	paths	protocol	8.8.8.8/8	2014-Feb-07 08:10:41.204000	18.204.216.8/24	BGP	18.204.216.8/24	2014-Feb-07 08:10:41.204000	18.204.216.37/32	BGP	18.204.216.37/32	2014-Feb-07 18:10:16.800000		BGP
prefix	last_modified	paths	protocol																				
8.8.8.8/8	2014-Feb-07 08:10:41.204000	18.204.216.8/24	BGP																				
18.204.216.8/24	2014-Feb-07 08:10:41.204000	18.204.216.37/32	BGP																				
18.204.216.37/32	2014-Feb-07 18:10:16.800000		BGP																				

View Corresponding BGP L3VPN Routes

Use the Contrail UI or the http introspect to view the public route's corresponding BGP L3VPN routes, as in the following.



Example

- Using the Contrail UI:

Routing Table	Prefix	Protocol	Source	Next hop	Label	Security	Origin VNI
bgp.l3vpn.0	10.204.216.253/5:0.0.0.0/0	BGP	10.204.216.253	10.204.216.253	16	-	-
	10.204.216.253/5:10.204.216.0/24	BGP	10.204.216.253	10.204.216.253	16	-	-
	10.204.216.67/1:10.1.1.253/32	XMPP	nodec10	10.204.216.67	16	1	default-domain:admin-vn.1
		BGP	10.204.216.64	10.204.216.67	16	1	default-domain:admin-vn.1
	10.204.216.67/2:10.204.216.37/32	XMPP	nodec10	10.204.216.67	16	1	default-domain:admin-gpu-blic_vn
		BGP	10.204.216.64	10.204.216.67	16	1	default-domain:admin-gpu-blic_vn

- Using the control-node Introspect:

http://nodec7:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0

http://nodec8:8083/Snh_ShowRouteReq?x=bgp.l3vpn.0

Verification from the MX80 Gateway

This section provides options for verifying floating IP pools from the MX80 gateway.

Verify BGP Sessions are Established

Use the following commands from the gateway to verify that BGP sessions are established with the control nodes nodec7 and nodec8:

```

root@mx-host> show bgp neighbor 10.204.216.64

Peer: 10.204.216.64+59287 AS 64512 Local: 10.204.216.253+179 AS 64512

Type: Internal      State: Established      Flags: <Sync>

Last State: OpenConfirm  Last Event: RecvKeepAlive

Last Error: Hold Timer Expired Error

Options: <Preference LocalAddress KeepAll AddressFamily Rib-group Refresh>

Address families configured: inet-vpn-unicast

Local Address: 10.204.216.253 Holdtime: 90 Preference: 170

Number of flaps: 216

Last flap event: HoldTime

Error: 'Hold Timer Expired Error' Sent: 68 Recv: 0

Error: 'Cease' Sent: 0 Recv: 43

Peer ID: 10.204.216.64   Local ID: 10.204.216.253   Active Holdtime: 90

Keepalive Interval: 30      Group index: 0      Peer index: 3

BFD: disabled, down

NLRI for restart configured on peer: inet-vpn-unicast

NLRI advertised by peer: inet-vpn-unicast

```

```

NLRI for this session: inet-vpn-unicast

Peer does not support Refresh capability

Stale routes from peer are kept for: 300

Peer does not support Restarter functionality

Peer does not support Receiver functionality

Peer does not support 4 byte AS extension

Peer does not support Addpath

```

Show Routes Learned from Control Nodes

From the MX80, use `show route` to display the routes for the virtual machine 10.204.219.37 that are learned from both control-nodes.

In the following example, the routes learned are 10.204.216.64 and 10.204.216.65, pointing to a dynamic GRE tunnel next hop with a label of 16 (of the virtual machine).

```

public.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)

+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 10w6d 18:47:50
                   > to 10.204.218.254 via ge-1/0/1.0

10.204.218.0/24    *[Direct/0] 10w6d 18:47:51
                   > via ge-1/0/1.0

10.204.218.1/32    *[Local/0] 10w6d 18:48:07

```

```
Local via ge-1/0/1.0

10.204.219.37/32  *[BGP/170] 09:42:43, localpref 100, from 10.204.216.64

AS path: ?, validation-state: unverified

> via gr-1/0/0.32779, Push 16

[BGP/170] 09:42:43, localpref 100, from 10.204.216.65

AS path: ?, validation-state: unverified

> via gr-1/0/0.32779, Push 16
```

Viewing the Compute Node VnsW Agent

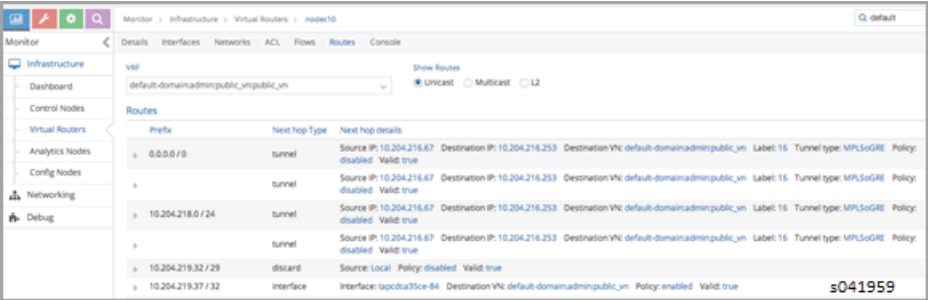
The compute node introspect can be accessed from port 8085. In the following examples, the compute nodes are nodec9 and nodec10.

View Routing Instance Next Hops

On the routing instance of VN1, the routes 0.0.0.0/0 and 10.204.218.0/24 should have the next hop pointing to the MX gateway (10.204.216.253).

Example

Using the Contrail UI:



Using the Unicast Route Table Index to View Next Hops

Alternatively, from the agent introspect, you can view the next hops at the unicast route table.

First, use the following to get the unicast route table index (ucindex) for the routing instance **default-domain:admin:public_vn:public_vn**.

http://nodec10:8085/Snh_VrfListReq?x=default-domain:admin:public_vn:public_vn

Example

In the following example, the unicast route table index is 2.

name	ucindex	mcindex	l2index
default-domain:admin:public_vn:public_vn	2	2	2

more: false

Next, perform a route request query on ucindex 2, as shown in the following. The tunnel detail indicates the source and destination endpoints of the tunnel and the MPLS label 16 (the label of the virtual machine).

The query should also show a route for 10.204.219.37 with an interface next hop of tap-interface.

http://nodec10:8085/Snh_Inet4UcRouteReq?x=2

src_ip	src_port	src_vrf	path_list												
9.9.9.9	0	default-domain:admin:public_vn:public_vn	<table border="1"> <thead> <tr> <th>ip</th> <th>label</th> <th>vlan_id</th> <th>peer</th> </tr> </thead> <tbody> <tr> <td>10.204.216.67</td> <td>16</td> <td>0</td> <td>10.204.216.64</td> </tr> <tr> <td>10.204.216.253</td> <td>16</td> <td>0</td> <td>10.204.216.65</td> </tr> </tbody> </table>	ip	label	vlan_id	peer	10.204.216.67	16	0	10.204.216.64	10.204.216.253	16	0	10.204.216.65
ip	label	vlan_id	peer												
10.204.216.67	16	0	10.204.216.64												
10.204.216.253	16	0	10.204.216.65												

10.204.219.37	32	default-domain:admin:public_vn:public_vn	path_list			
			nh	label	vlan_id	peer
			nh	16	0	10.204.216.64
			type	interface		
			ref_count	g		
			valid	true		
			policy	enabled		
			itf	tapcdca35ce-84		
			mac	2:cd:ca:35:ce:84		
			mcast	disabled		
			nh	16	0	10.204.216.65
			type	interface		
			ref_count	g		
			valid	true		
			policy	enabled		
			itf	tapcdca35ce-84		
			mac	2:cd:ca:35:ce:84		
			mcast	disabled		

A ping from the MX gateway to the virtual machine's floating IP in the public routing-instance should work.

Advanced Troubleshooting

If you still have reachability problems after performing all of the tests in this article, for example, a ping between the virtual machine and the MX IP or to public addresses is failing, try the following:

- Validate that all the required Contrail processes are running by using the **contrail-status** command on all of the nodes.
- On the compute node where the virtual machine is present (nodec10 in this example), perform a tcpdump on the tap interface (**tcpdump -ni tapcdca35ce-84**). The output should show the incoming packets from the virtual machine.
- Check to see if any packet drops occur in the kernel vrouter module:

http://nodec10:8085/Snh_KDropsStatsReq?

In the output, scroll down to find any drops. Note: You can ignore any ds_invalid_arp increments.

- On the physical interface where packets transmit onto the compute-node, perform a tcpdump matching the host IP of the MX to show the GRE encapsulated packets, as in the following.

```
[root@nodec10 ~]# cat /etc/contrail/agent.conf |grep -A 1 eth-port
```

```
<eth-port>
```

```
<name>p1p0p0</name>
```

```

</eth-port>

<metadata-proxy>

[root@nodec10 ~]# tcpdump -ni plp0p0 host 10.204.216.253 -vv

tcpdump: WARNING: plp0p0: no IPv4 address assigned

tcpdump: listening on plp0p0, link-type EN10MB (Ethernet), capture size 65535 bytes

02:06:51.729941 IP (tos 0x0, ttl 64, id 57430, offset 0, flags [DF], proto GRE
(47), length 112)

    10.204.216.253 > 10.204.216.67: GREv0, Flags [none], length 92

        MPLS (label 16, exp 0, [S], ttl 54)

        IP (tos 0x0, ttl 54, id 35986, offset 0, flags [none], proto ICMP (1), length
84)

        172.29.227.6 > 10.204.219.37: ICMP echo request, id 53240, seq 242, length 64

02:06:51.730052 IP (tos 0x0, ttl 64, id 324, offset 0, flags [none], proto GRE
(47), length 112)

    10.204.216.67 > 10.204.216.253: GREv0, Flags [none], length 92

        MPLS (label 16, exp 0, [S], ttl 64)

        IP (tos 0x0, ttl 64, id 33909, offset 0, flags [none], proto ICMP (1), length
84)

        10.204.219.37 > 172.29.227.6: ICMP echo reply, id 53240, seq 242, length 64

02:06:52.732283 IP (tos 0x0, ttl 64, id 12675, offset 0, flags [DF], proto GRE
(47), length 112)

    10.204.216.253 > 10.204.216.67: GREv0, Flags [none], length 92

        MPLS (label 16, exp 0, [S], ttl 54)

        IP (tos 0x0, ttl 54, id 54155, offset 0, flags [none], proto ICMP (1), length

```



```

84)

172.29.227.6 > 10.204.219.37: ICMP echo request, id 53240, seq 243, length 64

02:06:52.732355 IP (tos 0x0, ttl 64, id 325, offset 0, flags [none], proto GRE
(47), length 112)

10.204.216.67 > 10.204.216.253: GREv0, Flags [none], length 92

MPLS (label 16, exp 0, [S], ttl 64)

IP (tos 0x0, ttl 64, id 33910, offset 0, flags [none], proto ICMP (1), length
84)

10.204.219.37 > 172.29.227.6: ICMP echo reply, id 53240, seq 243, length 64

^C

4 packets captured

5 packets received by filter

0 packets dropped by kernel

[root@nodec10 ~]#

```

- On the MX gateway, use the following to inspect the GRE tunnel rx/tx (received/transmitted) packet count:

```

root@mx-host> show interfaces gr-1/0/0.32779 |grep packets

Input packets : 542

Output packets: 559

root@blr-mx1> show interfaces gr-1/0/0.32779 |grep packets

Input packets : 544

Output packets: 561

```

- Look for any packet drops in the FPC, as in the following:

```
show pfe statistics traffic fpc <id>
```

- Also inspect the dynamic tunnels, using the following:

```
show dynamic-tunnels database
```

Removing Stale Virtual Machines and Virtual Machine Interfaces

IN THIS SECTION

- [Problem Example | 273](#)
- [Show Virtual Machines | 274](#)
- [Show Virtual Machines Using Python API | 276](#)
- [Delete Methods | 277](#)

This topic gives examples for removing stale VMs (virtual machines) and VMIs (virtual machine interfaces). Before you can remove a stale VM or VMI, you must first remove any back references associated to the VM or VMI.

Problem Example

The troubleshooting examples in this topic are based on the following problem example. A **net-delete** of the virtual machine 2a8120ec-bd18-49f4-aca0-acfc6e8fe74f returned the following messages that there are two VMIs that still have back-references to the stale VM.

The two VMIs must be deleted first, then the Neutron **net-delete <vm_ID>** command will complete without errors.

```
From neutron.log:

2014-03-10 14:18:05.208

DEBUG [urllib3.connectionpool]

"DELETE/virtual-network/2a8120ec-bd18-49f4-aca0-acfc6e8fe74f HTTP/1.1" 409 203
```

```

2014-03-10 14:18:05.278

ERROR [neutron.api.v2.resource] delete failed

Traceback (most recent call last):

  File "/usr/lib/python2.7/dist-packages/neutron/api/v2/resource.py", line
84, in resource

    result = method(request=request, **args)

  File "/usr/lib/python2.7/dist-packages/neutron/api/v2/base.py", line
432, in delete

    obj_deleter(request.context, id, **kwargs)

  File

"/usr/lib/python2.7/dist-packages/neutron/plugins/juniper/contrail/contrail
plugin.py", line 294, in delete_network

    raise e

RefsExistError: Back-References from

http: //127.0.0.1:8082/virtual-machine-interface/51daf6f4-7366-4463-a819-bd1
17fe3a8c8,

http: //127.0.0.1:8082/virtual-machine-interface/30882e66-e175-4fbb-862e-354
bb700b579 still exist

```

Show Virtual Machines

Use the following command to show all of the virtual machines known to the Contrail API server. Replace the variable **<config-node-IP>** shown in the example with the IP address of the **config-node** in your setup.

http://<config-node-IP>:8082/virtual-machines

Example

In the following example, 03443891-99cc-4784-89bb-9d1e045f8aa6 is a stale VM that needs to be removed.

```
virtual-machines:

[

{

  href:"http:
//example-node:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6",

  fq_name:

[

    "03443891-99cc-4784-89bb-9d1e045f8aa6"

  ],

  uuid:"03443891-99cc-4784-89bb-9d1e045f8aa6"

},
```

When the user attempts to delete the stale VM, a message displays that children to the VM still exist:

```
root@example-node:~# curl -X DELETE -H "Content-Type: application/json; charset=UTF-8"
http: //127.0.0.1:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6
Children http:
//127.0.0.1:8082/virtual-machine-interface/0c32a82a-7bd3-46c7-b262-6d85b9911a0d still
exist
root@example-node:~#
```

The user opens <http://example-node:8082/virtual-machine/03443891-99cc-4784-89bb-9d1e045f8aa6>, and sees a **virtual-machine-interface** (VMI) attached to it. The VMI must be removed before the VM can be removed.

However, when the user attempts to delete the VMI from the stale VM, they get a message that there is still a back-reference:

```

root@example-node:~# curl -X DELETE -H "Content-Type: application/json; charset=UTF-8"
http:
//<example-IP>:8082/virtual-machine-interface/0c32a82a-7bd3-46c7-b262-6d85b9911a0d

Back-References from http:
//<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist

root@example-node:~#

```

Because there is a back-reference from an **instance-ip** object still present, the **instance-ip** object must first be deleted, as follows:

```

root@example-node:~# curl -X DELETE -H "Content-Type: application/json; charset=UTF-8"
http: //<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4

root@example-node:~#

```

When the **instance-ip** is deleted, then the VMI and the VM can be deleted.

NOTE: To prevent inconsistency, be certain that the VM is not present in the Nova database before deleting the VM.

Show Virtual Machines Using Python API

The following example shows how to view virtual machines using a Python API. This example shows virtual machines and back-references. Once you identify back-references and existing children, you can delete them first, then delete the stale VM.

```

root@example-node:~# source /opt/contrail/api-venv/bin/activate

File "<stdin>", line 1, in <module>

File
"/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/gen/vnc_api_client_gen.py",
line 3793, in virtual_machine_interface_delete

    content = self._request_server(rest.OP_DELETE, uri)

File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/vnc_api.py", line
342, in _request_server

```

```

        raise RefsExistError(content)

cfgm_common.exceptions.RefsExistError: Back-References from http: //
<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist

>>> (api-venv)root@example-node:~# python

Python 2.7.5 (default, Mar 10 2014, 03:55:35)

[GCC 4.6.3] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> from vnc_api.vnc_api import VncApi

>>> vh=VncApi()

>>> vh.virtual_machine_interface_delete(id='0c32a82a-7bd3-46c7-b262-6d85b9911a0d')

```

Traceback (most recent call last):

```

File "<stdin>", line 1, in <module>

    File
"/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/gen/vnc_api_client_gen.py",
line 3793, in virtual_machine_interface_delete

    content = self._request_server(rest.OP_DELETE, uri)

File "/opt/contrail/api-venv/lib/python2.7/site-packages/vnc_api/vnc_api.py", line
342, in _request_server

    raise RefsExistError(content)

cfgm_common.exceptions.RefsExistError: Back-References from http: //
<example-IP>:8082/instance-ip/6ffa29a1-023f-462b-b205-353da8e3a2a4 still exist

>>>

```

Delete Methods

Use help (vh) to show all delete methods supported.

Typical commands for deleting VMs and VMIs include:

- `virtual_machine_delete()` to delete a virtual machine
- `instance_ip_delete()` to delete an `instance-ip`.

Troubleshooting Link-Local Services in Contrail

IN THIS SECTION

- [Overview of Link-Local Services | 278](#)
- [Troubleshooting Procedure for Link-Local Services | 279](#)
- [Metadata Service | 281](#)
- [Troubleshooting Procedure for Link-Local Metadata Service | 281](#)

Use the troubleshooting steps and guidelines in this topic when you have errors with Contrail link-local services.

Overview of Link-Local Services

Virtual machines might be set up to access specific services hosted on the fabric infrastructure. For example, a virtual machine might be a Nova client that requires access to the Nova API service running in the fabric network. Access to services hosted on the fabric network can be provided by configuring the services as link-local services.

A link-local address and a service port is chosen for the specific service running on a TCP / UDP port on a server in the fabric. With the link-local service configured, virtual machines can access the service using the link-local address. For link-local services, Contrail uses the address range 169.254.169.x.

Link-local service can be configured using the Contrail WebUI: **Configure > Infrastructure > Link Local Services**.

Create Link Local Service

Service Name	Link Local Service Address		Fabric Address		
	IP	Port	IP / DNS		Port
ntp	169.254.169.100	123	IP 172.17.28.5	+	123

Cancel

Save

Troubleshooting Procedure for Link-Local Services

Use the following steps when you are troubleshooting link-local services errors.

1. Verify the reachability of the fabric server that is hosting the link-local service from the compute node.
2. Check the state of the virtual machine and the interface:
 - Is the **Status** of virtual machine **Up**?
 - Is the corresponding tap interface **Active**?

Checking the virtual machine status in the Contrail UI:

Monitor > Infrastructure > Virtual Routers > nodec15						
Details Interfaces Networks ACL Flows Routes Console						
Interfaces						
Name	Label	Status	Network	IP Address	Floating IP	Instance
tap4b094dbe-f0	18	Up	vn1 (demo)	1.2.3.247	None	4f4b917a-a071-4517-961a-0e41067fec63 / vn1-v m2

Checking the tap interface status in the http agent introspect:

http://<compute-node-ip>:8085/Snh_ItfReq?name=

itf_list				
index	name	uuid	vrf_name	active
3	tap722a7a11-6d	722a7a11-6d2e-47e9-a4cc-687a105a240f	default-domain:demo:vn1:vn1	Active

3. Check the link-local configuration in the vrouter agent. Make sure the configured link-local service is displayed.

http://<compute-node-ip>:8085/Snh_LinkLocalServiceInfo?

service_list					
linklocal_service_name	linklocal_service_ip	linklocal_service_port	ipfabric_dns_name	ipfabric_ip	ipfabric_port
ntp	169.254.169.100	123	-	ipfabric_ip 172.17.28.5	123

4. Validate the BGP neighbor config and the BGP peering config object. When the virtual machine communicates with the configured link-local service, a forward and reverse flow for the communication is set up. Check that the flow for this communication is created and the flow action is NAT.

http://<compute-node-ip>:8085/Snh_KFlowReq?flow_idx=

Check that all flow entries display NAT action programmed and display flags for the fields (source or destination IP and ports) that have NAT programmed. Also shown are the number of packets and bytes transmitted in the respective flows.

flow_list									
index	rflow	slp	sport	dip	dport	proto	vrf_id	action	flags
467472	234436	1.2.3.247	123	169.254.169.100	123	17	1	NAT	ACTIVE VRFT SNAT SPAT DNAT SPAT
234436	467472	172.17.28.5	123	10.204.216.72	43226	17	0	NAT	ACTIVE VRFT SNAT DNAT s041997

The forward flow displays the source IP of the virtual machine and the destination IP of the link-local service. The reverse flow displays the source IP of the fabric host and the destination IP of the compute node's vhost interface. If the service is hosted on the same compute node, the destination address of the reverse flow displays the metadata address allocated to the virtual machine.

Note that the **index** and **rflow** index for the two flows are reversed.

You can also view similar information in the vrouter agent introspect page, where you can see the policy and security group for the flow. Check that the flow actions display as **pass**.

http://<compute-node-ip>:8085/Snh_FetchAllFlowRecords?

Metadata Service

OpenStack allows virtual instances to access metadata by sending an HTTP request to the link-local address 169.254.169.254. The metadata request from the instance is proxied to Nova, with additional HTTP header fields added, which Nova uses to identify the source instance. Then Nova responds with appropriate metadata.

The Contrail vrouter acts as the proxy, trapping the metadata requests, adding the necessary header fields, and sending the requests to the Nova API server.

Troubleshooting Procedure for Link-Local Metadata Service

Metadata service is also a link-local service, with a fixed service name (metadata), a fixed service address (169.254.169.254:80), and a fabric address pointing to the server where the OpenStack Nova API server is running. All of the configuration and troubleshooting procedures for Contrail link-local services also apply to the metadata service.

However, for metadata service, the flow is always set up to the compute node, so the vrouter agent will update and proxy the HTTP request. The vrouter agent listens on a local port to receive the metadata requests. Consequently, the reverse flow has the compute node as the source IP, the local port on which the agent is listening is the source port, and the instance’s metadata IP is the destination IP address.

After performing all of the troubleshooting procedures for link-local services, the following additional steps can be used to further troubleshoot metadata service.

- 1. Check the metadata statistics for: the number of metadata requests received by the vrouter agent, the number of proxy sessions set up with the Nova API server, and number of internal errors encountered.

http://<compute-node-ip>:8085/Snh_MetadataInfo?

The port on which the vrouter agent listens for metadata requests is also displayed.

metadata_server_port	45094
metadata_requests	2
metadata_responses	0
metadata_proxy_sessions	2
metadata_internal_errors	0

s041998

2. Check the metadata trace messages, which show the trail of metadata requests and responses.

`http://<compute-node-ip>:8085/Snh_SandeshTraceRequest?x=Metadata`

3. Check the Nova configuration. On the server running the OpenStack service, inspect the **nova.conf** file.

- Ensure that the metadata proxy is enabled, as follows:

`service_neutron_metadata_proxy = True`

`service_quantum_metadata_proxy = True` (on older installations)

- Check to see if the metadata proxy shared secret is set:

`neutron_metadata_proxy_shared_secret`

`quantum_metadata_proxy_shared_secret` (on older installations)

If the shared secret is set in **nova.conf**, the same secret must be configured on each compute node in the file **/etc/contrail/contrail-vrouter-agent.conf**, and the same shared secret must be updated in the **METADATA** section as **`metadata_proxy_secret=<secret>`**.

4. Restart the vrouter agent after modifying the shared secret:

`service contrail-vrouter restart`

4

PART

Conrail Commands and APIs

Conrail Commands | **284**

Conrail Application Programming Interfaces (APIs) | **314**

Contrail Commands

IN THIS CHAPTER

- [Getting Contrail Node Status | 284](#)
- [contrail-logs \(Accessing Log File Messages\) | 297](#)
- [contrail-status \(Viewing Node Status\) | 301](#)
- [contrail-version \(Viewing Version Information\) | 303](#)
- [Backing Up Contrail Databases Using JSON Format | 305](#)

Getting Contrail Node Status

IN THIS SECTION

- [Overview | 284](#)
- [UVE for NodeStatus | 284](#)
- [Node Status Features | 285](#)
- [Using Introspect to Get Process Status | 293](#)
- [contrail-status script | 294](#)

Overview

This topic describes how to view the status of a Contrail node on a physical server. Contrail nodes include config, control, analytics, compute, and so on.

UVE for NodeStatus

The User-Visible Entity (UVE) mechanism is used to aggregate and send the status information. All node types send a NodeStatus structure in their respective node UVEs. The following is a control node UVE of NodeStatus:

```

struct NodeStatus {

    1: string name (key="ObjectBgpRouter")

    2: optional bool deleted

    3: optional string status

    // Sent by process

    4: optional list<process_info.ProcessStatus> process_status (aggtype="union")

    // Sent by node manager

    5: optional list<process_info.ProcessInfo> process_info (aggtype="union")

    6: optional string description

}

uve sandesh NodeStatusUVE {

    1: NodeStatus data

}

```

Node Status Features

The most important features of NodeStatus include:

ProcessStatus

ProcessInfo

ProcessStatus

Also `process_status`, is sent by the processes corresponding to the virtual node, and displays the status of the process and an aggregate state indicating if the process is functional or non-functional. The `process_status` includes the state of the process connections (`ConnectionInfo`) to important services and other information necessary for the process to be functional. Each process sends its `NodeStatus` information, which is aggregated as union (`aggtype="union"`) at the analytics node. The following is the `ProcessStatus` structure:

```

1.  struct ProcessStatus {
2.      1: string module_id
3.      2: string instance_id
4.      3: string state
5.      4: optional list<ConnectionInfo> connection_infos
6.      5: optional string description
7.  }
8.
9.  struct ConnectionInfo {
10.     1: string type
11.     2: string name
12.     3: optional list<string> server_addrs
13.     4: string status
14.     5: optional string description
15.  }

```

ProcessInfo

Sent by the node manager, /usr/bin/contrail-nodemgr. Node manager is a monitor process per contrail virtual node that tracks the running state of the processes. The following is the ProcessInfo structure:

```

16. struct ProcessInfo {
17.     1: string                process_name
18.     2: string                process_state
19.     3: u32                   start_count
20.     4: u32                   stop_count
21.     5: u32                   exit_count
22.     // time when the process last entered running stage
23.     6: optional string       last_start_time
24.     7: optional string       last_stop_time
25.     8: optional string       last_exit_time
26.     9: optional list<string> core_file_list
27. }

```

Example: NodeStatus

The following is an example output of NodeStatus obtained from the Rest API:

```

http://:8081/analytics/uves/control-...ilt=NodeStatus .

{
  NodeStatus:
  {
    process_info:
    [

```



```
{  
  
  process_name: "contrail-control",  
  
  process_state: "PROCESS_STATE_RUNNING",  
  
  last_stop_time: null,  
  
  start_count: 1,  
  
  core_file_list: [ ],  
  
  last_start_time: "1409002143776558",  
  
  stop_count: 0,  
  
  last_exit_time: null,  
  
  exit_count: 0  
  
},  
  
{  
  
  process_name: "contrail-control-nodemgr",  
  
  process_state: "PROCESS_STATE_RUNNING",  
  
  last_stop_time: null,  
  
  start_count: 1,  
  
  core_file_list: [ ],  
  
  last_start_time: "1409002141773481",  
  
  stop_count: 0,  
  
  last_exit_time: null,
```

```

        exit_count: 0

    },

    {

        process_name: "contrail-dns",

        process_state: "PROCESS_STATE_RUNNING",

        last_stop_time: null,

        start_count: 1,

        core_file_list: [ ],

        last_start_time: "1409002145778383",

        stop_count: 0,

        last_exit_time: null,

        exit_count: 0

    },

    {

        process_name: "contrail-named",

        process_state: "PROCESS_STATE_RUNNING",

        last_stop_time: null,

        start_count: 1,

        core_file_list: [ ],

        last_start_time: "1409002147780118",

```

```

        stop_count: 0,

        last_exit_time: null,

        exit_count: 0

    }

],

process_status:
[

{

    instance_id: "0",

    module_id: "ControlNode",

    state: "Functional",

    description: null,

    connection_infos:
    [

        {

            server_addrs:
            [

                "10.84.13.45:8443"

            ],

            {

                server_addrs:

```

```

[
  "10.84.13.45:8086"
],
status: "Up",
type: "Collector",
name: null,
description: "Established"
},

{
  server_addrs:
[
  "10.84.13.45:5998"
],
status: "Up",
type: "Discovery",
name: "Collector",
description: "SubscribeResponse"
},

{
  server_addrs:
[

```

```

        "10.84.13.45:5998"

    ],

    status: "Up",

    type: "Discovery",

    name: "IfmapServer",

    description: "SubscribeResponse"

},

{

    server_addrs:

    [

        "10.84.13.45:5998"

    ],

    status: "Up",

    type: "Discovery",

    name: "xmpp-server",

    description: "Publish Response - HeartBeat"

}

]

}

]

}

```

```
}
```

Using Introspect to Get Process Status

The user can also view the state of a specific process by using the introspect mechanism.

Example: Introspect of NodeStatus

The following is an example of the process state of contrail-control that is obtained by using

`http://server-ip:8083/Snh_SandeshUVECacheReq?x=NodeStatus`

NOTE: The example output is the ProcessStatus of only one process of contrail-control. It does not show the full aggregated status of the control node through its UVE (as in the previous example).

```
root@a6s45:~# curl http://10.84.13.45:8083/Snh_SandeshU...q?x=NodeStatus

<?xml-stylesheet type="text/xsl"
href="/universal_parse.xsl"?><__NodeStatusUVE_list type="slist"><NodeStatusUVE
  type="sandesh"><data type="struct" identifier="1"><NodeStatus><name
type="string" identifier="1" key="ObjectBgpRouter">a6s45</name><process_status
  type="list" identifier="4" aggttype="union"><list type="struct"
size="1"><ProcessStatus><module_id type="string"
identifier="1">ControlNode</module_id><instance_id type="string"
identifier="2">0</instance_id><state type="string"
identifier="3">Functional</state><connection_infos type="list"
identifier="4"><list type="struct" size="5"><ConnectionInfo><type type="string"
  identifier="1">IFMap</type><name type="string"
identifier="2">IFMapServer</name><server_addrs type="list" identifier="3"><list
  type="string"
size="1"><element>10.84.13.45:8443</element></list></server_addrs><status
type="string" identifier="4">Up</status><description type="string"
identifier="5">Connection with IFMap Server
(ironD)</description></ConnectionInfo><ConnectionInfo><type type="string"
identifier="1">Collector</type><name type="string"
```

```

identifier="2"></name><server_addrs type="list" identifier="3"><list
type="string"
size="1"><element>10.84.13.45:8086</element></list></server_addrs><status
type="string" identifier="4">Up</status><description type="string"
identifier="5">Established</description></ConnectionInfo><ConnectionInfo><type
type="string" identifier="1">Discovery</type><name type="string"
identifier="2">Collector</name><server_addrs type="list" identifier="3"><list
type="string"
size="1"><element>10.84.13.45:5998</element></list></server_addrs><status
type="string" identifier="4">Up</status><description type="string"
identifier="5">SubscribeResponse</description></ConnectionInfo><ConnectionInfo><type
type="string" identifier="1">Discovery</type><name type="string"
identifier="2">IfmapServer</name><server_addrs type="list" identifier="3"><list
type="string"
size="1"><element>10.84.13.45:5998</element></list></server_addrs><status
type="string" identifier="4">Up</status><description type="string"
identifier="5">SubscribeResponse</description></ConnectionInfo><ConnectionInfo><type
type="string" identifier="1">Discovery</type><name type="string"
identifier="2">xmpp-server</name><server_addrs type="list" identifier="3"><list
type="string"
size="1"><element>10.84.13.45:5998</element></list></server_addrs><status
type="string" identifier="4">Up</status><description type="string"
identifier="5">Publish Response -
HeartBeat</description></ConnectionInfo></list></connection_infos><description
type="string"
identifier="5"></description></ProcessStatus></list></process_status></NodeStatus></data></NodeStatusUVE><SandeshUVECacheResp
type="sandesh"><returned type="u32" identifier="1">1</returned><more
type="bool"
identifier="0">false</more></SandeshUVECacheResp></__NodeStatusUVE_list>

```

contrail-status script

The contrail-status script is used to give the status of the Contrail processes on a server.

The contrail-status script first checks if a process is running, and if it is, performs introspect into the process to get its functionality status, then outputs the aggregate status.

The possible states to display include:

- active - the process is running and functional; the internal state is good
- inactive - not started or stopped by user

- failed – the process exited too quickly and has not restarted
- initializing – the process is running, but the internal state is not yet functional.

Example Output: Contrail-Status Script

The following is an example output from the contrail-status script.

```
root@a6s45:~# contrail-status

== Contrail vRouter ==

supervisor-vrouter:          active

contrail-vrouter-agent       active

contrail-vrouter-nodemgr     active


== Contrail Control ==

supervisor-control:         active

contrail-control            active

contrail-control-nodemgr    active

contrail-dns                active

contrail-named              active


== Contrail Analytics ==

supervisor-analytics:       active

contrail-analytics-api      active

contrail-analytics-nodemgr  active

contrail-collector          active
```



```
contrail-query-engine      active
```

```
== Contrail Config ==
```

```
supervisor-config:        active
```

```
contrail-api:0             active
```

```
contrail-config-nodemgr    active
```

```
contrail-schema            active
```

```
contrail-svc-monitor       active
```

```
rabbitmq-server            active
```

```
== Contrail Web UI ==
```

```
supervisor-webui:         active
```

```
contrail-webui             active
```

```
contrail-webui-middleware  active
```

```
redis-webui                active
```

```
== Contrail Database ==
```

```
supervisord-contrail-database:active
```

```
contrail-database          active
```

```
contrail-database-nodemgr  active
```

contrail-logs (Accessing Log File Messages)

IN THIS SECTION

- [Command-Line Options for Contrail-Logs | 297](#)
- [Option Descriptions | 298](#)
- [Example Uses | 299](#)

A command-line utility, **contrail-logs**, uses REST APIs to retrieve system log messages, object log messages, and trace messages.

Command-Line Options for Contrail-Logs

The command-line utility for accessing log file information is **contrail-logs** in the analytics node. The following are the options supported at the command line for **contrail-logs**, as viewed using the **--help** option.

```
[root@host]# contrail-logs --help
usage: contrail-logs [-h]
                    [--opserver-ip OPSERVER_IP]
                    [--opserver-port OPSERVER_PORT]
                    [--start-time START_TIME]
                    [--end-time END_TIME]
                    [--last LAST]
                    [--source SOURCE]
                    [--module {ControlNode, VRouterAgent, ApiServer, Schema,
OpServer, Collector, QueryEngine, ServiceMonitor, DnsAgent}]
                    [--category CATEGORY]
                    [--level LEVEL]
                    [--message-type MESSAGE_TYPE]
                    [--reverse]
                    [--verbose]
                    [--all]
                    [--object {ObjectVNTTable, ObjectVMTable, ObjectSITable,
ObjectVRouter, ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter,
ObjectXmppConnection, ObjectCollectorInfo, ObjectGeneratorInfo, ObjectConfigNode}]
                    [--object-id OBJECT_ID]
                    [--object-select-field {ObjectLog,SystemLog}]
```



```

--object-id OBJECT_ID
                        Logs of object name (default: None)
--object-select-field {ObjectLog,SystemLog}
                        Select field to filter the log (default: None)
--trace TRACE
                        Dump trace buffer (default: None)

```

Example Uses

The following examples show how you can use the option arguments available for **contrail-logs** to retrieve the information you specify.

1. View only the system log messages from all boxes for the last 10 minutes.

contrail-logs

2. View all log messages (systemlog, objectlog, uve, ...) from all boxes for the last 10 minutes.

contrail-logs --all

3. View only the control node system log messages from all boxes for the last 10 minutes.

contrail-logs --module ControlNode

--module accepts the following values - **ControlNode, VRouterAgent, ApiServer, Schema, ServiceMonitor, Collector, OpServer, QueryEngine, DnsAgent**

4. View the control node system log messages from source **a6s23.contrail.juniper.net** for the last 10 minutes.

contrail-logs --module ControlNode --source a6s23.contrail.juniper.net

5. View the XMPP category system log messages from all modules on all boxes for the last 10 minutes.

contrail-logs --category XMPP

6. View the system log messages from all the boxes from the last hour.

contrail-logs --last 1h

7. View the system log messages from the VN object named **demo:admin:vn1** from all boxes for the last 10 minutes.

contrail-logs --object ObjectVNTTable --object-id demo:admin:vn1

--object accepts the following values - **ObjectVNTTable, ObjectVMTable, ObjectSITable, ObjectVRouter, ObjectBgpPeer, ObjectRoutingInstance, ObjectBgpRouter, ObjectXmppConnection, ObjectCollectorInfo**

8. View the system log messages from all boxes for the last 10 minutes in reverse chronological order:

contrail-logs --reverse

9. View the system log messages from a specific time interval and display them in a specified date format.

contrail-logs --start-time "2013 May 12 18:30:27.0" --end-time "2013 May 12 18:31:27.0"

contrail-status (Viewing Node Status)

Syntax

```
[root@host ~]# contrail-status
```

Release Information

Command introduced in Contrail Release 1.0.

Description

Display a list of all components of a Contrail server node (such as control, configuration, database, Web-UI, analytics, or vrouter) and report their current status of active or inactive.

Required Privilege Level

admin

Sample Output

The following example usage displays on a server that is configured for the roles of **vrouter**, **controller**, **analytics**, **configuration**, **web-ui**, and **database**.

Sample Output

```
root@host:~# contrail-status
== Contrail vRouter ==
supervisor-vrouter:           active
contrail-vrouter-agent        active
contrail-vrouter-nodemgr      active

== Contrail Control ==
supervisor-control:           active
contrail-control               active
contrail-control-nodemgr      active
contrail-dns                   active
contrail-named                 active

== Contrail Analytics ==
supervisor-analytics:         active
contrail-analytics-api         active
contrail-analytics-nodemgr     active
contrail-collector             active
contrail-query-engine          active
```

```
== Contrail Config ==
supervisor-config:      active
contrail-api:0          active
contrail-config-nodemgr active
contrail-discovery:0    active
contrail-schema         active
contrail-svc-monitor    active
ifmap                  active
rabbitmq-server         active

== Contrail Web UI ==
supervisor-webui:      active
contrail-webui         active
contrail-webui-middleware active
redis-webui           active

== Contrail Database ==
supervisord-contrail-database:active
contrail-database      active
contrail-database-nodemgr active
```

contrail-version (Viewing Version Information)

Syntax

```
[root@host]# contrail-version
```

Release Information

Command introduced in Contrail Release 1.0.

Description

Display a list of all installed components with their version and build numbers.

Required Privilege Level

admin

Sample Output

The following example shows version and build information for all installed components.

Sample Output

```
root@host> contrail-version
```

Package RPM Name	Version	Build-ID Repo
-----	-----	
contrail-analytics	1-1309090026.el6	141
contrail-analytics-venv	0.1-1309062310.el6	141
contrail-api	0.1-1309090026.el6	141
contrail-api-lib	0.1-1309090026.el6	141
contrail-api-venv	0.1-1309080539.el6	141
contrail-control	2012.0-1309090026.el6	141
contrail-database	0.1-1309050028	141
contrail-dns	1-1309090026.el6	141
contrail-fabric-utils	1-1309090026	141
contrail-libs	1-1309090026.el6	141
contrail-nodejs	0.8.15-1309090026.el6	141
contrail-openstack-analytics	0.1-1309090026.el6	141
contrail-openstack-cfgm	0.1-1309090026.el6	141
contrail-openstack-control	0.1-1309090026.el6	141

Sample Output

The following example shows version and build information for only the installed contrail components.

Sample Output

```
root@host> contrail-version | grep contrail
```

Package RPM Name	Version	Build-ID Repo
-----	-----	
contrail-analytics	1-1309090026.el6	141
contrail-analytics-venv	0.1-1309062310.el6	141
contrail-api	0.1-1309090026.el6	141
contrail-api-lib	0.1-1309090026.el6	141
contrail-api-venv	0.1-1309080539.el6	141
contrail-control	2012.0-1309090026.el6	141
contrail-database	0.1-1309050028	141
contrail-dns	1-1309090026.el6	141
contrail-fabric-utils	1-1309090026	141
contrail-libs	1-1309090026.el6	141
contrail-nodejs	0.8.15-1309090026.el6	141
contrail-openstack-analytics	0.1-1309090026.el6	141
contrail-openstack-cfgm	0.1-1309090026.el6	141
contrail-openstack-control	0.1-1309090026.el6	141
contrail-openstack-database	0.1-1309090026.el6	141
contrail-openstack-webui	0.1-1309090026.el6	141
contrail-setup	1-1309090026.el6	141

contrail-webui	1-1309090026	141
openstack-quantum-contrail	2013.2-1309090026	141

Backing Up Contrail Databases Using JSON Format

IN THIS SECTION

- [Preliminary Cautions | 305](#)
- [Simple Backup Using JSON Format | 306](#)
- [Restore Simple Database Backup | 306](#)
- [Example Backup and Restore With JSON | 307](#)

This document shows how to backup Contrail databases (Cassandra and Zookeeper) using a JSON format. Instructions are given for both non-containerized and containerized versions of Contrail, starting with Contrail 4.0.

Preliminary Cautions



CAUTION: Because the state of the Contrail database is associated with other system databases, such as OpenStack databases, database backups must be consistent across all systems and database changes associated with northbound APIs must be stopped on all systems before performing any backup operation. For example, you might block the external VIP for northbound APIs at the load balancer level, such as HAproxy.

Simple Backup Using JSON Format

Perform a simple backup (database dump). Working from a controller node, use **db_json_exim.py**, located at **/usr/lib/python2.7/dist-packages/cfgm_common**.

NOTE: The controller node for non-containerized Contrail is a virtual machine (VM).

The controller node for containerized Contrail is a controller container.

```
cd /usr/lib/python2.7/dist-packages/cfgm_common
```

```
python db_json_exim.py --export-to db-dump.json
```

- To see a cleaner version of the dump.

```
cat db-dump.json | python -m json.tool | less
```

- To omit keyspace in the dump, for example, to share with Juniper.

```
python db_json_exim.py --export-to db-dump.json --omit-keyspace dm_keyspace
```

Restore Simple Database Backup

Use the following steps to restore a system from a simple backup.

1. Stop **supervisor-config** on all controllers, if present, or ensure it is already stopped.

```
service supervisor-config stop
```

2. Stop Cassandra on all **config-db** controllers or ensure it is already stopped.

```
service cassandra stop
```

3. Stop Zookeeper on all controllers or ensure it is already stopped.

```
service zookeeper stop
```

4. Stop Kafka on all controllers. Be sure to check analytics controllers.

```
service kafka stop
```

5. Stop **contrail-hamon** on all controllers, if it is running on controllers.

```
service contrail-hamon stop
```

6. Backup the Zookeeper data directory on all controllers.

```
cd /var/lib/zookeeper/
cp -R version-2/ version-2-save
```

7. Backup the Cassandra data directory on all controllers.

```
cd /var/lib/
cp -R cassandra cassandra-save
```

8. Wipe out the Zookeeper data directory contents on all controllers.

```
rm -rf /var/lib/zookeeper/version-2/*
```

9. Wipe out the Cassandra data directory contents on all controllers.

```
rm -rf /var/lib/cassandra/*
```

10. Start Zookeeper on all controllers.

```
service zookeeper start
```

11. Start Cassandra on all controllers.

```
service cassandra start
```

12. Run `python db_json_exim.py --import-from db-dump.json` on any one controller.

```
cd /usr/lib/python2.7/dist-packages/cfgm_common
python db_json_exim.py --import-from db-dump.json
```

13. Start `supervisor-config` on all controllers (if present).

```
service supervisor-config start
```

14. Start Kafka on all controllers (check in analytics controllers).

```
service kafka start
```

15. Start `contrail-hamon` on all controllers, if previously stopped.

```
service contrail-hamon start
```

Example Backup and Restore With JSON

This section provides an example of a simple database backup and restore of a system that has three controllers with config-db and separate IPs with the following host IDs:

- 5b5s42
- 5b5s43
- 5b5s44

Example: Perform Simple Backup

```
root@5b5s42:~# python db_json_exim.py --export-to db-dump.json
root@5b5s42:~# cat db-dump.json | python -m json.tool | less
{
  "cassandra": {
    "config_db_uuid": {
      "obj_fq_name_table": {
        "access_control_list": {

<snip>
```

Example: Perform Restore

1. Stop **supervisor-config** on all controllers, if present.

Non-Containerized Version:

```

root@5b5s42:~# service supervisor-config stop
supervisor-config stop/waiting
root@5b5s42:~#
root@5b5s43:~# service supervisor-config stop
supervisor-config stop/waiting
root@5b5s43:~#
root@5b5s44:~# service supervisor-config stop
supervisor-config stop/waiting
root@5b5s44:~#

```

Containerized Version:

```

root@host-4.1:~# docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
8802395bc033	172.30.109.59:5100/contrail410-contrail-analytics:mainline	"/lib/systemd/syst..."	7 weeks ago	Up 2 weeks	
	analytics				
f5aed0a2efc3	172.30.109.59:5100/contrail410-contrail-analyticsdb:mainline	"/lib/systemd/syst..."	7 weeks ago	Up 2 weeks	
	analyticsdb				
0ff200b12112	172.30.109.59:5100/contrail410-contrail-controller:mainline	"/lib/systemd/syst..."	7 weeks ago	Up 2 weeks	
	controller				
6fec888f8145	registry:2	"/entrypoint.sh /e..."	7 weeks ago	Up 2 weeks	
	registry				

```

root@host-4.1:~# docker exec -it 0ff200b12112 /bin/bash

```

2. Stop **Cassandra** on all controllers.

```

root@5b5s42:~# service cassandra stop
root@5b5s42:~#
root@5b5s43:~# service cassandra stop
root@5b5s43:~#
root@5b5s44:~# service cassandra stop
root@5b5s44:~#

```

3. Stop **Zookeeper** on all controllers.

```

root@5b5s42:~# service zookeeper stop
zookeeper stop/waiting
root@5b5s42:~#
root@5b5s43:~# service zookeeper stop
zookeeper stop/waiting
root@5b5s43:~#
root@5b5s44:~# service zookeeper stop
zookeeper stop/waiting
root@5b5s44:~#

```

4. Stop Kafka on all controllers.

```

root@5b5s42:~# service kafka stop
kafka: stopped
root@5b5s42:~#
root@5b5s43:~# service kafka stop
kafka: stopped
root@5b5s43:~#
root@5b5s44:~# service kafka stop
kafka: stopped
root@5b5s44:~#

```

5. Stop **contrail-hamon** on all controllers, if present.

```

root@5b5s42:~# service contrail-hamon stop
contrail-hamon stop/waiting
root@5b5s43:~# service contrail-hamon stop
contrail-hamon stop/waiting
root@5b5s44:~# service contrail-hamon stop
contrail-hamon stop/waiting

```

6. Backup the Zookeeper data directory on all controllers.

```

root@5b5s42:~# cd /var/lib/zookeeper/
root@5b5s42:/var/lib/zookeeper# cp -R version-2/ version-2-save
root@5b5s42:/var/lib/zookeeper#
root@5b5s43:~# cd /var/lib/zookeeper/
root@5b5s43:/var/lib/zookeeper# cp -R version-2/ version-2-save
root@5b5s43:/var/lib/zookeeper#
root@5b5s44:~# cd /var/lib/zookeeper/

```

```
root@5b5s44:/var/lib/zookeeper# cp -R version-2/ version-2-save
root@5b5s44:/var/lib/zookeeper#
```

7. Backup the Cassandra data directory on all controllers.

```
root@5b5s42:~# cd /var/lib/
root@5b5s42:/var/lib# cp -R cassandra cassandra-save
root@5b5s42:/var/lib#
root@5b5s43:~# cd /var/lib/
root@5b5s43:/var/lib# cp -R cassandra cassandra-save
root@5b5s43:/var/lib#
root@5b5s44:~# cd /var/lib/
root@5b5s44:/var/lib# cp -R cassandra/ cassandra-save
root@5b5s44:/var/lib#
```

8. Wipe out the Zookeeper data directory contents on all controllers.

```
root@5b5s42:~# rm -rf /var/lib/zookeeper/version-2/*
root@5b5s42:~#
root@5b5s43:~# rm -rf /var/lib/zookeeper/version-2/*
root@5b5s43:~#
root@5b5s44:~# rm -rf /var/lib/zookeeper/version-2/*
root@5b5s44:~#
```

9. Wipe out the Cassandra data directory contents on all controllers.

```
root@5b5s42:~# rm -rf /var/lib/cassandra/*
root@5b5s42:~#
root@5b5s43:~# rm -rf /var/lib/cassandra/*
root@5b5s43:~#
root@5b5s44:~# rm -rf /var/lib/cassandra/*
root@5b5s44:~#
```

10. Start Zookeeper on all controllers.

```
root@5b5s42:~# service zookeeper start
zookeeper start/running, process 14180
root@5b5s42:~#
root@5b5s43:~# service zookeeper start
zookeeper start/running, process 11635
```



```

root@5b5s43:~#
root@5b5s44:~# service zookeeper start
zookeeper start/running, process 28040
root@5b5s44:~#

```

11. Start Cassandra on all controllers.

```

root@5b5s42:~# service cassandra start
root@5b5s42:~#
root@5b5s43:~# service cassandra start
root@5b5s43:~#
root@5b5s44:~# service cassandra start
root@5b5s44:~#

```

12. Run `python db_json_exim.py --import-from db-dump.json` on any *one* controller.

```

root@5b5s42:~# python db_json_exim.py --import-from db-dump.json
root@5b5s42:~#

```

13. Start `supervisor-config` on all controllers, if present.

```

root@5b5s42:~# service supervisor-config start
supervisor-config start/running, process 19286
root@5b5s42:~#
root@5b5s43:~# service supervisor-config start
supervisor-config start/running, process 28937
root@5b5s43:~#
root@5b5s44:~# service supervisor-config start
supervisor-config start/running, process 21242
root@5b5s44:~#

```

14. Start Kafka on all controllers.

```

root@5b5s42:~# service kafka start
kafka: started
root@5b5s42:~#
root@5b5s43:~# service kafka start
kafka: started
root@5b5s43:~#
root@5b5s44:~# service kafka start

```

```
kafka: started  
root@5b5s44:~#
```

15. Start **contrail-hamon** on all controllers, if present.

```
root@5b5s42:~# service contrail-hamon start  
contrail-hamon start/running, process 1379  
root@5b5s42:~#  
root@5b5s43:~# service contrail-hamon start  
contrail-hamon start/running, process 1230  
root@5b5s43:~#  
root@5b5s44:~# service contrail-hamon start  
contrail-hamon start/running, process 26843  
root@5b5s44:~#
```

Contrail Application Programming Interfaces (APIs)

IN THIS CHAPTER

- [Contrail Analytics Application Programming Interfaces \(APIs\) and User-Visible Entities \(UVEs\) | 314](#)
- [Log and Flow Information APIs | 329](#)
- [Working with Neutron | 337](#)
- [Support for Amazon VPC APIs on Contrail OpenStack | 341](#)

Contrail Analytics Application Programming Interfaces (APIs) and User-Visible Entities (UVEs)

IN THIS SECTION

- [User-Visible Entities | 315](#)
- [Common UVEs in Contrail | 316](#)
- [Virtual Network UVE | 316](#)
- [Virtual Machine UVE | 317](#)
- [vRouter UVE | 317](#)
- [UVEs for Contrail Nodes | 318](#)
- [Wild Card Query of UVEs | 318](#)
- [Filtering UVE Information | 318](#)

The Contrail **analytics-api** server provides a REST API interface to extract the operational state of the Contrail system.

APIs are used by the Contrail Web user interface to present the operational state to users. Other applications might also use the server's REST APIs for analytics or other uses.

This section describes some of the more common APIs and their uses. To see all of the available APIs, navigate the URL tree at the REST interface, starting at the root `http://<ip>:<analytics-api-port>`. You can also view Contrail API information at:

<http://configuration-schema-documentation.s3-website-us-west-1.amazonaws.com/R3.2/>.

User-Visible Entities

In Contrail, a User-Visible Entity (UVE) is an object entity that might span multiple components in Contrail and might require aggregation before the complete information of the UVE is presented. Examples of UVEs in Contrail are virtual network, virtual machine, vRouter, and similar objects. Complete operational information for a virtual network might span multiple vRouters, config nodes, control nodes, and the like. The analytics-api server aggregates all of this information through REST APIs.

To get information about a UVE, you must have the UVE type and the UVE key. In Contrail, UVEs are identified by type, such as virtual network, virtual machine, vRouter, and so on. A system-wide unique key is associated with each UVE. The key type could be different, based on the UVE type. For example, perhaps a virtual network uses its name as its UVE key, and in the same system, a virtual machine uses its UUID as its key.

The URL `/analytics/uves` shows the list of all UVE types available in the system.

The following is sample output from `/analytics/uves`:

```
[
  {
    href: "http://<system IP>:8081/analytics/uves/xmpp-peers",
    name: "xmpp-peers"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/service-instances",
    name: "service-instances"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/config-nodes",
    name: "config-nodes"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/virtual-machines",
    name: "virtual-machines"
  },
  {
    href: "http://<system IP>:8081/analytics/uves/bgp-routers",
    name: "bgp-routers"
  },
]
```

```

{
  href: "http://<system IP>:8081/analytics/uves/collectors",
  name: "collectors"
},
{
  href: "http://<system IP>:8081/analytics/uves/service-chains",
  name: "service-chains"
},
{
  href: "http://<system IP>:8081/analytics/uves/generators",
  name: "generators"
},
{
  href: "http://<system IP>:8081/analytics/uves/bgp-peers",
  name: "bgp-peers"
},
{
  href: "http://<system IP>:8081/analytics/uves/virtual-networks",
  name: "virtual-networks"
},
{
  href: "http://<system IP>:8081/analytics/uves/vrouters",
  name: "vrouters"
},
{
  href: "http://<system IP>:8081/analytics/uves/dns-nodes",
  name: "dns-nodes"
}
]

```

Common UVEs in Contrail

This section presents descriptions of some common UVEs in Contrail.

Virtual Network UVE

This UVE provides information associated with a virtual network, such as:

- list of networks connected to this network
- list of virtual machines spawned in this network
- list of access control lists (ACLs) associated with this virtual network

- global input and output statistics
- input and output statistics per virtual network pair

The REST API to get a UVE for a specific virtual network is through HTTP GET, using the URL:

/analytics/uves/virtual-network/<key>

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

/analytics/uves/virtual-networks

Virtual Machine UVE

This UVE provides information associated with a virtual machine, such as:

- list of interfaces in this virtual machine
- list of floating IPs associated with each interface
- input and output statistics

The REST API to get a UVE for a specific virtual machine is through HTTP GET, using the URL:

/analytics/uves/virtual-machine/<key>

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

/analytics/uves/virtual-machines

vRouter UVE

This UVE provides information associated with a vRouter, such as:

- virtual networks present on this vRouter
- virtual machines spawned on the server of this vRouter
- statistics of the traffic flowing through this vRouter

The REST API to get a UVE for a specific vRouter is through HTTP GET, using the URL:

/analytics/uves/vrouter/<key>

The REST API to get UVEs for all virtual machines is through HTTP GET, using the URL:

/analytics/uves/vrouters

UVEs for Contrail Nodes

There are multiple node types in Contrail (including the node type vRouter previously described). Other node types include control node, config node, analytics node, and compute node.

There is a UVE for each node type. The common information associated with each node UVE includes:

- the IP address of the node
- a list of processes running on the node
- the CPU and memory utilization of the running processes

Each UVE also has node-specific information, such as:

- the control node UVE has information about its connectivity to the vRouter and other control nodes
- the analytics node UVE has information about the number of generators connected

The REST API to get a UVE for a specific config node is through HTTP GET, using the URL:

/analytics/uves/config-node/<key>

The REST API to get UVEs for all config nodes is through HTTP GET, using the URL:

/analytics/uves/config-nodes

NOTE: Use similar syntax to get UVEs for each of the different types of nodes, substituting the node type that you want in place of **config-node**.

Wild Card Query of UVEs

You can use wildcard queries when you want to get multiple UVEs at the same time. Example queries are the following:

The following HTTP GET with wildcard retrieves all virtual network UVEs:

/analytics/uves/virtual-network/*

The following HTTP GET with wildcard retrieves all virtual network UVEs with name starting with **project1**:

/analytics/uves/virtual-network/project1*

Filtering UVE Information

It is possible to retrieve filtered UVE information. The following flags enable you to retrieve partial, filtered information about UVEs.

Supported filter flags include:

- sfilt** : filter by source (usually the hostname of the generator)
- mfilt** : filter by module (the module name of the generator)
- cfilt** : filter by content, useful when only part of a UVE needs to be retrieved
- kfilt** : filter by UVE keys, useful to get multiple, but not all, UVEs of a particular type

Examples

The following HTTP GET with filter retrieves information about virtual network **vn1** as provided by the source **src1**:

```
/analytics/uves/virtual-network/vn1?sfilt=src1
```

The following HTTP GET with filter retrieves information about virtual network **vn1** as provided by all **ApiServer** modules:

```
/analytics/uves/virtual-network/vn1?mfilt=ApiServer
```

Example Output: Virtual Network UVE

Example output for a virtual network UVE:

```
[user@host ~]# curl <system
IP>:8081/analytics/virtual-network/default-domain:demo:front-end | python
-mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current                                  Dload  Upload   Total   Spent    Left  Speed
100  2576  100  2576    0     0   152k      0 --:--:-- --:--:-- --:--:--
157k
{
  "UveVirtualNetworkAgent": {
    "acl": [
      [
        {
          "@type": "string"
        },
        "a3s18:VRouterAgent"
      ]
    ]
  }
}
```



```

    ]
  ],
  "in_bytes": {
    "#text": "2232972057",
    "@aggtype": "counter",
    "@type": "i64"
  },
  "in_stats": {
    "@aggtype": "append",
    "@type": "list",
    "list": {
      "@size": "3",
      "@type": "struct",
      "UveInterVnStats": [
        {
          "bytes": {
            "#text": "2114516371",
            "@type": "i64"
          },
          "other_vn": {
            "#text": "default-domain:demo:back-end",
            "@aggtype": "listkey",
            "@type": "string"
          },
          "tpkts": {
            "#text": "5122001",
            "@type": "i64"
          }
        }
      ],
      {
        "bytes": {
          "#text": "1152123",
          "@type": "i64"
        },
        "other_vn": {
          "#text": "__FABRIC__",
          "@aggtype": "listkey",
          "@type": "string"
        },
        "tpkts": {
          "#text": "11323",
          "@type": "i64"
        }
      }
    ]
  }
}

```

```

        }
    },
    {
        "bytes": {
            "#text": "8192",
            "@type": "i64"
        },
        "other_vn": {
            "#text": "default-domain:demo:front-end",
            "@aggtype": "listkey",
            "@type": "string"
        },
        "tpkts": {
            "#text": "50",
            "@type": "i64"
        }
    }
]
}
},
"in_tpkts": {
    "#text": "5156342",
    "@aggtype": "counter",
    "@type": "i64"
},
"interface_list": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
        "@size": "1",
        "@type": "string",
        "element": [
            "tap2158f77c-ec"
        ]
    }
},
"out_bytes": {
    "#text": "2187615961",
    "@aggtype": "counter",
    "@type": "i64"
},
"out_stats": {

```

```

"@aggtype": "append",
"@type": "list",
"list": {
  "@size": "4",
  "@type": "struct",
  "UveInterVnStats": [
    {
      "bytes": {
        "#text": "2159083215",
        "@type": "i64"
      },
      "other_vn": {
        "#text": "default-domain:demo:back-end",
        "@aggtype": "listkey",
        "@type": "string"
      },
      "tpkts": {
        "#text": "5143693",
        "@type": "i64"
      }
    },
    {
      "bytes": {
        "#text": "1603041",
        "@type": "i64"
      },
      "other_vn": {
        "#text": "__FABRIC__",
        "@aggtype": "listkey",
        "@type": "string"
      },
      "tpkts": {
        "#text": "9595",
        "@type": "i64"
      }
    },
    {
      "bytes": {
        "#text": "24608",
        "@type": "i64"
      },
      "other_vn": {

```

```

        "#text": "__UNKNOWN__",
        "@aggtype": "listkey",
        "@type": "string"
    },
    "tpkts": {
        "#text": "408",
        "@type": "i64"
    }
},
{
    "bytes": {
        "#text": "8192",
        "@type": "i64"
    },
    "other_vn": {
        "#text": "default-domain:demo:front-end",
        "@aggtype": "listkey",
        "@type": "string"
    },
    "tpkts": {
        "#text": "50",
        "@type": "i64"
    }
}
]
}
},
"out_tpkts": {
    "#text": "5134830",
    "@aggtype": "counter",
    "@type": "i64"
},
"virtualmachine_list": {
    "@aggtype": "union",
    "@type": "list",
    "list": {
        "@size": "1",
        "@type": "string",
        "element": [
            "dd09f8c3-32a8-456f-b8cc-fab15189f50f"
        ]
    }
} }

```

```

    },
    "UveVirtualNetworkConfig": {
      "connected_networks": {
        "@aggtype": "union",
        "@type": "list",
        "list": {
          "@size": "1",
          "@type": "string",
          "element": [
            "default-domain:demo:back-end"
          ]
        }
      },
      "routing_instance_list": {
        "@aggtype": "union",
        "@type": "list",
        "list": {
          "@size": "1",
          "@type": "string",
          "element": [
            "front-end"
          ]
        }
      },
      "total_acl_rules": [
        [
          {
            "#text": "3",
            "@type": "i32"
          },
          ":",
          "a3s14:Schema"
        ]
      ]
    }
  }
}

```

Example Output: Virtual Machine UVE

Example output for a virtual machine UVE:

```
[user@host ~]# curl <system
IP>:8081/analytics/virtual-machine/f38eb47e-63d2-4b39-80de-8fe68e6af1e4 |
python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current                                  Dload  Upload   Total   Spent    Left  Speed
100  736  100  736    0    0   160k      0  --:--:--  --:--:--  --:--:--
179k
{
  "UveVirtualMachineAgent": {
    "interface_list": [
      [
        {
          "@type": "list",
          "list": {
            "@size": "1",
            "@type": "struct",
            "VmInterfaceAgent": [
              {
                "in_bytes": {
                  "#text": "2188895907",
                  "@aggtype": "counter",
                  "@type": "i64"
                },
                "in_pkts": {
                  "#text": "5130901",
                  "@aggtype": "counter",
                  "@type": "i64"
                },
                "ip_address": {
                  "#text": "192.168.2.253",
                  "@type": "string"
                },
                "name": {
                  "#text":
"f38eb47e-63d2-4b39-80de-8fe68e6af1e4:ccb085a0-c994-4034-be0f-6fd5ad08ce83",
                  "@type": "string"
                },
                "out_bytes": {
                  "#text": "2201821626",
```

```

        "@aggtype": "counter",
        "@type": "i64"
    },
    "out_pkts": {
        "#text": "5153526",
        "@aggtype": "counter",
        "@type": "i64"
    },
    "virtual_network": {
        "#text": "default-domain:demo:back-end",

        "@aggtype": "listkey",
        "@type": "string"
    }
}
]
}
]
}

```

Example Output: vRouter UVE

Example output for a vRouter UVE:

```

[user@host ~]# curl <system IP>:8081/analytics/vrouter/a3s18 | python
-mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current                                  Dload  Upload  Total   Spent    Left  Speed
100  706  100  706    0     0   142k      0  --:--:-- --:--:-- --:--:--
172k
{
  "VrouterAgent": {
    "collector": [
      {

```

```

        "#text": "10.xx.17.1",
        "@type": "string"
    },
    "a3s18:VRouterAgent"
]
],
"connected_networks": [
    [
        {
            "@type": "list",
            "list": {
                "@size": "1",
                "@type": "string",
                "element": [
                    "default-domain:demo:front-end"
                ]
            }
        },
        "a3s18:VRouterAgent"
    ]
],
"interface_list": [
    [
        {
            "@type": "list",
            "list": {
                "@size": "1",
                "@type": "string",
                "element": [
                    "tap2158f77c-ec"
                ]
            }
        },
        "a3s18:VRouterAgent"
    ]
],
"virtual_machine_list": [
    [
        {
            "@type": "list",
            "list": {
                "@size": "1",

```



```

        "@type": "string",
        "element": [
            "dd09f8c3-32a8-456f-b8cc-fab15189f50f"
        ]
    },
    "a3s18:VRouterAgent"
]
],
"xmpp_peer_list": [
    [
        {
            "@type": "list",
            "list": {
                "@size": "2",
                "@type": "string",
                "element": [
                    "10.xx.17.2",
                    "10.xx.17.3"
                ]
            }
        },
        "a3s18:VRouterAgent"
    ]
]
}
}

```

RELATED DOCUMENTATION

[Juniper Contrail Configuration API Server Documentation](#)

[Log and Flow Information APIs](#) | 329

Log and Flow Information APIs

IN THIS SECTION

- [HTTP GET APIs | 329](#)
- [HTTP POST API | 329](#)
- [POST Data Format Example | 330](#)
- [Query Types | 332](#)
- [Examining Query Status | 332](#)
- [Examining Query Chunks | 332](#)
- [Example Queries for Log and Flow Data | 333](#)

In Contrail, log and flow analytics information is collected and stored using a horizontally scalable Contrail collector and NoSQL database. The **analytics-api** server provides REST APIs to extract this information using queries. The queries use well-known SQL syntax, hiding the underlying complexity of the NoSQL tables.

HTTP GET APIs

Use the following GET APIs to identify tables and APIs available for querying.

/analytics/tables -- lists the SQL-type tables available for querying, including the hrefs for each of the tables

/analytics/table/<table> -- lists the APIs available to get information for a given table

/analytics/table/<table>/schema -- lists the schema for a given table

HTTP POST API

Use the following POST API information to extract data from a table.

/analytics/query -- format your query using the following SQL syntax:

```
SELECT field1, field2 ...
FROM table1
WHERE field1 = value1 AND field2 = value2 ...
FILTER BY ...
```

SORT BY ...

LIMIT *n*

Additionally, it is mandatory to include the start time and the end time for the data range to define the time period for the query data. The parameters of the query are passed through POST data, using the following fields:

start_time — the start of the time period

end_time — the end of the time period

table — the table from which to extract data

select_fields — the columns to display in the extracted data

where — the list of match conditions

POST Data Format Example

The POST data is in **JSON** format, stored in an **idl** file. A sample file is displayed in the following.

NOTE: The result of the query API is also in **JSON** format.

```
/*
 * Copyright (c) 2013 Juniper Networks, Inc. All rights reserved.
 */

/*
 * query_rest.idl
 *
 * IDL definitions for query engine REST API
 *
 * PLEASE NOTE: After updating this file, do update json_parse.h
 *
 */

enum match_op {
    EQUAL = 1,
    NOT_EQUAL = 2,
    IN_RANGE = 3,
    NOT_IN_RANGE = 4,    // not supported currently
    // following are only for numerical column fields
    LEQ = 5, // column value is less than or equal to filter value
}
```

```

    GEQ = 6, // column value is greater than or equal to filter value
    PREFIX = 7, // column value has the "value" field as prefix
    REGEX_MATCH = 8 // for filters only
}

enum sort_op {
    ASCENDING = 1,
    DESCENDING = 2,
}

struct match {
    1: string name;
    2: string value;
    3: match_op op;
    4: optional string value2; // this is for only RANGE match
}

typedef list<match> term; (AND of match)

enum flow_dir_t {
    EGRESS = 0,
    INGRESS = 1
}

struct query {
    1: string table; // Table to query (FlowSeriesTable, MessageTable,
ObjectVNTTable, ObjectVMTable, FlowRecordTable)
    2: i64 start_time; // Microseconds in UTC since Epoch
    3: i64 end_time; // Microseconds in UTC since Epoch
    4: list<string> select_fields; // List of SELECT fields
    5: list<term> where; // WHERE (OR of terms)
    6: optional sort_op sort;
    7: optional list<string> sort_fields;
    8: optional i32 limit;
    9: optional flow_dir_t dir; // direction of flows being queried
    10: optional list<match> filter; // filter the processed result by value
}

struct flow_series_result_entry {
    1: optional i64 T; // Timestamp of the flow record
    2: optional string sourcevn;
    3: optional string sourceip;
    4: optional string destvn;
    5: optional string destip;
    6: optional i32 protocol;

```

```

7: optional i32 sport;
8: optional i32 dport;
9: optional flow_dir_t direction_ing;
10: optional i64 packets; // mutually exclusive to 12,13
11: optional i64 bytes; // mutually exclusive to 12,13
12: optional i64 sum_packets; // represented as "sum(packets)" in JSON
13: optional i64 sum_bytes; // represented as "sum(bytes)" in JSON
};
typedef list<flow_series_result_entry> flow_series_result;

```

Query Types

The **analytics-api** supports two types of queries. Both types use the same POST parameters as described in POST API.

- **sync** — Default query mode. The results are sent inline with the query processing.
- **async** — To execute a query in async mode, attach the following header to the POST request: **Expect: 202-accepted**.

Examining Query Status

For an asynchronous query, the **analytics-api** responds with the code: **202 Accepted**. The response contents are a status entity href URL of the form: **/analytics/query/<QueryID>**. The QueryID is assigned by the **analytics-api**. To view the response contents, poll the status entity by performing a GET action on the URL. The status entity has a variable named **progress**, with a number between 0 and 100, representing the approximate percentage completion of the query. When progress is 100, the query processing is complete.

Examining Query Chunks

The status entity has an element named **chunks** that lists portions (chunks) of query results. Each element of this list has three fields: **start_time**, **end_time**, **href**. The **analytics-api** determines how many chunks to list to represent the query data. A chunk can include an empty string ("") to indicate that the data query is not yet available. If a partial result is available, the chunk href is of the form: **/analytics/query/<QueryID>/chunk-partial/<chunk number>**. When the final result of a chunk is available, the href is of the form: **/analytics/query/<QueryID>/chunk-final/<chunk number>**.

Example Queries for Log and Flow Data

The following example query lists the tables available for query.

```
[root@host ~]# curl 127.0.0.1:8081/analytics/tables | python -mjson.tool
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100    846    100    846      0      0   509k      0 --:--:-- --:--:-- --:--:--   826k
[
  {
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable",
    "name": "MessageTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVNTable",
    "name": "ObjectVNTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVMTable",
    "name": "ObjectVMTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectVRouter",
    "name": "ObjectVRouter"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectBgpPeer",
    "name": "ObjectBgpPeer"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectRoutingInstance",
    "name": "ObjectRoutingInstance"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/ObjectXmppConnection",
    "name": "ObjectXmppConnection"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/FlowRecordTable",
    "name": "FlowRecordTable"
  },
  {
    "href": "http://127.0.0.1:8081/analytics/table/FlowSeriesTable",
    "name": "FlowSeriesTable"
  }
]
```

```
}  
]
```

The following example query lists details for the table named **MessageTable**.

```
[root@host ~]# curl 127.0.0.1:8081/analytics/table/MessageTable | python -mjson.tool  
  
% Total    % Received % Xferd  Average Speed   Time    Time       Time  Current  
          Dload  Upload   Total     Spent    Left     Speed  
100   192   100   192     0     0   102k      0 --:--:-- --:--:-- --:--:--   187k  
[  
  {  
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable/schema",  
    "name": "schema"  
  },  
  {  
    "href": "http://127.0.0.1:8081/analytics/table/MessageTable/column-values",  
    "name": "column-values"  
  }  
]
```

The following example query lists the schema for the table named **MessageTable**.

```
[root@host ~]# curl 127.0.0.1:8081/analytics/table/MessageTable/schema | python -mjson.tool  
  
% Total    % Received % Xferd  Average Speed   Time    Time       Time  Current  
          Dload  Upload   Total     Spent    Left     Speed  
100    630   100    630     0     0   275k      0 --:--:-- --:--:-- --:--:--   307k  
{  
  "columns": [  
    {  
      "datatype": "int",  
      "index": "False",  
      "name": "MessageTS"  
    },  
    {  
      "datatype": "string",  
      "index": "True",  
      "name": "Source"  
    },  
  ],  
}
```

```

    {
      "datatype": "string",
      "index": "True",
      "name": "ModuleId"
    },
    {
      "datatype": "string",
      "index": "True",
      "name": "Category"
    },
    {
      "datatype": "int",
      "index": "True",
      "name": "Level"
    },
    {
      "datatype": "int",
      "index": "False",
      "name": "Type"
    },
    {
      "datatype": "string",
      "index": "True",
      "name": "Messagetype"
    },
    {
      "datatype": "int",
      "index": "False",
      "name": "SequenceNum"
    },
    {
      "datatype": "string",
      "index": "False",
      "name": "Context"
    },
    {
      "datatype": "string",
      "index": "False",
      "name": "Xmlmessage"
    }
  ],
  "type": "LOG"
}

```


The following set of example queries explore a message table.

```

root@a6s45:~# cat filename
{ "end_time": "now" , "select_fields": ["MessageTS", "Source", "ModuleId",
"Category", "Messagetype", "SequenceNum", "Xmlmessage", "Type", "Level", "NodeType",
"InstanceId"] , "sort": 1 , "sort_fields": ["MessageTS"] , "start_time": "now-10m"
, "table": "MessageTable" , "where": { "name": "ModuleId", "value":
"contrail-control", "op": 1, "suffix": null, "value2": null}, { "name":
"Messagetype", "value": "BGPRouterInfo", "op": 1, "suffix": null, "value2": null}
}

root@a6s45:~#
root@a6s45:~# curl -X POST --data @filename 127.0.0.1:8081/analytics/query --header
"Content-Type:application/json" | python -mjson.tool
  % Total    % Received % Xferd  Average Speed   Time    Time       Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
100  9765    0  9297  100    468    9168    461   0:00:01   0:00:01  -:--:--   9177
{
  "value": [
    {
      "Category": null,
      "InstanceId": "0",
      "Level": 2147483647,
      "MessageTS": 1428442589947392,
      "Messagetype": "BGPRouterInfo",
      "ModuleId": "contrail-control",
      "NodeType": "Control",
      "SequenceNum": 1302,
      "Source": "a6s45",
      "Type": 6,
      "Xmlmessage": "<BGPRouterInfo type=\"><data
type=\"><BgpRouterState><name type=\"
>a6s45</name><cpu_info type=\"><CpuLoadInfo><num_cpu type=\">4</num_cpu
><meminfo type=\"><MemInfo><virt type=\">438436</virt><peakvirt type=\"
>561048</peakvirt><res type=\">12016</res></MemInfo></meminfo><cpu_share
type=\">0.0416667</cpu_share></CpuLoadInfo></cpu_info><cpu_share type=\"
>0.0416667</cpu_share></BgpRouterState></data></BGPRouterInfo>"
    },
    {
      "Category": null,
      "InstanceId": "0",
      "Level": 2147483647,

```

...

Working with Neutron

IN THIS SECTION

- [Data Structure | 337](#)
- [Network Sharing in Neutron | 338](#)
- [Commands for Neutron Network Sharing | 338](#)
- [Support for Neutron APIs | 339](#)
- [Contrail Neutron Plugin | 339](#)
- [DHCP Options | 340](#)
- [Incompatibilities | 340](#)

OpenStack's networking solution, Neutron, has representative elements for Contrail elements for Network (VirtualNetwork), Port (VirtualMachineInterface), Subnet (IpamSubnets), and Security-Group. The Neutron plugin translates the elements from one representation to another.

Data Structure

Although the actual data between Neutron and Contrail is similar, the listings of the elements differs significantly. In the Contrail API, the networking elements list is a summary, containing only the UUID, FQ name, and an href, however, in Neutron, all details of each resource are included in the list.

The Neutron plugin has an inefficient list retrieval operation, especially at scale, because it:

- reads a list of resources (for example. **GET /virtual-networks**), then
- iterates and reads in the details of the resource (**GET /virtual-network/<uuid>**).

As a result, the API server spends most of the time in this type of GET operation just waiting for results from the Cassandra database.

The following features in Contrail improve performance with Neutron:

- An optional detail query parameter is added in the GET of collections so that the API server returns details of all the resources in the list, instead of just a summary. This is accompanied by changes in the Contrail API library so that a caller gets returned a list of the objects.
- The existing Contrail list API takes in an optional **parent_id** query parameter to return information about the resource anchored by the parent.

- The Contrail API server reads objects from Cassandra in a multiget format into **obj_uuid_cf**, where object contents are stored, instead of reading in an xget/get format. This reduces the number of round-trips to and from the Cassandra database.

Network Sharing in Neutron

Using Neutron, a deployer can make a network accessible to other tenants or projects by using one of two attributes on a network:

- set the **shared** attribute to allow sharing
- set the **router:external** attribute, when the plugin supports an **external_net** extension

Using the Shared Attribute

When a network has the **shared** attribute set, users in other tenants or projects, including non-admin users, can access that network, using:

neutron net-list --shared

Users can also launch a virtual machine directly on that network, using:

nova boot <other-parameters> --nic net-id=<shared-net-id>

Using the Router:External Attribute

When a network has the **router:external** attribute set, users in other tenants or projects, including non-admin users, can use that network for allocating floating IPs, using:

neutron floatingip-create <router-external-net-id>

then associating the IP address pool with their instances.

NOTE: The VN hosting the FIP pool should be marked shared and external.

Commands for Neutron Network Sharing

The following table summarizes the most common Neutron commands used with Contrail.

Action	Command
List all shared networks.	neutron net-list --shared
Create a network that has the shared attribute.	neutron net-create <net-name> --shared

Action	Command
Set the shared attribute on an existing network.	neutron net-update <net-name> -shared
List all router:external networks.	neutron net-list --router:external
Create a network that has the router:external attribute.	neutron net-create <net-name> -router:external
Set the router:external attribute on an existing network.	neutron net-update <net-name> -router:external

Support for Neutron APIs

The OpenStack Neutron project provides virtual networking services among devices that are managed by the OpenStack compute service. Software developers create applications by using the OpenStack Networking API v2.0 (Neutron).

Contrail provides the following features to increase support for OpenStack Neutron:

- Create a port independently of a virtual machine.
- Support for more than one subnet on a virtual network.
- Support for allocation pools on a subnet.
- Per tenant quotas.
- Enabling DHCP on a subnet.
- External router can be used for floating IPs.

For more information about using OpenStack Networking API v2.0 (Neutron), refer to: <http://docs.openstack.org/api/openstack-network/2.0/content/> and the OpenStack Neutron Wiki at: <http://wiki.openstack.org/wiki/Neutron>.

Contrail Neutron Plugin

The Contrail Neutron plugin provides an implementation for the following core resources:

- Network
- Subnet
- Port

It also implements the following standard and upstreamed Neutron extensions:

- Security group
- Router IP and floating IP

- Per-tenant quota
- Allowed address pair

The following Contrail-specific extensions are implemented:

- Network IPAM
- Network policy
- VPC table and route table
- Floating IP pools

The plugin does not implement native bulk, pagination, or sort operations and relies on emulation provided by the Neutron common code.

DHCP Options

In Neutron commands, DHCP options can be configured using `extra-dhcp-options` in `port-create`.

Example

```
neutron port-create net1 --extra-dhcp-opt
opt_name=<dhcp_option_name>,opt_value=<value>
```

The `opt_name` and `opt_value` pairs that can be used are maintained in GitHub:

<https://github.com/Juniper/contrail-controller/wiki/Extra-DHCP-Options> .

Incompatibilities

In the Contrail architecture, the following are known incompatibilities with the Neutron API.

- Filtering based on any arbitrary key in the resource is not supported. The only supported filtering is by **id**, **name**, and **tenant_id**.
- To use a floating IP, it is not necessary to connect the public subnet and the private subnet to a Neutron router. Marking a public network with **router:external** is sufficient for a floating IP to be created and associated, and packet forwarding to it will work.
- The default values for quotas are sourced from `/etc/contrail/contrail-api.conf` and not from `/etc/neutron/neutron.conf`.

Support for Amazon VPC APIs on Contrail OpenStack

IN THIS SECTION

- [Overview of Amazon Virtual Private Cloud | 341](#)
- [Mapping Amazon VPC Features to OpenStack Contrail Features | 342](#)
- [VPC and Subnets Example | 342](#)
- [Euca2ools CLI for VPC and Subnets | 343](#)
- [Security in VPC: Network ACLs Example | 344](#)
- [Euca2ools CLI for Network ACLs | 345](#)
- [Security in VPC: Security Groups Example | 346](#)
- [Euca2ools CLI for Security Groups | 347](#)
- [Elastic IPs in VPC | 347](#)
- [Euca2ools CLI for Elastic IPs | 348](#)
- [Euca2ools CLI for Route Tables | 348](#)
- [Supported Next Hops | 349](#)
- [Internet Gateway Next Hop Euca2ools CLI | 349](#)
- [NAT Instance Next Hop Euca2ools CLI | 349](#)
- [Example: Creating a NAT Instance with Euca2ools CLI | 350](#)

Overview of Amazon Virtual Private Cloud

The current Grizzly release of OpenStack supports Elastic Compute Cloud (EC2) API translation to OpenStack Nova, Quantum, and Keystone calls. EC2 APIs are used in Amazon Web Services (AWS) and virtual private clouds (VPCs) to launch virtual machines, assign IP addresses to virtual machines, and so on. A VPC provides a container where applications can be launched and resources can be accessed over the networking services provided by the VPC.

Contrail enhances its use of EC2 APIs to support the Amazon VPC APIs.

The Amazon VPC supports networking constructs such as: subnets, DHCP options, elastic IP addresses, network ACLs, security groups, and route tables. The Amazon VPC APIs are now supported on the Openstack Contrail distribution, so users of the Amazon EC2 APIs for their VPC can use the same scripts to move to an Openstack Contrail solution.

Euca2ools are command-line tools for interacting with Amazon Web Services (AWS) and other AWS-compatible web services, such as OpenStack. **Euca2ools** have been extended in OpenStack Contrail to add support for the Amazon VPC, similar to the support that already exists for the Amazon EC2 CLI.

For more information about Amazon VPC and AWS EC2, see:

- Amazon VPC documentation: http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html
- Amazon VPC API list: <http://docs.aws.amazon.com/AWSEC2/latest/APIReference/query-apis.html>

Mapping Amazon VPC Features to OpenStack Contrail Features

The following table compares Amazon VPC features to their equivalent features in OpenStack Contrail.

Table 66: Amazon VPC and OpenStack Contrail Feature Comparison

Amazon VPC Feature	OpenStack Contrail Feature
VPC	Project
Subnets	Networks (Virtual Networks)
DHCP options	IPAM
Elastic IP	Floating IP
Network ACLs	Network ACLs
Security Groups	Security Groups
Route Table	Route Table

VPC and Subnets Example

When creating a new VPC, the user must provide a classless inter-domain routing (CIDR) block of which all subnets in this VPC will be part.

In the following example, a VPC is created with a CIDR block of 10.1.0.0/16. A subnet is created within the VPC CIDR block, with a CIDR block of 10.1.1.0/24. The VPC has a default network ACL named **acl-default**.

All subnets created in the VPC are automatically associated to the default network ACL. This association can be changed when a new network ACL is created. The last command in the list below creates a virtual machine using the image **ami-00000003** and launches with an interface in **subnet-5eb34ed2**.

```
# euca-create-vpc 10.1.0.0/16
VPC VPC:vpc-8352aa59 created

# euca-describe-vpcs
VpcId          CidrBlock      DhcpOptions
-----
vpc-8352aa59   10.1.0.0/16    None

# euca-create-subnet -c 10.1.1.0/24 vpc-8352aa59
Subnet: subnet-5eb34ed2 created

# euca-describe-subnets
Subnet-id      Vpc-id         CidrBlock
-----
subnet-5eb34ed2 vpc-8352aa59   10.1.1.0/24

# euca-describe-network-acls
AclId
-----
acl-default(def)
vpc-8352aa59

      Rule    Dir    Action  Proto  Port  Range  Cidr
      ----    --    -
      100     ingress allow   -1     0     65535  0.0.0.0/0
      100     egress  allow   -1     0     65535  0.0.0.0/0
      32767   ingress deny    -1     0     65535  0.0.0.0/0
      32767   egress  deny    -1     0     65535  0.0.0.0/0

      Association      SubnetId      AclId
      -----
      aclassoc-0c549d66 subnet-5eb34ed2 acl-default

# euca-run-instances -s subnet-5eb34ed2 ami-00000003
```

Euca2ools CLI for VPC and Subnets

The following **euca2ools** CLI commands are used to create, define, and delete VPCs and subnets:

- euca-create-vpc
- euca-delete-vpc
- euca-describe-vpcs
- euca-create-subnet
- euca-delete-subnet
- euca-describe-subnets

Security in VPC: Network ACLs Example

Network ACLs support ingress and egress rules for traffic classification and filtering. The network ACLs are applied at a subnet level.

In the following example, a new ACL, **acl-ba7158**, is created and an existing subnet is associated to the new ACL.

```
# euca-create-network-acl vpc-8352aa59
acl-ba7158c

# euca-describe-network-acls
AclId
-----
acl-default(def)
vpc-8352aa59

      Rule   Dir   Action  Proto  Port  Range  Cidr
      ----   --   -
      100    ingress allow   -1     0    65535  0.0.0.0/0
      100    egress  allow   -1     0    65535  0.0.0.0/0
      32767  ingress deny    -1     0    65535  0.0.0.0/0
      32767  egress  deny    -1     0    65535  0.0.0.0/0

      Association      SubnetId      AclId
      -----
      aclassoc-0c549d66  subnet-5eb34ed2  acl-default

AclId
-----
acl-ba7158c
vpc-8352aa59

      Rule   Dir   Action  Proto  Port  Range  Cidr
      ----   --   -
      32767  ingress deny    -1     0    65535  0.0.0.0/0
```

```

32767    egress    deny    -1      0      65535    0.0.0.0/0

# euca-replace-network-acl-association -a aclassoc-0c549d66 acl-ba7158c
aclassoc-0c549d66

# euca-describe-network-acls
AclId
-----
acl-default(def)
vpc-8352aa59
      Rule      Dir      Action    Proto    Port    Range    Cidr
      ----      ---      -
      100      ingress    allow    -1      0      65535    0.0.0.0/0
      100      egress     allow    -1      0      65535    0.0.0.0/0
      32767    ingress    deny     -1      0      65535    0.0.0.0/0
      32767    egress     deny     -1      0      65535    0.0.0.0/0

      Association      SubnetId      AclId
      -----
      -----
      -----

AclId
-----
acl-ba7158c
vpc-8352aa59
      Rule      Dir      Action    Proto    Port    Range    Cidr
      ----      ---      -
      32767    ingress    deny     -1      0      65535    0.0.0.0/0
      32767    egress     deny     -1      0      65535    0.0.0.0/0

      Association      SubnetId      AclId
      -----
      -----
      -----
      aclassoc-0c549d66    subnet-5eb34ed2    acl-ba7158c

```

Euca2ools CLI for Network ACLs

The following **euca2ools** CLI commands are used to create, define, and delete VPCs and subnets:

- `euca-create-network-acl`
- `euca-delete-network-acl`
- `euca-replace-network-acl-association`
- `euca-describe-network-acls`
- `euca-create-network-acl-entry`
- `euca-delete-network-acl-entry`
- `euca-replace-network-acl-entry`

Security in VPC: Security Groups Example

Security groups provide virtual machine level ingress/egress controls. Security groups are applied to virtual machine interfaces.

In the following example, a new security group is created. The rules can be added or removed for the security group based on the commands listed for **euca2ools**. The last line launches a virtual machine using the newly created security group.

```
# euca-describe-security-groups
```

GroupId	VpcId	Name	Description
-----	-----	----	-----
sg-6d89d7e2	vpc-8352aa59	default	

	Direction	Proto	Start	End	Remote
	-----	-----	-----	---	-----
	Ingress	any	0	65535	[0.0.0.0/0]
	Egress	any	0	65535	[0.0.0.0/0]


```
# euca-create-security-group -d "TestGroup" -v vpc-8352aa59 testgroup
GROUP sg-c5b9d22a testgroup TestGroup
```



```
# euca-describe-security-groups
```

GroupId	VpcId	Name	Description
-----	-----	----	-----
sg-6d89d7e2	vpc-8352aa59	default	

	Direction	Proto	Start	End	Remote

```

-----
Ingress      any      0      65535    [0.0.0.0/0]
Egress       any      0      65535    [0.0.0.0/0]

GroupId      VpcId      Name      Description
-----
sg-c5b9d22a  vpc-8352aa59  testgroup  TestGroup

Direction    Proto      Start     End       Remote
-----
Egress       any      0      65535    [0.0.0.0/0]

# euca-run-instances -s subnet-5eb34ed2 -g testgroup ami-000000003

```

Euca2ools CLI for Security Groups

The following **euca2ools** CLI commands are used to create, define, and delete security groups:

- **euca-create-security-group**
- **euca-delete-security-group**
- **euca-describe-security-groups**
- **euca-authorize-security-group-egress**
- **euca-authorize-security-group-ingress**
- **euca-revoke-security-group-egress**
- **euca-revoke-security-group-ingress**

Elastic IPs in VPC

Elastic IPs in VPCs are equivalent to the floating IPs in the Contrail Openstack solution.

In the following example, a floating IP is requested from the system and assigned to a particular virtual machine. The prerequisite is that the provider or Contrail administrator has provisioned a network named “public” and allocated a floating IP pool to it. This “public” floating IP pool is then internally used by the tenants to request public IP addresses that they can use and attach to virtual machines.

```

# euca-allocate-address --domain vpc
ADDRESS 10.84.14.253      eipalloc-78d9a8c9

```

```
# euca-describe-addresses --filter domain=vpc
Address          Domain    AllocationId      InstanceId(AssociationId)
-----
10.84.14.253     vpc       eipalloc-78d9a8c9

# euca-associate-address -a eipalloc-78d9a8c9 i-00000008
ADDRESS eipassoc-78d9a8c9

# euca-describe-addresses --filter domain=vpc
Address          Domain    AllocationId      InstanceId(AssociationId)
-----
10.84.14.253     vpc       eipalloc-78d9a8c9  i-00000008(eipassoc-78d9a8c9)
```

Euca2ools CLI for Elastic IPs

The following **euca2ools** CLI commands are used to create, define, and delete elastic IPs:

- **euca-allocate-address**
- **euca-release-address**
- **euca-describe-addresses**
- **euca-associate-address**
- **euca-disassociate-address**

Euca2ools CLI for Route Tables

Route tables can be created in an Amazon VPC and associated with subnets. Traffic exiting a subnet is then looked up in the route table and, based on the route lookup result, the next hop is chosen.

The following **euca2ools** CLI commands are used to create, define, and delete route tables:

- **euca-create-route-table**
- **euca-delete-route-table**
- **euca-describe-route-tables**
- **euca-associate-route-table**
- **euca-disassociate-route-table**
- **euca-replace-route-table-association**
- **euca-create-route**

- **euca-delete-route**
- **euca-replace-route**

Supported Next Hops

The supported next hops for the current release are:

- Local Next Hop

Designating local next hop indicates that all subnets in the VPC are reachable for the destination prefix.

- Internet Gateway Next Hop

This next hop is used for traffic destined to the Internet. All virtual machines using the Internet gateway next hop are required to use an Elastic IP to reach the Internet, because the subnet IPs are private IPs.

- NAT instance

To create this next hop, the user needs to launch a virtual machine that provides network address translation (NAT) service. The virtual machine has two interfaces: one internal and one external, both of which are automatically created. The only requirement here is that a “public” network should have been provisioned by the admin, because the second interface of the virtual machine is created in the “public” network.

Internet Gateway Next Hop Euca2ools CLI

The following **euca2ools** CLI commands are used to create, define, and delete Internet gateway next hop:

- **euca-attach-internet-gateway**
- **euca-create-internet-gateway**
- **euca-delete-internet-gateway**
- **euca-describe-internet-gateways**
- **euca-detach-internet-gateway**

NAT Instance Next Hop Euca2ools CLI

The following **euca2ools** CLI commands are used to create, define, and delete NAT instance next hops:

- **euca-run-instances**
- **euca-terminate-instances**

Example: Creating a NAT Instance with Euca2ools CLI

The following example creates a NAT instance and creates a default route pointing to the NAT instance.

```
# euca-describe-route-tables
RouteTableId      Main      VpcId      AssociationId      SubnetId
-----
rtb-default       yes       vpc-8352aa59      rtbassoc-0c549d66      subnet-5eb34ed2

                Prefix      NextHop
                -----
                10.1.0.0/16      local

# euca-describe-images
IMAGE    ami-00000003      None (ubuntu)      2c88a895fdea4461a81e9b2c35542130
IMAGE    ami-00000005      None (nat-service) 2c88a895fdea4461a81e9b2c35542130

# euca-run-instances ami-00000005

# euca-create-route --cidr 0.0.0.0/0 -i i-00000006 rtb-default

# euca-describe-route-tables
RouteTableId      Main      VpcId      AssociationId      SubnetId
-----
rtb-default       yes       vpc-8352aa59      rtbassoc-0c549d66      subnet-5eb34ed2

                Prefix      NextHop
                -----
                10.1.0.0/16      local
                0.0.0.0/0        i-00000006
```